UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Posgrado en Ciencia e Ingeniería de la Computación

Análisis de Requerimientos asistido por computadora basado en Minería de Repositorios de Software y Usabilidad como medio para reflexionar el diseño de software en proyectos académicos descritos en documentos de titulación

T E S I S
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
MANUEL IGNACIO CASTILLO LÓPEZ

Directora de Tesis:
Dra. María del Pilar Ángeles
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

Ciudad Universitaria, Cd. Mx. Noviembre 2021

Agradecimientos

Sin un orden en particular, quiero agradecer al *Consejo Nacional de Ciencia y Tecnología* por la beca que me permitió enfocarme a realizar mis estudios de maestría y realizar este proyecto. Agradezco también al *Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas*; así como su personal docente y administrativo que hacen posible este programa de posgrado, el rigor y calidad del contenido académico es enriquecedor, interesante y formativo.

Quiero extender este agradecimiento a la *Escuela Nacional Preparatoria plantel 4*; en la que no solo me convertí en parte de la comunidad de esta Universidad, sino que además su excelente personal docente me ayudó a descubrir mi pasión por la lógica, las matemáticas y el cómputo. En esta misma línea, agradezco también a la *Facultad de Ciencias*, cuyos profesores me ayudaron y alentaron aún más a canalizar esta pasión como profesional de las Ciencias de la Computación. Sin estas dos entidades y aquellos que fueron (y aún considero) mis maestros, este trabajo posiblemente nunca hubiera tenido la oportunidad de ser.

Hago un agradecimiento especial a los integrantes del grupo *Espacios y Sistemas Interactivos para la Educación d*el *Instituto de Ciencias Aplicadas y Tecnología*. Su apoyo y motivación para continuar desarrollando mi vida académica; así como sus enseñanzas y oportunidades de colaboración, fueron elementos fundamentales para el desarrollo de este trabajo.

Agradezco también a mis colegas y amigos; muchos de los cuales conocí en las entidades de esta Universidad anteriormente mencionadas, quienes pese a las diversas adversidades que he tenido que enfrentar a lo largo de esta corta trayectoria académica y profesional, me han dado ánimos, motivos y enseñanzas que me han permitido alcanzar otra meta más en este sueño que creí inalcanzable por muchos años.

Finalmente, agradezco también a los programas *ESPORA Psicológica*; su apoyo fue vital para superar situaciones extraordinarias que tuve que enfrentar a la mitad de mis estudios de maestría durante el brote del SARS-CoV-2, sin esta ayuda no tendría certeza de alcanzar esta meta.

- El límite es la imaginación

Tabla de contenidos

Tabla de contenidos

Tabla de contenidos

Tabla de contenidos

1 Introducción

Análisis de Requerimientos asistido por computadora basado en Minería de Repositorios de Software y Usabilidad como medio para reflexionar el diseño de software en proyectos académicos descritos en documentos de titulación

La deficiencia en la documentación de proyectos de software ha sido identificada por varios autores como una de las causas por las que pese al ejercicio y desarrollo de la Ingeniería de Software, los proyectos de software en general enfrentan problemas que pueden limitar sus margenes de éxito y elevar sus costos de ejecución. En particular, los proyectos académicos de software suelen enfrentar problemas derivados de la falta de documentación, puesto que los equipos de trabajo suelen incluir o consistir de estudiantes que participan por tiempo limitado, mientras desarrollan sus proyectos de titulación.

Los alcances de estos proyectos y el grado de aplicación de principios de la IS en los laboratorios e institutos en los que se desarrollan estos proyectos pueden omitir la redacción, mantenimiento o revisión de documentos relacionados al diseño, necesidades y motivos detrás de los proyectos. Sin embargo, esta información suele estar presente en los documentos de titulación que producen los estudiantes. Por otro lado, recuperar información como casos de uso o requerimientos de software, a partir de estos documentos, puede ser una tarea ardua y se ha observado que suele consumir una parte importante del tiempo que tienen los estudiantes o laboratorios para realizar sus proyectos.

Una herramienta con la capacidad de extraer requerimientos de estos documentos de forma automática podría resolver el problema de recuperar esta información en poco tiempo, pero no necesariamente atendería el problema de entender y reflexionar la información recuperada para poder emplearla efectivamente. La Minería de Repositorios de Software (MRS) es un enfoque emergente de la Minería de Textos que se especializa en la Extracción de Información a partir de objetos producidos en proyectos de software, como puede ser código fuente, minutas de reuniones, reportes de fallas, Especificaciones de Casos de Uso, etc.

La aplicación de la Minería de Textos en este tipo de objetos, exhibe la particular dificultad de analizar texto que suele involucrar segmentos que no corresponden al lenguaje natural, y el uso de una jerga lingüística especializada. Estos factores han resultado en un bajo desempeño general de la Extracción de Información de las propuestas que se han presentado hasta ahora en la MRS. Con las técnicas y modelos del lenguaje disponibles en la actualidad, para extraer requerimientos a partir de documentos de titulación, se considera necesario un determinado grado de participación por parte de los usuarios.

Esta participación puede aprovecharse para abordar el problema de analizar la información que se extrae para poder darle un uso práctico. La propuesta que se presenta en esta tesis es una prueba de concepto de una Herramienta de Ingeniería de Software Asistida por Computadora que realiza la extracción de textos *candidatos* a ser requerimientos, para presentarlos a los usuarios ofreciéndoles un espacio para realizar un análisis asistido de requerimientos que resulta en la generación de un Especificación de Requerimientos de Software (ERS); cuyo formato y contenido se apega a estándares y recomendaciones de la Ingeniería de Software.

El prototipo basa su Extracción de Información en patrones gramaticales del español; inspirado en algunos de los proyectos con los resultados más alentadores para realizar tareas similares con textos en inglés. Además, su diseño fue centrado en los usuarios objetivo; estudiantes o académicos que participan en proyectos de software descritos en un documento de titulación. De tal forma, el presente proyecto cuenta con dos ejes principales: la MRS y la usabilidad.

El desempeño de la herramienta ha sido evaluado desde ambos enfoques y los resultados se han interpretado en conjunto para conocer la efectividad del prototipo; y determinar si es posible mejorar la documentación de proyectos académicos de software descritos en documentos de titulación, ofreciendo un espacio para la reflexión y construcción de ERS que permitan recuperar y hacer disponible a bajo costo la información en torno a los requerimientos.

1.1 Motivaciones

La documentación de un proyecto de software en desarrollo registra las necesidades y expectativas de los clientes, interesados y usuarios; actuando como una plataforma que permite estudiar las bases del proyecto, fomenta la transparencia entre dichos actores y los desarrolladores; además de ser la base de colaboración entre proveedores, consumidores y desarrolladores externos [1]–[5]. La documentación es útil en todas etapas del ciclo de vida de un proyecto de software, ya que permite entender y conocer las funciones del software, los escenarios para las que están previstas dichas funciones, las necesidades para las que fue diseñado, qué debe esperarse de otro software que lo pretenda reemplazar; entre otros aspectos [1]–[5].

Para que una documentación sea útil y abarque los puntos anteriores, debe satisfacer estándares y recomendaciones [2], [5]. También es importante considerar que los proyectos de software suelen evolucionar a lo largo de su ciclo de vida; por lo que la documentación producida en un momento específico, puede volverse inexacta e incluso obsoleta posteriormente. Por lo anterior, no solo es importante contar con una buena documentación, sino que además es indispensable mantenerla actualizada, de forma que refleje de manera concreta, específica y breve el proyecto de software y sus cambios [2], [4], [5].

Cómo señala Booch en [6], varios autores consideran que tras la llamada “crisis del software” en la segunda mitad de la década de los sesentas, se consolidó el campo de la Ingeniería de Software buscando atender una preocupante brecha entre las expectativas y funciones que se esperaban de un producto de software; y las que ofrecían en realidad [7]. A pesar de que el ejercicio y desarrollo de la Ingeniería de Software ha logrado reducir esta brecha a través de distintas estrategias, las cuales han evolucionado y diversificado en respuesta a los avances tecnológicos involucrados tanto en las herramientas de desarrollo, como en los objetivos de distintas clases de software; hoy en día los proyectos de software representan una empresa con márgenes de éxito inciertos, en parte debido a la falta de transparencia interna y externa en estos proyectos [6], [8]–[12].

Autores como Coscia et. al. [13], Sommerville [14], Visconti y Cook [15], Chomal y Saini [16], Martin y Martin [17] y Clements et. al. [18] sugieren que dicha incertidumbre de éxito está estrechamente relacionada con la ausencia o mala calidad de la documentación. De acuerdo con lo anterior, las consecuencias de la deficiencia de la documentación abarcan todo tipo de proyectos de software; incluyendo proyectos académicos de software. En este tipo de proyectos, es común que participen estudiantes que realizan prácticas profesionales o que se encuentran en proceso de titulación. Los estudiantes suelen rotar conforme terminan sus prácticas o su titulación; y sus roles en el proyecto son ejercidos por otros estudiantes.

Estos roles, suelen tener responsabilidades directas dentro de las etapas del ciclo de vida del software que abarque cada proyecto. Algunos estudiantes diseñan, implementan, adquieren, operan, mantienen; o realizan otras actividades directamente relacionadas con la producción de software. Los siguientes pares de proyecto de investigación y tesis son ejemplos de este tipo de actividades: [19] y [20], [21] y [22], [23] y [24], [25] y [26]; o bien, los siguientes proyectos académicos que involucran software e incluyen tesistas o practicantes de servicio social de acuerdo con los comunicados y convocatorias [27]–[30].

De acuerdo con [14], [16]–[18], esta rotación de personal dentro de proyectos de software no debería representar un problema; siempre y cuando se cuente con documentación suficiente que albergue el conocimiento de los miembros del equipo que se retiran. Sin embargo; en este tipo de proyectos académicos, es común que se omita la generación de documentos específicos del proyecto de software en cuestión, como pueden ser: Especificaciones de Casos de Uso, Arquitectura, Planes de Integración, Especificaciones de Requerimientos, entre otros.

Una herramienta con la capacidad de extraer información a partir de Tesis y Reportes de Servicio Social para generar Especificaciones de Requerimientos de Software, podría ayudaría a atenuar las consecuencias de la ausencia de documentación de software producido en proyectos que involucran la producción o mantenimiento de software. Además, se alinearía con recomendaciones y estándares que recomiendan la presencia de Especificaciones de Requerimientos en proyectos de software [1], [2], [5], [14], y también el uso de herramientas que asistan en la producción y actualización de esta documentación [1], [14], [31], [32].

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar un prototipo de herramienta de software que realice Extracción de Información contenida en Tesis y Reportes de Trabajo Social; cuyo contenido describa el desarrollo o mantenimiento de un proyecto de software. La herramienta asistirá la creación de Especificaciones de Requerimientos, considerando parcialmente estándares para este tipo de especificaciones; a fin de aportar conocimiento útil para el desarrollo de los proyectos de software relacionados con los documentos de entrada.

1.2.2 Objetivos específicos

Sin un orden prioritario específico; los objetivos específicos de este proyecto se enlistan a continuación:

1. **Desarrollar un prototipo de herramienta que tome Tesis y Reportes de Trabajo Social**; relacionadas con la producción o mantenimiento de un producto de software, **y realice Extracción de Información sobre dichas entradas, buscando extraer Requerimientos de Software**. Se deberán considerar métricas que permitan evaluar la calidad de la información extraída para refinar la calidad de los resultados.

2. **La salida del prototipo debe consistir en documentación textual en español, la cual debe describir los Requerimientos de Software extraídos durante su ejecución**. Para alcanzar este objetivo, se estudiarán estándares para este tipo de documentación de software; así como herramientas lingüísticas que asistan en la generación de texto con un grado reducido de ambigüedad.

3. **La salida del prototipo debe aportar conocimiento útil respecto a los Requerimientos que describe**. La presencia de documentos no asegura una mejora inmediata para los proyecto de software. El exceso de documentación, documentos poco detallados o demasiado informales, puede ser tan improductivos como no contar con documentación en absoluto; o puede tener otras consecuencias [17], [33], [34].

4. **El uso del prototipo, debe ayudar a los desarrolladores e interesados a entender los Requerimientos de Software extraídos; sin necesidad de realizar otras tareas intensivas**. El uso del prototipo debe cubrir aspectos básicos de usabilidad; es decir, el uso e interacción con el prototipo deben ser accesibles para los integrantes de los proyectos asociados con los documentos de entrada [2], [5], [14], [34]–[37].

1.3 Hipótesis

Es posible obtener documentación de proyectos académicos de software descritos en Tesis o Reportes de Servicio Social en una etapa de su ciclo de vida posterior a la implementación, utilizando técnicas de Procesamiento de Lenguaje Natural y de Extracción de Información; compilando Especificaciones de Requerimientos de Software que satisfacen de forma parcial estándares para este tipo de documentación, redactados en español y ofreciendo ventajas en los procesos de software de dichos proyectos.

1.4 Estructura del documento

La estructura de esta tesis es la siguiente: después de esta primera sección introductoria, se exploran los conceptos, métodos, estándares y recomendaciones que conforman el marco teórico con el que se ha desarrollado el prototipo de herramienta y los experimentos en los que se ejercitó para explorar los objetivos señalados en ésta sección. El marco teórico a su vez se subdivide en secciones que presentan la teoría por bloques: inicia con un panorama general, exhibiendo el trasfondo y trabajos relacionados en los que se presenta la *minería de repositorios de software* (MRS) y propuestas en dicha área respecto a Herramientas de Ingeniería de Software Asistida por Computadora (ISAC). Después se profundiza en la MRS para conocer las metodologías, técnicas y conceptos que se utilizan en dicha área. Posteriormente se presentan estándares para Especificaciones de Requerimientos de Software (ERS) y su aplicación en el desarrollo del prototipo. Finalmente, el marco teórico termina con una introducción a la usabilidad y su importancia en herramientas ISAC.

En la tercera sección, se presenta brevemente la construcción del prototipo. En esta sección se revisan distintas herramientas utilizadas en el área de la MRS, luego se presentan las consideraciones técnicas de la solución; cómo los lenguajes de programación y bibliotecas a utilizar, sus versiones, y las plataformas de desarrollo y de ejecución sobre las que se ha desarrollado el prototipo. Esta sección termina presentando la arquitectura general del prototipo, sus requerimientos y casos de uso.

El cuarto capítulo de esta tesis describe la experimentación con la que se probó el prototipo construido, la realización de dichos experimentos y las estrategias para recolectar datos; y finalmente el proceso de análisis de resultados, en el que se expone el grado de éxito alcanzado para cubrir los objetivos y la hipótesis presentados en este primer capítulo. Finalmente, este documento termina con un apartado de conclusiones, en las que se reflexiona sobre los resultados obtenidos respecto al marco teórico y los experimentos utilizados. Estas reflexiones cierran con comentarios respecto al trabajo futuro para continuar, alterar o mejorar los procedimientos seguidos a lo largo de este proyecto, con la intención de ofrecer una base para continuar con el esfuerzo de la producción de una herramienta ISAC que refine los resultados alcanzados.

2 Marco teórico

En esta sección, se exploran los conceptos, métodos, estándares y recomendaciones que conforman el marco teórico con el que se va a desarrollar el prototipo de herramienta; así como los experimentos en los que se va a ejercitar para explorar los objetivos señalados en ésta sección. El marco teórico a su vez se subdivide en secciones que presentan la teoría por bloques: inicia con un panorama general, exhibiendo el trasfondo y trabajos relacionados en los que se presenta la *minería de repositorios de software* (MRS) y propuestas en dicha área respecto a Herramientas de Ingeniería de Software Asistida por Computadora (ISAC). Después se profundiza en la MRS para conocer las metodologías, técnicas y conceptos que se utilizan en dicha área. A continuación se presentan estándares para Especificaciones de Requerimientos de Software (ERS) y su aplicación en el desarrollo del prototipo. Finalmente, el marco teórico termina con una introducción a la usabilidad y su importancia en herramientas ISAC.

2.1 Trasfondo y trabajos relacionados

En el dominio de la Inteligencia Artificial, se usa el término *minería* para referirse a aplicaciones de la Extracción de Información (junto con otros conceptos y metodologías) [38]–[41]. Además, se asocia con otros términos que indican el campo de aplicación específico de *minería*. Algunos de estos campos de aplicación son: la *minería de datos*; que se centra en descubrir patrones en grandes bases de datos [39]–[43], la *minería de textos*; que se distingue de la minería de datos por enfocarse en descubrir patrones en información textual y no en bases de datos [38], [39], [42], [43]; y la *Minería de Repositorios de Software* (MRS), que como sugiere su nombre, busca descubrir información en repositorios de proyectos de software [32], [44]–[46]. Las técnicas y conceptos utilizados los diversos enfoques de minería, son útiles entre ellos; por lo comparten una estrecha relación [32], [44]–[46].

La MRS es un área emergente que tiene al rededor de veinte años de práctica [32], [47]. Pese a que existen múltiples aplicaciones para este enfoque de minería, la generación, síntesis o compilación de documentación no han sido de las aplicaciones más exploradas, ya que estas aplicaciones implican retos específicos relacionados con el *Procesamiento de Lenguaje Natural* (PLN) [32], [44], [46]. Como en este trabajo se busca proveer Especificaciones de Requerimientos para proyectos académicos de software, mediante la aplicación de Extracción de Información sobre documentos derivados de la producción o mantenimiento de un producto de software; este trabajo pertenece al dominio de la MRS.

Algunos de los trabajos más recientes y relacionados con esta propuesta; incluyen el artículo de Ellmann [48], quien utilizó PLN y *Aprendizaje de Máquina* para buscar similitudes entre reportes de incidentes en *Sistemas de Seguimiento de Incidentes* (SSI). Parte de la premisa de que este tipo de reportes se presentan en grandes volúmenes, y su objetivo fue detectar reportes duplicados; pues por un lado, su presencia puede entorpecer los procesos del ciclo de vida del software, y por otro, omitir información relevante que solo aparece en determinadas instancias de las repeticiones.

Ellmann [48] encontró que la mayoría de los reportes duplicados aparecen durante el primer año del ciclo de vida; el número de defectos en el software incrementa la dificultad para detectar los duplicados y que los proyectos que utilizan estándares para realizar estos reportes, presentan una frecuencia menor de duplicados. De forma similar, Ardimento y Dinapoli [49] aplicaron PLN y *Aprendizaje de Máquina* sobre reportes de defectos en *Sistemas de Seguimiento de Defectos*; con el objetivo de predecir tiempos de compostura para reportes de defectos y asistir el proceso de Ingeniería de Software, puesto que la asignación de tareas de compostura de software requiere una inversión considerable de recursos.

Desafortunadamente, los autores tuvieron poco éxito en esta tarea [49]. Los reportes de defectos suelen contener una amplia variedad de atributos; como archivos anexos (es común encontrar imágenes y binarios adjuntos), descripciones

de casos de prueba, comentarios y discusiones textuales, fechas de reporte y actualización, descripción, palabras clave, severidad, prioridad, autor del reporte, días hasta la resolución, entorno de ejecución en el que se presentó el defecto, versión del software en que se presentó el defecto, usuarios etiquetados y estado del reporte. Puesto que resulta difícil trabajar con todos estos atributos en conjunto; especialmente con los que representan información binaria (como los adjuntos), los autores eligieron un subconjunto de estos atributos y consideraron ciertos valores constantes para algunos de ellos [49].

Aún con los atributos considerados, especificar las reglas bajo las que su implementación de *Aprendizaje de Máquina* debería clasificar los reportes entre compostura rápida y lenta; representó un reto que resultó en un clasificador sesgado para identificar tiempos de compostura rápidos, que al ser evaluado con conjuntos de reportes de prueba ofrecían resultados poco confiables. Pese a esto, sus resultados superan a los de trabajos previos con el objetivo de identificar tiempos de compostura en reportes de defectos [49].

Por otra parte, Mahadi, et. al. [50] aplicaron también PLN y Aprendizaje de Máquina; pero su objetivo fue extraer discusiones de diseño en repositorios de software libre, buscando identificar los componentes específicos del software entorno a los cuales giran las discusiones de diseño extraídas. Uno de los aspectos más relevantes de este trabajo, es la aplicación de *transferencia de conocimiento*; los autores realizaron la extracción de discusiones de diseño en un conjunto de datos y luego trataron de aprovechar la configuración resultante del entrenamiento de su clasificador para realizar extracción en conjuntos de datos diferentes. Si bien lograron entrenar un clasificador que mejora a otros para distinguir discusiones de diseño de entre los textos analizados; no tuvieron éxito en el segundo objetivo de utilizar ese mismo clasificador sobre conjuntos de datos diferentes [50].

Hirsch at. al. [51] propusieron una herramienta ISAC llamada *Gapidt*, la cuál tiene el propósito de identificar y reducir anti-patrones en documentos en *Lenguaje de Descripción de Servicios Web* (WSDL por sus siglas en inglés). Los anti-patrones son “soluciones comúnmente ocurrentes para problemas que generan consecuencias negativas” [52]. Los WSDL son un tipo de archivo basado en XML que describen un Servicio Web (SW), son relevantes como un componente del SW; pues son procesables por una computadora y actúan como una interfaz que le permite conocer a un *consumidor* o *cliente* del SW qué funciones ofrece y cómo debe comunicarse con él. Además, también son relevantes como documentación del SW; ya que incluyen descripciones legibles por humanos que explican las funciones que ofrece el servicio [53].

Los autores de *Gapidt* consideraron una alta correlación estadística entre la calidad de código Java de un SW y la calidad de sus respectivos documentos WSDL [13]. Su objetivo fue crear una herramienta; que como muchas otras, generara estos documentos de forma automática a partir del código fuente. Lo que hace *Gapidt* diferente a otras herramientas con este propósito, es su capacidad de refactorizar código fuente en Java de un SW y generar los documentos WSDL evitando introducir anti-patrones. Los autores utilizaron *Aprendizaje de Máquina* para identificar anti-patrones en el código y un sistema de reglas para extraer información y generar los documentos WSDL; evitando incluir o generar anti-patrones. *Gapidt* puede producir documentos WSDL de alta calidad sin necesidad de refactorizar el código [51]; la funcionalidad de mejorar la calidad del código es una ventaja adicional considerando buenas prácticas de desarrollo de software, como las que sugiere Martin [54].

Uno de los trabajos más afín la propuesta de esta tesis, es el artículo de Tiwari et. al. [55]; quienes crearon un prototipo de herramienta ISAC, *Text2UseCase*; que extrae Escenarios de Casos de Uso y genera Especificaciones de Casos de Uso (ECU) a partir de Especificaciones de Requerimientos (ER) en formato ISO/IEC/IEEE 29148:2011 sección 8.4, mediante la aplicación de PLN y *Aprendizaje de Máquina*. Para generar las ECU, después de que *Text2UseCase* haya extraído la información que conforma cada Caso de Uso de acuerdo a un clasificador, utiliza generación de lenguaje natural para

producir el documento de salida. La información en este documento debe ser revisada de forma manual primero por los usuarios de *Text2UseCase* [55].

Para esta revisión, la herramienta utiliza la técnica W^5H^2 [56] y genera un cuestionario en el que se le pregunta a los desarrolladores del software si la identificación de actores, flujos de eventos, dependencias y otros componentes de los Casos de Uso; es correcta o no, en cuyo caso pueden hacer modificaciones con texto libre. Al experimentar con esta herramienta; los autores encontraron que este paso de evaluación de la información extraída, ayuda a los desarrolladores a entender mejor los requerimientos y casos de uso del producto de software en cuestión. Por lo anterior, si bien *Text2UseCase* no es completamente automática, genera ECU de buena calidad y asiste en los procesos de diseño; al hacer reflexionar sobre la información del proyecto como se haya capturado en sus ER [55].

Los trabajos mencionados nos ayudan a entender mejor el tipo de aplicaciones y tareas que se desarrollan en la MRS. A continuación, exploraremos con más detenimiento la *minería de repositorios de software*.

2.1.1 Minería de Repositorios de Software – MRS

Tal cómo se indica en la sección anterior, la *Minería de Repositorios de Software* (MRS) es un enfoque de *minería*; que aplica diversas técnicas y estrategias de la Inteligencia Artificial, particularmente al rededor de la Extracción de Información, para descubrir información en Repositorios de software [32], [44]–[46]. Un Repositorio de software es un compendio de artefactos de software; entre los que podemos encontrar código fuente, documentación (Especificaciones de Requerimientos, de Casos de Uso, documentos de diseño, manuales de usuario, Especificaciones de Interfaces de Programación de Aplicaciones, etc), métricas de código, historiales de cambios, reportes de defectos; entre otros [44].

El término *Minería de Repositorios de Software* se está convirtiendo en la denominación preferida de este enfoque de minería, originalmente referido cómo *minería de datos de Ingeniería de software* [32], [44], [45]. Minar Repositorios de Software puede hacerse con objetivos variados; ya que por un lado contempla utilizar como entradas distintos componentes de un proyecto de software, y por otra parte no impone restricciones en el tipo de patrones que pueden ser descubiertos en dichos artefactos, ni el tratamiento que se le debiera dar a los patrones encontrados [32], [44]–[46].

Algunos objetivos de la MRS incluyen: reconocimiento de entidades de software, predicción de defectos, estimación de tiempos de compostura o lanzamiento, síntesis o generación de artefactos de software; así como generación de mensajes de confirmación para *Sistemas de Control de Versiones* [32], [44]–[46], [57]. La aplicación de técnicas y conceptos de la *minería de texto* es útil para descubrir información en artefactos de software basados en texto [44]. Sin embargo; en un Repositorio de software, existen artefactos de texto que no están escritos en lenguaje natural, principalmente el código fuente.

Utilizando *Procesamiento de Lenguaje Natural* (PLN), existe una notoria dificultad para procesar este tipo de artefactos de software [46], [58], [59]; por lo que en la MRS se pueden adecuar los conceptos, técnicas y herramientas del PLN para procesar Lenguajes de Programación [57], [59]. Por otro lado, los artefactos como los registros contenidos en un *Sistema de Seguimiento de Incidentes*; si bien pueden interpretarse cómo información textual, por contener amplias descripciones, comentarios y anotaciones, se encuentran en grandes colecciones como información estructurada o semi-estructurada; por lo que para trabajar con este tipo de artefactos, la MRS toma herramientas y conceptos de la *minería de datos* [32], [44], [46], [49]. La estructura del contenido de artefactos de código fuente y los registros de incidentes son ejemplos de las relaciones que existen entre la minería de datos, la minería de textos y la minería de repositorios de software.

2.2 Conceptos y metodologías utilizadas en la MRS

Los trabajos relacionados que exploramos en la sección 2.1, nos dan un primer panorama de las metodologías y herramientas que se utilizan para Extraer Información a partir de Repositorios de Software. A continuación se presenta la teoría en la que se fundamentan los procesos de MRS, explorando los fundamentos necesarios para posteriormente diseñar una herramienta para documentar Requerimientos de Software a partir de Tesis y Reportes de Servicio social propuesta en este proyecto (en la sección 1.2).

2.2.1 Proceso general de análisis de datos y descubrimiento de patrones

Tal como se presentó anteriormente, la *minería de texto* es un enfoque de minería en el contexto de la Inteligencia Artificial en la que se busca explorar patrones en información textual [38], [39], [42], [43]. Para realizar esta exploración, se aplican principalmente procesos en el dominio de la lingüística computacional; específicamente PLN, y técnicas de aprendizaje de máquina o clasificación de información [39], [60]. En general los procesos de minería suelen describirse mediante una secuencia de operaciones, las cuáles varían dependiendo del origen y calidad de los datos de entrada, los modelos y técnicas con los que van a analizarse y los resultados e interpretaciones esperados [61], [62].

El propósito de estas secuencias de operaciones es recolectar, manipular y obtener información de los datos a analizar; con la finalidad de asistir en las actividades de interpretación de los datos mitigando errores y acelerando el proceso de análisis. En cada operación de estas secuencias, los datos son manipulados en alguna forma persiguiendo los objetivos del análisis. Es recomendado almacenar la salida de las operaciones más significativas sobre los datos en la secuencia; de forma que puedan hacerse disponibles para análisis posteriores [61], [62].

En el caso de la minería de texto, la naturaleza no estructurada en tipos de datos sino basada en estructuras lingüísticas; son motivos por los que la primera parte de estas secuencias de operaciones sean aplicaciones del PLN. Esto permite identificar y exponer información implícita en los datos textuales y permiten un análisis automatizado basado en los resultados de estos análisis lingüísticos [38], [39]. En general, las secuencias de operaciones en procesos de minería incluyen las siguientes actividades [61], [62]:

1. **Obtención o recolección de datos** a partir de una o varias fuentes. Esta primera actividad hace disponibles los datos a analizar. Los datos pueden hacerse disponibles de forma continua por medio de un *flujo de datos* o pueden almacenarse en medios como bases de datos. A los almacenes de datos se les llama *lago de datos* (data lake). Una vez que se cuentan con los datos a analizar, son validados y registrados antes de la siguiente etapa [62].

2. **Preprocesamiento y limpieza de los datos**. Los datos primarios pueden presentar formatos diferentes, carecer de propiedades o atributos, o tener un formato poco apto para el análisis. En esta etapa, los datos primarios son modificados para darles un formato uniforme que permita un análisis eficiente. Para ello, suelen aplicarse análisis de calidad de datos, imputación, codificación, muestreo, reducción dimensional; entre otros. Los datos pueden ser etiquetados o categorizados tras ser preprocesados; con la finalidad de explorar patrones, utilizarlos en procesos predictivos o evaluar el desempeño general del proceso [62].

3. **Procesamiento o Exploración de los datos**. Consiste en descubrir distribuciones implícitas en los datos, patrones entre variables y correlaciones entre grupos o categorías de datos. En algunos casos puede resultar útil iterar esta operación sobre sus propios resultados.

4. **Modelado de los datos**. Esta etapa consiste formar modelos a partir de los resultados anteriores para exhibir la información extraída o *minada* de los datos. También suelen incluirse *funciones de pérdida*, con las que se estima el desempeño de la minería o del modelo resultante [61].

5. **Interpretación**. La operación de salida del proceso no suele ser automatizada, involucra los objetivos y valores de la organización, grupo o individuo que realiza o estructura el análisis para dar un sentido acorde a los modelos obtenidos; aún cuando los resultados no favorezcan estos objetivos o valores.

En las siguientes secciones, se presentan los conceptos y metodologías que se implementaron en el proceso de *minería de repositorios de software* del prototipo propuesto en este proyecto; denominado *Requex*. Más adelante se presentará la secuencia de operaciones en concreto en las que se basa el análisis de las Tesis y Reportes de Servicio Social para extraer requerimientos de software de estos documentos.

2.2.2 Procesamiento de Lenguaje Natural – PLN

Como se ha presentado en secciones anteriores, la *minería de repositorios de software* trabaja con fuentes de datos predominantemente en forma textual; ya sea que se trate de código fuente o documentación de un proyecto software. Específicamente, en este proyecto como se indicó en la secciones 1.1 y 1.2; se consideran Tesis y Reportes de Servicio Social producidos en proyectos académicos en la UNAM. Esto significa que en particular se trabaja con información textual en español; un lenguaje natural. En esta sección se presenta una de las dos herramientas fundamentales para ejercer un proceso de análisis sobre datos textuales en lenguaje natural: el *Procesamiento de Lenguaje Natural*.

El *Procesamiento de Lenguaje Natural* (PLN) es un enfoque de la *lingüística computacional*, que surgió con el objetivo de extraer y sintetizar la información textual en formato electrónico y redactada en algún *lenguaje natural* [63], [64]. Un *lenguaje natural* es aquel hablado y desarrollado de forma natural por alguna civilización; se distinguen de los lenguajes construidos artificialmente para desarrollar áreas conceptuales (como los lenguajes de programación o lenguajes formales) [63].

El PLN describe conceptos y mecanismos que permiten identificar la *estructura de un texto*, tales como: *palabras funcionales*, que son palabras que satisfacen las necesidades gramaticales de un lenguaje pero no aportan contenido significativo al sentido de los textos (como las conjunciones y las preposiciones), *palabras de contenido*, aquellas que dan sentido a un texto, *el rol que cumplen las palabras en el texto*, normalizar el texto *reduciendo las palabras a sus raíces* encontrando su origen morfológico (*lematización*), entre otros análisis [64]–[68].

Los modelos más utilizados en el PLN son estadísticos; ya que los modelos basados en reglas y restricciones para describir lenguajes resultan poco escalables, son costosos de producir y mantener, y fallan ante el extenso uso metafórico del lenguaje. El estudio y aprendizaje de las propiedades que contienen los textos en un lenguaje, se han utilizado para generar modelos estadísticos que han probado ser mucho más robustos, generalizables y con mejor desempeño ante textos naturales. Por otra parte, estos modelos también han generado un nuevo interés respecto a la adquisición humana del lenguaje [66].

De lo anterior se aprecian las herramientas del PLN que permiten implementar la etapa de preprocesamiento para el análisis de datos textuales, permitiendo incluso el etiquetado de la información contenida en los textos. En las siguientes secciones se presentan los conceptos y métodos dentro del PLN al rededor de estas tareas.

2.2.2.1 Tokenización

La tokenización consiste en dividir un texto en segmentos significativos, llamados *tokens*, los cuales usualmente se presentan en forma de palabras y signos de puntuación de acuerdo a la gramática y reglas de puntuación de cada lenguaje [67], [69]. Por ejemplo:

don’t → do + not Cd. Obregón → Cd + . + Obregón

2.2.2.2 Palabras vacías

Un conjunto de *palabras vacías*, es aquél formado por palabras que por su función o rol en las oraciones, su frecuencia o por su significado; aportan muy poco valor al sentido de los textos o pueden ser causa de errores en los modelos contemplados para el análisis que se esté realizando. Es importante distinguir las *palabras vacías* de las *palabras funcionales*; pues si bien en muchos análisis se usan de forma indiferente, el conjunto de *palabras vacías* depende de los objetivos del análisis que se esté realizando y en ocasiones se desea considerar todas o algunas palabras funcionales como palabras no vacías [66], [67], [69].

2.2.2.3 Lematización y extracción de raíces

La raíz de una palabra, puede ser una forma no necesariamente léxica de la palabra (es decir, reconocida en el lenguaje), la extracción de raíces consiste en identificar la raíz de la que esta deriva. Se obtiene removiendo los afijos; grupos no necesariamente contiguos de caracteres que pueden aparecer al inicio, al final o dentro de una palabra. Los afijos son definidos por construcciones del lenguaje cómo los diminutivos, formas plurales, negaciones, conjugaciones verbales regulares, formas despectivas, etc. La raíz también puede ser el origen morfológico de la palabra; conocidos cómo *lemas* [67]–[70].

Algunos ejemplos de raíces removiendo afijos son:

gotera → got política → politic desleal → leal

Y ejemplos de lemas:

caserón → casa durmiendo → dormir

2.2.2.4 N-gramas

Los n-gramas son grupos de tamaño n de distintos grupos de elementos como por ejemplo fonemas, sílabas, letras o palabras. En el caso de palabras, se toman de forma contigua en un texto a analizar para predecir la siguiente palabra con base en las anteriores. Por ejemplo, cuando n=2, se les conoce cómo bi-gramas; de acuerdo a lo anterior asisten a predecir qué palabra puede ser la consecutiva de otra. En el caso de que n=3, entonces tenemos un tri-grama, y asiste a predecir que palabra es la consecutiva dadas las primeras dos (en el orden original del texto). Predecir palabras con n-gramas es de gran utilidad en sistemas de análisis lingüístico basados en estadística, permiten estimar la estructura de un texto, identificando el rol y sentido de las palabras en el mismo de forma estadística [66], [68].

2.2.2.5 Reconocimiento de Roles Semánticos y Análisis de dependencias

Los roles semánticos son una forma de representar la estructura de las oraciones. Entender el rol semántico de una palabra nos permite obtener información contextual amplia del sentido de las oraciones. Los roles semánticos pueden describirse directamente, usando *agentes*; el objeto o persona que realiza alguna acción, *pacientes*; objeto o persona que se relaciona con la acción del *agente*, e *instrumentos* y metas que describen otras entidades y objetivos por medio de relaciones semánticas [66]–[68].

Otra forma de representar estas relaciones y los roles que tienen las palabras en las oraciones, es mediante la sintaxis. Determinando el sujeto, predicado, objetos y otros elementos de la oración, podemos deducir los roles que tienen; así como su relación con otras palabras. Puesto que las acciones y sus participantes varían de acuerdo al verbo y tiempo, se requiere cierta información al rededor de los verbos del lenguaje para poder determinar los roles del resto de las palabras en la oración [66], [67].

La oración se divide en sus componentes hasta obtener palabras; de acuerdo con las reglas sintácticas del lenguaje. Las relaciones sintácticas entre palabras suelen representarse en un árbol, cuyos arcos describen las relaciones gramaticales entre las palabras, los nodos a los que apunta; se dice que dominan a sus modificadores. Los aristas de estos árboles pueden ser decorados con etiquetas que indican el tipo de relaciones sintácticas [66], [67], [71].

Esto permite realizar inferencias simples, permitiendo también identificar ambigüedad y reconocer las funciones que cumplen las palabras de forma individual en las oraciones. Por ejemplo, si tenemos la oración *la compañía A compró a la empresa B*; podríamos querer responder a la pregunta *¿la empresa B fue adquirida?* Para identificar ambigüedad con árboles de dependencias, se determina si es posible construir más de un árbol de derivación sintáctica para la misma oración [66]–[68].

2.2.2.6 Reconocimiento de Partes de la Oración

Consiste en identificar el rol; muchas veces sintáctico, que tiene una palabra en una oración que emerge de la relación gramatical entre los *tokens* que conforman la oración. Esto es, identificar si una palabra es un verbo, sustantivo, adjetivo o pertenece a algún otro sintagma o categoría. Por ejemplo, distinguir un nombre propio de un sustantivo común; o distinguir un adjetivo calificativo de uno relacional, el tiempo y persona en el que está conjugado un verbo, etc [64], [66]–[68].

2.2.2.7 Reconocimiento de Entidades Nombradas

Una Entidad Nombrada es *un objeto del mundo real* que posee un nombre específico; por ejemplo, una persona, un país, un producto, una organización, etc. Recordemos que puesto que los modelos son estadísticos, dependen de los ejemplos en los que hayan sido entrenados y pueden necesitar ajustes para cada caso de uso [67], [69].

2.2.2.8 Modelos vectoriales de texto

Consisten en interpretar vectores como una forma de proyectar palabras en un espacio matemático, conservando la información en las palabras. Una de sus principales utilidades es para determinar similitud a partir de distancia entre los vectores; lo que a su vez resulta de gran utilidad para determinar tópicos, conjuntos de palabras que tienen un significado similar. Los algoritmos de aprendizaje de máquina utilizan estos vectores para hacer predicciones [66], [67].

2.2.2.9 Herramientas lingüísticas adicionales para el análisis de dependencias

Cómo se menciona en la sección 2.2.2.5, una de las estrategias para realizar el análisis de dependencias es identificar la estructura gramatical de las oraciones; los modelos en los que se basa la predicción de estas estructuras, usualmente se codifican en corpus o en herramientas. Pueden ser construidos a partir de obras literarias en el dominio público o documentos especializados; como pueden ser registros médicos, textos de una ingeniería particular o documentos legales y transcripciones de litigios [72].

Sin embargo, cuando se trabaja en un dominio del lenguaje cuyos modelos gramaticales no han sido ampliamente explorados; las predicciones de las estructuras de los textos puede ser no óptima [58]. Para atender este problema; algunos autores cómo Thakur y Atul [60], Tiwari et. al [55] o Pollock et. al [59]; sugieren aprovechar las estructuras gramaticales en torno a los verbos. Para este fin, una estrategia recurrente es aludir a los los *patrones de verbos de Hornby*; los cuales describen patrones que emergen en oraciones en inglés en relación con el verbo de la oración [73]. Originalmente, Hornby et. al. [74] describieron 25 de estos patrones a partir de tres factores [73]:

1. El lenguaje contiene muchos patrones. Usualmente cada palabra está asociada con un número pequeño de patrones sintácticos.

2. La comunicación ordinaria día a día consiste de declaraciones basadas en patrones de uso, construidas al rededor de un número bastante reducido de palabras frecuentes; que se emplean en un número también pequeño de patrones o estructuras. Al mismo tiempo, el uso comprende un gran número de otras posibles palabras y estructuras; algunas de las cuales casi no son utilizadas.

3. El verbo es el pivote de una cláusula. La información en cada cláusula puede determinarse con respecto al verbo; tal como podemos ubicar el sujeto, objeto o complementos antes, después o dentro de las frases verbales o incluso determinar si estos elementos están presentes o son implícitos de acuerdo al verbo. De esta forma el verbo representa un elemento central para analizar la estructura semántica de una oración; y con ello determinar el rol y sentido que adquieren las palabras que la conforman [64].

Estos patrones de verbos consisten en dos o más palabras (argumentos) en una cláusula. Si los sustantivos, adjetivos y otras palabras que conforman cada argumento de un verbo en una cláusula se asignan con el tipo semántico correcto, entonces el significado de la cláusula en su totalidad puede ser identificado con suficiente criterio. Los significados son expresados como implicaciones pragmáticas en lugar de implicaciones lógicas o semánticas, ancladas al patrón correspondiente [73].

A pesar de que existen críticas y discusiones al rededor de los motivos y factores que motivaron la creación de estos patrones para el inglés; tanto en la docencia ELE (Español como Lengua Extranjera), como en la pedagogía al rededor de la lengua española, la organización de los currículum y planes de estudio no contemplan herramientas similares a los patrones de verbos en el inglés. En su lugar, se puede apreciar un enfoque que busca instruir de forma exhaustiva las reglas del lenguaje; no necesariamente creando cohesión entre los distintos aspectos de la lengua [73], [75].

Pese a lo anterior, existe una serie de “patrones de gramática léxica; unidades de uso fraseológica asociada al léxico de forma no accidental” [75] reconocidos en el español. Estos patrones podrían describirse de forma similar a los Patrones de verbos de Hornby, utilizando los componentes de cláusulas en español. Uno de los planes de estudio de ELE que mejor incorpora este tipo de patrones, es el Plan Curricular del Instituto Cervantes (PCIC); el cuál cubre una amplia clasificación de componentes léxicos [75].

Uno de los recursos literarios más utilizados en ELE basado en el PCIC es el libro “Gramática básica del estudiante de español” [75]. A continuación se exponen los patrones que incluye el libro en su sección 4 “Verbos” [76] que son potencialmente útiles en un proceso de PLN para formar el tipo de cláusulas que se esperan encontrar en una ERS (las cuáles se describen en la sección 2.3.4, y que se han observado en documentos de titulación que describen proyectos académicos de software; tal como se describe en la sección 3.3.1.2).

2.2.2.9.1 Patrones verbales en formas no personales

2.2.2.9.1.1 Patrones en uso como adjetivo

El participio es la forma no personal de un verbo que le permite funcionar como adjetivo. Un participio siempre concuerda en género y número con el sustantivo al que se refiere. Se describen los siguientes patrones a partir del uso del participio en una oración [76]:

- Sustantivo + Participio.
- Verbo + Participio.

2.2.2.9.1.2 Patrones en uso en las formas compuestas de un verbo

Cuando el verbo auxiliar **haber** se usa con un participio, se construye una forma compuesta del verbo participio, las cuales se describen con los siguientes patrones en una oración [76]:

- Pretérito perfecto: Haber [Verbo/Presente] +Participio.
- Pretérito pluscuamperfecto: Haber [Verbo/Imperfecto] +Participio
- Futuro perfecto: Haber [Verbo/Futuro] +Participio

2.2.2.9.2 Patrones en usos para predecir el futuro

Cuando usamos el Futuro para hablar del futuro cronológico estamos prediciendo cómo va a ser el Futuro o solicitando predicciones a otros. Si queremos presentar la información más objetivamente, usamos el siguiente patrón [76]:

- Ir [Verbo/Presente, cualquier persona] + a [preposición] INF

2.2.2.9.3 Perífrasis verbales

Hablamos de perífrasis verbal cuando un verbo transmite una acción concreta acompañado de otra palabra bajo una conjunción. Podemos expresar necesidad u obligación de dos maneras diferentes usando la perífrasis verbal mediante los siguientes patrones [76]:

- Para indicar explícitamente quién tiene la necesidad u obligación usamos:

  Tener [Verbo/cualquier tiempo, cualquier persona] + que [conjunción] + INF

- Sin indicar quién tiene la necesidad u obligación:

  Haber [Verbo/cualquier persona] + que [conjunción] + INF

Si queremos referirnos a un momento muy concreto del progreso de una acción, usamos [76]:

- Estar [Verbo/cualquier persona, cualquier tiempo] + GERUNDIO

En el patrón anterior, la perífrasis se obtiene por el hecho de que el verbo “Estar” no puede transmitir el progreso de la acción por sí mismo. Otra forma de expresar un estado, usamos verbos sin perífrasis; es decir, el estado se entiende sin necesidad de conjuntar el verbo que indica estado con otro que refleja el estado [76].

2.2.3 Extracción de información textual

De acuerdo con la sección 2.2.1, en el proceso general para el análisis de textos; las primeras etapas del proceso aprovechan las herramientas que ofrece el PLN para preparar las siguientes, en las que se exploran los datos para identificar patrones y extraer información. En esta sección, se presenta la teoría en torno a los procesos de extracción de información que conforman la secuencia de operaciones para el prototipo que se construye en este proyecto.

La *extracción de información* consiste en reconocer relaciones clave en el texto, mediante la exploración de estructuras esperadas en el texto (usualmente utilizando los resultados del *análisis de dependencias*). Después de analizar el texto, se toman las partes que se hayan considerado significativas y se preparan en un formato útil para los objetivos de cada ejercicio de minería de texto [42]. La identificación de estas estructuras puede llevarse a cabo por medio de distintas técnicas.

Algunas de las más populares se basan en procesos estadísticos; cómo las técnicas de agrupación y las técnicas de etiquetado predictivo. Estas técnicas requieren una etapa de *entrenamiento*, en la que modelos estadísticos de las estructuras deseadas en el texto se ajustan identificando patrones en un conjunto de datos de entrenamiento; el cual debe ser suficientemente grande como para considerarse representativo del dominio del problema particular para el que se implementa la minería de texto [77], [78].

Otras técnicas se basan en reglas, las cuales identifican valores y atributos en el texto para determinar los segmentos de un texto de entrada que contienen las estructuras para las que se implementa el análisis. Estos análisis pueden ser más difíciles de mantener que los estadísticos; pero resultan convenientes cuando las estructuras que se buscan identificar son limitadas, tienen características muy bien determinadas o no es factible construir un conjunto de datos de entrenamiento representativo [78].

2.2.3.1 Métricas para evaluar el desempeño de la Extracción de Información

En general, estas evaluaciones emplean estadísticas que cuantifican la cantidad de resultados positivos o la cercanía de grupos de datos en un espacio de resultados. Los mecanismos para realizar este proceso varían con las técnicas que se utilicen para realizar la extracción; como también dependen de la naturaleza de los datos con los que se esté trabajando. Una de las métricas más empleadas es la *tasa de error*; que no es más que la proporción de errores respecto al total de instancias correctas [79].

Cuando se dispone de una gran cantidad de datos que fácilmente puede alimentar el proceso de extracción, una forma de evaluar el desempeño de dicho proceso es crear varios conjuntos de prueba con particiones de estos datos y ejecutar el proceso; al final es posible utilizar métricas estadísticas sobre los resultados para estimar el grado de éxito del proceso. Es importante considerar que los conjuntos de prueba no deben contener datos de entrenamiento; en su lugar se prefiere que estén conformados por datos que no se hayan presentado previamente al sistema. Esto permite una evaluación más robusta; permite descubrir si el proceso de extracción ha sido sobre-entrenado (tiene un desempeño alentador con datos de entrenamiento, pero muy pobre con otros datos) y predecir el comportamiento general del sistema con datos independientes al proceso de desarrollo [79].

Otra consideración importante es *estratificar* el muestreo de los datos, esto significa elegir los conjuntos de datos con los que se va a trabajar, de manera que resulten suficientemente representativos del universo de los datos en el dominio del proceso de extracción de información. En caso que se disponga de un conjunto de datos poco numeroso o de alto costo de preparación para alimentar el proceso de extracción, puede ser difícil garantizar esta representatividad puesto que los conjuntos de datos con los que se desarrolla el sistema y los de prueba no deben compartir elementos; por lo que en estos casos puede ser aceptable utilizar conjuntos no representativos en su totalidad (pero es importante mantener estos conjuntos como ajenos) [79].

Cuando la proporción de estos conjuntos de datos es 50-50 (una mitad para procesos de desarrollo y otra mitad para pruebas), una forma de estimar el desempeño es promediar la *tasa de error* que produce la extracción sobre ambos conjuntos de datos; no se recomienda seguir esta estrategia a menos de que la cantidad de datos con la que se trabaja sea realmente escasa o difíciles de obtener. Una mejor estrategia que varía de la anterior es la *validación cruzada de k-iteraciones*; que consiste en dividir los datos con los que se dispone para trabajar (tanto para entrenamiento, como validación y prueba) en *k* particiones y realizar la evaluación de la extracción de información rotando el papel de las *k* particiones [79].

De esta manera, se ejecuta el proceso de entrenamiento y prueba completo *k* veces; de forma que cada partición finge como conjunto de entrenamiento y prueba una vez. Al final, se promedian los *k* valores obtenidos para la *tasa de error* para calcular el desempeño del proceso. Estudios sugieren que usar 10 iteraciones (k = 10) produce una validación robusta del desempeño; sin embargo, entre más pequeño sea el conjunto de datos disponible, se sugiere utilizar un numero mayor de iteraciones [79].

El proceso de extracción puede entenderse como un problema de clasificación. En particular, puede entenderse como el problema de decidir si un elemento en un conjunto de datos cumple o no una propiedad; lo cuál puede decidirse explorando sus atributos (determinando a su vez si los atributos de cada dato cumplen o no con determinadas

propiedades y finalmente realizando un consenso para el elemento en su totalidad). El resultado de la clasificación puede estudiarse como una de cuatro posibilidades: *verdaderos positivos* (VP), *verdaderos negativos* (VN); que corresponden con clasificaciones correctas, *falsos positivos* (FP) y *falsos negativos* (FN); que corresponden con clasificaciones incorrectas [79].

La *tasa de verdaderos positivos* se calcula cómo el número de VP clasificados entre el total de clasificaciones positivas reales (que es la suma de VP más los FN). Por su parte, la *tasa de falsos positivos* es el número de FN clasificados entre el total de clasificaciones negativas reales (la suma de FP más los VN). Finalmente, la tasa de éxito general de la clasificación es el total de clasificaciones positivas obtenidas entre el total de clasificaciones [79].

$$T_{éxito} = \frac{VP + VN}{VP + VN + FP + FN}$$

La *tasa de error*, se obtiene como el complemento de la tasa de éxito $T_{error} = 1 - T_{exito}$ [79]. Observe que si todas las clasificaciones fueran correctas, FP = FN = 0 y entonces la tasa de éxito sería 1, mientras que la tasa de error sería 0. Por lo tanto, un clasificador de buen desempeño tiene una tasa de éxito muy cercana a 1 sobre conjuntos de datos de prueba representativos del dominio de datos sobre el que opera. Para problemas que clasifican en más de dos clases complementarias, se utilizan otro tipo de métricas [79].

Otro par de métricas útiles para evaluar el desempeño de un clasificador son la *precisión* y la *exhaustividad*. La *precisión* se calcula cómo el número de VP entre el total de clasificaciones positivas arrojadas por el clasificador [79]:

$$precisión = \frac{VP}{VP + FP}$$

Mientras que la *exhaustividad* es equivalente a la *tasa de verdaderos positivos* descrita anteriormente [79]:

$$T_{VP} = exhaustividad = \frac{VP}{VP + FN}$$

2.2.4 Metodologías de la MRS elegidas para el desarrollo del prototipo

El proceso de minería de los documentos de entrada del prototipo para extraer requerimientos de software, se constituye por dos etapas principales. Empieza por una secuencia de análisis de textos de basada en los conceptos explorados alrededor del PLN, está secuencia de tareas tiene como objetivo etiquetar las *Partes de la Oración* en los documentos de entrada y realizar un *Análisis de Dependencias*.

Posteriormente, se realiza el proceso de Extracción de información, que toma la información resultante del análisis textual anterior y extrae Requerimientos de Software de forma que pueda ser evaluado el desempeño de la extracción. Este proceso de extracción se basa en reglas, de forma que el prototipo *Requex* pueda ser desarrollado sin tener que poner demasiada atención en el diseño del proceso de extracción (en particular la recolección de documentos de entrada suficientemente representativos); la intención es priorizar los aspectos relacionados al análisis de requerimientos de software para compilar ERS y ofrecer un diseño modular que permita evolucionar la herramienta sin incurrir en altos costos de mantenimiento o soporte especializado.

Además; en proyectos de MRS similares al que se presenta, es común que se utilicen sistemas de extracción basados en reglas como ocurre en las siguientes publicaciones: [55], [60], [80]; como se señaló en la secciones 2.1 y 2.1.1, en buena medida esto se debe a las dificultades particulares dentro de la MRS. Más adelante, se presentan las tareas y estrategias de implementación concretas para implementar este proceso de minería, detallando la secuencia de tareas de PLN y las

reglas que conforman el proceso de extracción; así como los mecanismos concretos de evaluación. Todos estos mecanismos serán basados en la teoría que se presenta en este capítulo.

2.3 Estándares para Especificaciones de Requerimientos de Software – ERS

Esta sección presentan estándares y recomendaciones para Especificaciones de Requerimientos de Software, con los cuáles se diseñarán los criterios que deberán cumplir los documentos generados por *Requex*; de forma que la salida de la herramienta abarque características que la hagan útil e informativa bajo un alcance y formato adecuado para reconocerse cómo ERS.

Una ERS es la especificación de un producto de software, programa de computadora o conjunto de programas que realizan determinadas funciones en ambientes específicos; en otras palabras, especifica un producto de software. Un software desarrollado a partir de una ERS que no expresa; por ejemplo, los mensajes de error que deben utilizarse, difícilmente satisfará al cliente o interesados. Pueden ser escritas por uno o más representantes del proveedor de software, uno o más representantes del cliente o ambos; en todo caso se recomienda una estrecha colaboración entre ambos agentes para el desarrollo de una ERS [14], [35], [81], [82].

2.3.1 ISO/IEC/IEEE 29148:2018

Este estándar internacional describe extensivamente un formato recomendado para una ERS. Es respaldado por más de 35 años de evolución, teniendo origen con el estándar americano ANSI/IEEE 830-1984 [81], el cuál tuvo su última revisión en 2009 con el estándar ANSI/IEEE 830-1998 Rev 2009 [82]; fue finalmente reemplazado por el estándar internacional ISO/IEC/IEEE 29148:2011 [83], que es la versión anterior de este estándar [35].

El estándar 29148 fue creado a partir de las revisiones de los estándares ISO/IEC/IEEE 15288 y 12207 [35]. El estándar internacional 15288 describe el ciclo de vida de un producto de software, estableciendo una serie de procesos y terminología; con el objetivo último de alcanzar la satisfacción de los interesados [84]. Por su parte, el estándar 12207 establece un marco de trabajo común para procesos del ciclo de vida de un software con terminología bien definida. Apela a la adquisición de productos y servicios de sistemas o software y porciones de software de un sistema [5].

De esta forma, el ISO/IEC/IEEE 15288 describe el ciclo de vida de un software y los procesos de ingeniería de software al rededor de este ciclo, mientras que el estándar 12207 establece un marco de trabajo que los proveedores de software pueden seguir para conformar estos procesos de ingeniería de software; pero es muy general y no entra en todos los detalles de los diferentes aspectos de dichos procesos. Por su parte, ANSI/IEEE 830-1998 atiende los procesos de levantamiento, documentación, aplicación y seguimiento de requerimientos descritos en el estándar 12207 [82]; lo que permitió una transición a estándar internacional y dio origen al estándar 29148.

Cada requerimiento debe resolver un problema, alcanzar un objetivo o satisfacer la necesidad de un interesado, debe poder ser evaluado bajo condiciones mensurables, limitado a sus restricciones, define su desempeño con base en el sistema y no respecto a sus usuarios y debe poderse verificar. Puesto que representan las funciones que deberá implementar el software, los requerimientos y sus restricciones deben indicarse con la palabra ‘deberá’; y debe evitarse usar esta palabra u otras que se puedan interpretar como una obligación del producto de software en el contenido de la ERS que no represente requerimientos. Además, se debe procurar utilizar sentencias positivas y evitar el uso de negativos cómo ‘no deberá’ y evitar también utilizar una voz pasiva [35].

Además, cada requerimiento debe ser no ambiguo; es decir, debe tener una única interpretación y debe ser completo, especificando todas las capacidades, características, restricciones o factores de calidad que deba cumplir. Por su parte,

la ERS en su totalidad debe ser consistente: dos requerimientos no pueden presentar conflicto, utilizar sistemas métricos homogéneos y describir de la misma forma a los mismos objetos. Los requerimientos deben indicar qué se necesita; no cómo obtener esas necesidades. Su descripción debe ser clara y concisa, evitando ambigüedades del lenguaje [35], [81].

Algunos de los atributos que se sugiere incluya un requerimiento, son los siguientes [35]:

- Identificador.
- Número de versión.
- Propietario (persona o entidad responsable de mantener el requerimiento).
- Prioridad determinada por sus interesados.
- Riesgo.
- Motivaciones.
- Dificultad.
- Tipo. Por ejemplo: funcional (una tarea que debe desempeñar el software), de interfaz (una característica que debe cumplir para interactuar con otro sistema, componente o usuarios), de calidad o no funcional (una cualidad o restricción debe satisfacer el software), de usabilidad (una consideración para atender las necesidades de los usuarios), de factor humano (una consideración en favor de limitaciones, salud y condiciones humanas).

Las cláusulas específicas de requerimientos en una ERS pueden ser organizadas de acuerdo a un consenso entre los interesados del sistema, y en conformidad con los métodos auxiliares de la organización proveedora de software que ayuden a entender los requerimientos. No existe una organización óptima para describir un sistema; algunos ejemplos de estrategias para organizar los requerimientos en una ERS son los siguientes [35]:

- Modo del sistema – Algunos sistemas se comportan de forma diferente dependiendo en el modo de operación. Por ejemplo, un sistema de control puede tener conjuntos diferentes de funciones dependiendo de su modo: entrenamiento, normal, degradado o emergencia.
- Clase de usuario – Algunos sistemas ofrecen funciones diferentes a distintas clases de usuarios. Por ejemplo, un control de un elevador presenta capacidades distintas a pasajeros, personal de mantenimiento y personal de servicios de emergencias.
- Objetos – Los objetos son entidades del mundo real que tienen un equivalente en el sistema. Por ejemplo; en un sistema para monitorear pacientes, los objetos incluyen pacientes, sensores, enfermeras, habitaciones, médicos, medicinas, etc. Cada objeto posee un conjunto de atributos y funciones; también conocidas como servicios, métodos o procesos.
- Características – Son servicios que se desean ofrezca el sistema, pueden requerir una secuencia de entradas para producir el resultado esperado. Por ejemplo, en un sistema telefónico, las características incluyen llamadas locales, transferencia de llamadas y llamadas en conferencia. Cada característica es usualmente descrita como un par de estímulo-respuesta.
- Estímulos – Algunos sistemas pueden organizarse mejor describiendo sus funciones en términos de estímulos. Por ejemplo, las funciones de un sistema de aterrizaje automático de una aeronave pueden ser organizadas en secciones como pérdida de potencia, cizalladuras (diferencias locales entre la velocidad o dirección del viento),

cambios en el alabeo (movimientos giratorios sobre el eje longitudinal de la aeronave), velocidad vertical excesiva, etc.

- Respuesta – Algunos sistemas se organizan mejor al describir sus funciones como soporte de la generación de una respuesta. Por ejemplo, las funciones de un sistema de personal pueden ser organizadas en secciones correspondientes a: las funciones asociadas con la generación de pagos de nómina, las funciones asociadas con la generación de listas de empleados, etc.
- Jerarquía funcional – Si ninguna de las organizaciones anteriores resulta útil, las funciones pueden organizarse en una jerarquía de funciones organizada por entradas comunes, salidas comunes o accesos internos de datos. Pueden emplearse diagramas de flujos de datos y diccionarios de datos para mostrar las relaciones entre las funciones y los datos.

2.3.2 ISO/IEC/IEEE 15289:2019

Este estándar internacional indica el contenido que se espera encontrar en los distintos documentos que se producen en las etapas del ciclo de vida de un software [4]. De forma similar al ISO/IEC/IEEE 29148, atiende los procesos descritos en los estándares internacionales 12207 y 15288; describiendo los documentos que se espera obtener en ellos. Puesto que nos interesa conocer el contenido recomendado específicamente para ERS, se presentan únicamente a las secciones que describen el contenido esperado en este tipo de documentación.

2.3.2.1 Sección 7.8 – contenido genérico de especificaciones

Esta sección describe el contenido que en general se espera encontrar en un documento de especificaciones genérico; por lo que aplica a documentos cómo Especificaciones de Casos de Uso, Especificaciones de Diseño o Especificaciones de Requerimientos de Software. Una especificación es un elemento informativo, que identifica de forma completa, precisa y verificable alguna o varias características de un sistema, servicio o proceso [4].

Las definiciones, terminología y restricciones en una especificación deben ser consistentes. Además, una especificación no debe definirse por duplicado; de lo contrario se vuelve propensa a cambios y usos inconsistentes. Se sugiere incluir la siguiente información en una especificación [4]:

- Fecha de emisión y estado.
- Alcances.
- Responsable de emisión.
- Referencias.
- Autoridad que otorga visto bueno.
- Contenido.
- Requerimientos de garantía.
- Condiciones, restricciones y características.
- Glosario.
- Historial de cambios.

2.3.2.2 Sección 10.60 – Especificaciones de Requerimientos de Sistemas

En esta sección, encontramos una descripción del contenido que se espera encontrar en una *Especificación de Requerimientos de Sistemas*; la cuál también es aplicable a ERS, pues de acuerdo al mismo estándar una ERS es un tipo particular de *Especificación de Requerimientos de Sistema*; estos pueden suelen más allá del software y describir los entornos de ejecución, su configuración y otros aspectos más generales [4].

En primer lugar, se recomienda separar los requerimientos de los interesados (o de servicio) de los requerimientos de software en sí mismos. Los requerimientos de los interesados definen un sistema o servicio que debe satisfacer las necesidades de sus usuarios y otros interesados bajo un entorno bien determinado; el cuál incluye sus deseos, expectativas y restricciones (tales cómo consecuencias, acuerdos existentes, así como decisiones administrativas y técnicas). En estos requerimientos encontramos las métricas de efectividad para las necesidades clave del sistema [4].

Además, es importante considerar que existen otro tipo de requerimientos generales; como los requerimientos de negocio, de la organización y de usuario. En general, los elementos que se incluyen en una ERS son los siguientes [4]:

a) Especificaciones técnicas del sistema que describe.

b) Ingeniería de factores humanos, ergonómicos o requerimientos de usabilidad establecidos.

c) Funciones a nivel del sistema y características de calidad.

d) Requerimientos de seguridad.

e) Requerimientos de empaquetado, distribución, adaptación al destino, instalación, mantenimiento y retiro.

f) Documentación de capacitación e información para los usuarios.

g) Restricciones críticas máximas y mínimas de desempeño, suposiciones y dependencias.

h) Requerimientos para interfaces internas y externas con otros sistemas, hardware, software, servicios de comunicación y usuarios humanos.

i) Pruebas del sistema, calificaciones y requerimientos de aceptación.

j) Referencias al diseño del sistema, usabilidad, pruebas y estándares de seguridad.

k) Precedencia y criticidad de los requerimientos.

l) Seguimiento de los requerimientos con las necesidades de los interesados; así como con las funciones o elementos del sistema.

Adicionalmente, pueden incluirse requerimientos de infraestructura y sistemas habilitadores; incluyendo recursos y herramientas [4].

2.3.3 Otras recomendaciones para ERS

A continuación se revisan otras recomendaciones para este tipo de documentación que aporten otras consideraciones importantes para los requerimientos de software. El objetivo es contar con una descripción extensa de la información que debe considerarse en un requerimiento y los documentos en los que se compilan; de forma que al diseñar tanto el proceso de extracción de requerimientos cómo el de generación del documento de salida del prototipo, puedan considerarse distintos aspectos que permitan cubrir los atributos que se elegirán para dicha extracción.

Además de los atributos indicados en ISO/IEC/IEEE 29148 que conforman un requerimiento, también se sugiere incluir un **estado** para conocer la situación de los requerimientos a lo largo de su ciclo de vida [2], [3]. Estos estados son acordados por el equipo de desarrollo y pueden ser tales cómo *en progreso*, *en revisión*, *suspendido* o *terminado*. También podemos considerar los estados que sugiere Essence [3]:

- *Identificado* – Se ha encontrado una condición específica que debe satisfacer el software y se registra cómo requerimiento. Sin embargo, es descrito a un alto nivel.
- *Descrito* – El requerimiento ha sido descrito de forma clara, concisa, completa, consistente y verificable. Además, es justificado como necesario y alcanzable; además de ser priorizado.
- *Implementado* – El requerimiento ha sido implementado como parte del sistema en desarrollo.
- *Verificado* – Se ha confirmado que el sistema implementa exitosamente el requerimiento.

Otra dimensión sugerida para los requerimientos, es la **volatilidad** con la que se estima cuanto puede cambiar a lo largo de su ciclo de vida un requerimiento de software [2], [36]. Por otro lado, también se sugiere utilizar un lenguaje con una semántica bien definida, procurando utilizar descripciones técnicas o semi-técnicas para describir los requerimientos de la forma más precisa posible [2], [36]; esto nos ayuda a evitar ambigüedad [2], [34]–[36].

Por último, también es recomendado que los requerimientos sean inspeccionados por un grupo de desarrolladores e interesados con poder de decisión sobre el contenido de la ERS [2], [14], [17], [34], [36]; el objetivo de la revisión es detectar errores, suposiciones incorrectas y falta de claridad.

2.3.4 Aplicación de los estándares en el desarrollo del prototipo

De acuerdo con los estándares y recomendaciones revisados en las secciones anteriores, se describen los elementos que conformarán la salida que producirá el prototipo *Requex*. Se describe el formato y contenido de la salida del prototipo; así como las dimensiones que conformarán a los requerimiento que incluya dicha ERS de salida. Puesto que la salida será definida a partir de estos estándares y recomendaciones, se espera que aporten conocimiento útil respecto a los Requerimientos que describe y además deberían poderse entender por sí mismos.

Para empezar, el estándar internacional 15289 [4] sugiere una lista de dimensiones y propiedades que se esperan encontrar en una ERS. De acuerdo con este estándar, el *alcance* de los documentos de salida se considera dentro un solo sistema de software; supondremos que cada Tesis y Reporte de Servicio Social describe un único proyecto de software del que se extraen requerimientos. La salida que producirá el prototipo se enfocará en requerimientos y excluye manuales, motivaciones y documentación para el control de calidad. Estas dimensiones se agruparán como *Información general*, en la que se agregaran dimensiones adicionales sugeridas por el estándar 15289 [4]:

- fecha de emisión,
- estado del documento y
- responsables de emisión.

Por su parte, el estándar ISO/IEC/IEEE 29148 establece una lista de dimensiones adicionales que se esperan encontrar en el documento de ERS y otras dimensiones específicas para los Requerimientos de Software que conforman dicho documento. El prototipo incluirá las siguientes dimensiones en el documento de salida como cláusulas independientes [35]:

- Tabla de contenidos,

- alcances,
- referencias y
- glosario.

Los requerimientos se compilarán en una cláusula titulada “Requerimientos de Software”. Considerando los retos de realizar *minería de repositorios de* software y permitiendo también la ausencia de información completa en los documentos de entrada, algunas de las dimensiones que conforman cada requerimiento no serán extraídas automáticamente por la herramienta. En su lugar; bajo suposiciones que plantearemos con detalle en el siguiente capítulo, se les asignarán valores por defecto que los usuarios podrán alterar antes de generar las ERS.

El prototipo incluirá las siguientes dimensiones en cada requerimiento identificado en el documento de entrada [2]–[4], [35], [36]:

- Identificador,
- título,
- descripción del requerimiento y
- estado.

Puesto que es altamente recomendado que cualquier redacción y presentación respecto a estas dimensiones se haga con un lenguaje con una semántica bien establecida [2], [34]–[36], la herramienta deberá contar con mecanismos para inspeccionar y editar manualmente estas dimensiones. Los estándares y recomendaciones consultados hacen énfasis en el ejercicio de revisiones y control sobre el contenido de las ERS en estos estándares y recomendaciones [2], [14], [17], [34]–[36], se considera una revisión manual por parte de los usuarios del prototipo a desarrollar; la cuál habilite la inspección de la información candidata a ser parte de la salida.

Puesto que el estándar 29148 sugiere organizar los requerimientos en cláusulas de acuerdo a las necesidades que atienden los requerimientos según los interesados del software en cuestión [35], esta revisión deberá incluir una dimensión “cláusula de organización” que permita realizar este agrupamiento; la cuál es una dimensión opcional. Las categorías que agrupan requerimientos que sugiere este estándar, responden a necesidades específicas del sistema, sus actores, su entorno, sus propiedades o la jerarquía de sus funciones [35]; por lo que la herramienta deberá permitir especificar “sub-cláusulas de organización” a la que debe pertenecer cada requerimiento. Esta dimensión también es opcional y no tiene sentido considerarla cuando cláusula de organización no ha sido asignada.

Para terminar, esta revisión permite incluir información que no se espera extraer de los documentos de entrada. Dicha inspección debe atender los siguientes aspectos de la salida de la herramienta:

- Su redacción,
- la información extraída del documento de entrada,
- los alcances del documento,
- el glosario de términos identificados,
- la consistencia del documento,
- la homogeneidad de los sistemas métricos presentes,
- la ausencia de instrucciones de implementación de cualquier naturaleza,

- la versión del documento y
- el riesgo, volatilidad, prioridad, motivaciones, tipo y cláusula de organización individuales de cada requerimiento.

La dimensión general del documento de salida “estado del documento” y la dimensión de cada requerimiento “estado” deberán indicar respectivamente, si la información incluida en el documento generado por la herramienta ha sido aprobada por sus usuarios; así como si la información incluida en cada requerimiento ha sido revisada.

2.4 Usabilidad

La usabilidad es la suma de características de un producto, que buscan hacerlo física y mentalmente apto para sus usuarios bajo un contexto determinado [35], [37]. En el caso de los productos de software, la usabilidad se consolidó en la década de 1980 con la introducción de la PC; ofreciendo a cualquier persona la oportunidad de convertirse en un usuario de computadora desde su propio hogar. Con esta consolidación, se popularizó el área de lo que hoy conocemos como la *Interacción Persona-Computadora* (IPC) [37].

La IPC se abarca varios campos e incorpora técnicas diferentes; cómo el modelado de usuarios, análisis de tareas, cálculo de desempeño de tareas, análisis y diseño de interfaces de usuario físicas y virtuales, interactividad, tareas colaborativas, etc. La usabilidad se extiende más allá del diseño gráfico y las Interfaces Gráficas de Usuario, por ejemplo, una tendencia es la usabilidad basada en *interacción natural*; la cual consiste en intercambiar información con un software de forma similar a cómo se hace entre personas. Es decir; interactuar con el software por medio del habla, expresiones faciales, lenguaje corporal, etc. Este tipo de interacciones se conocen cómo *multimodales*; puesto que involucran varias modalidades para realizar el intercambio de información (canales visuales, de sonido, hápticos, etc) [37].

Algunos de los aspectos más importantes en la IPC; sin importar el medio o técnicas con el que se interactúe con los usuarios, son los siguientes:

- *Calidad técnica*. El software debe ser robusto y manejar errores de forma apropiada. La confianza de un usuario en el software se pierde fácilmente ante la presencia de problemas técnicos; y pueden preferir optar por soluciones técnicamente inferiores (más lentas, inseguras o ineficientes en memoria por ejemplo) con mejor usabilidad [5], [14], [37].
- *Funcionalidad*. Permite al usuario producir los resultados esperados; hace útil al software, ejercitando el propósito para el que fue producido. El propósito de un producto de software es uno de los aspectos más importantes en los procesos de software; pues no siempre se deducen o comunican de forma clara, por lo que es importante establecer acuerdos no solo con los clientes e interesados; sino con los usuarios finales del software, para que el producto final les sea realmente útil [5], [35], [37].
- *Facilidad de uso*. Ocurre cuando las funciones que ofrece el software pueden ser usadas sin que el usuario requiera herramientas, entrenamiento, configuraciones o condiciones adicionales especiales [2], [5], [14], [34]–[37].
- *Experiencia del usuario*. Son las emociones, impresiones y sensaciones que transmite el software a los usuarios. Muchas cualidades del software pueden tener impacto en la experiencia del usuario; la forma en la que sus funciones producen los resultados esperados, la velocidad con la que opera, el aspecto visual, etc [37].

La usabilidad es un factor primario para la retención de usuarios; cuando un producto no atiende positivamente los cuatro aspectos anteriores, los usuarios buscarán otras alternativas. Esto no significa que todo producto deba satisfacer

por completo los cuatro aspectos anteriores, pero debe procurarse atenderlos en el mayor grado posible y establecer mecanismos al rededor de aspectos negativos que no se puedan evitar. Además, cuando los usuarios buscan alternativas motivados por la carencia de aspectos de usabilidad, la imagen pública del producto se torna negativa, lo que afecta negativamente a quienes lo proveen y a sus colaboradores [26], [37].

2.4.1 Importancia de la usabilidad en Herramientas ISAC

Las Herramientas ISAC son aplicaciones de software que habilitan o contribuyen a la ejecución de tareas en procesos de software. Un Entorno de Desarrollo Integrado (por ejemplo Eclipse, Android Studio o Xcode), un tablero digital *kanban* o un software diagramador UML; son ejemplos de herramientas ISAC [14], [34]. Entre las décadas de 1980 y 1990, tuvieron mayor presencia un tipo específico de estas herramientas para el diseño de software y gestión de los procesos con los que se implementa. Eran dirigidas a gerentes, analistas y otros profesionales con poder de decisión en proyectos de software y fueron inspiradas por herramientas de Diseño Asistido por Computadora; imitando su aspecto y funciones [85].

Sin embargo, con la popularización de las metodologías ágiles, este tipo de herramientas ISAC perdieron popularidad; aunque lograban aumentar la productividad, los modelos de desarrollo en los que se basaban no son compatibles con las estrategias de trabajo ágil [86], [87]. Actualmente, con la popularización de la IA y los avances respecto a la capacidad de cómputo del hardware, se explora una nueva generación de herramientas ISAC; las cuales buscan aplicar diferentes metodologías y estrategias para asistir procesos de software como [87]:

- Toma de requerimientos asistida por Aprendizaje de Máquina.
- Generación semi-automática de funciones de software a partir de descripciones en lenguaje natural.
- Síntesis de consultas SQL a partir de consultas en lenguaje natural.
- Producción de arquitecturas de software basada en Redes Neuronales.
- Extracción de bloques de código como sugerencias para programadores.

La usabilidad ha sido siempre un factor importante en las herramientas ISAC; los sistemas de gestión que se habían popularizado entre 1980 y 1999, eran calificadas como *fáciles de usar e interactivas*; siendo un factor importante de su inicial popularización [86]. Por otra parte; la toma de requerimientos en un proyecto de software debe involucrar la obtención de aspectos de usabilidad, principalmente a partir de necesidades explícitas de los clientes e interesados, pero también puede involucrar procesos de diseño centrado en el usuario [1], [2], [4], [5], [14], [31], [35]. Esta última estrategia suele incluir etapas de preproducción, en las que se desarrollan prototipos de baja fidelidad que son evaluados por los usuarios; con el fin de ajustar las interfaces e interacciones a partir de su retroalimentación [2], [14], [36].

En el caso de los procesos de análisis de requerimientos, un problema común es el alcanzar acuerdos con los interesados; así como establecer un entendimiento en común de lo que se espera del software. Siguiendo metodologías ágiles, usualmente este problema se atenúa conforme se implementa y evalúa el software; pero esto suele implicar re-implementaciones, incurriendo en costos adicionales para el proyecto [14], [17], [34], [36]. Después de todo, la falta de entendimiento mutuo entre lo que se espera del software y lo que hace el producto final; es uno de los principales motivos de lo que fue la “crisis del software” [6], [7].

Con todo lo anterior; la herramienta ISAC que representa el prototipo a desarrollar en este proyecto; denominada Requex, además de seguir la tendencia de incorporar mecanismos de IA para asistir procesos de software; específicamente *Minería de Repositorios de Software* para *analizar requerimientos*, debería incorporar aspectos de

usabilidad en los mecanismos y funciones para realizar este análisis. Es indispensable que el proceso para inspeccionar el contenido que formará parte del documento de salida; descrito en la sección 2.3.4, ofrezca asistencia para seguir las recomendaciones y estándares que se han explorado; mostrando información sobre los aspectos que debe cumplir una ERS cuando el usuario lo desee, además de contemplar los cuatro aspectos principales de la usabilidad para el diseño del prototipo.

2.5 Resumen

Los conceptos, métodos y recomendaciones que hemos explorado en este capítulo se resumen a continuación:

1. En el dominio de la Inteligencia Artificial, *minería* es un término que se utiliza para referirse a aplicaciones de la Extracción de Información [38]–[41].
2. La *minería de repositorios de software* se enfoca en identificar patrones en repositorios de software; para lo que se suele utilizar conceptos y metodologías de la *minería de datos* y la *minería de textos* [32], [44]–[46].
3. Los repositorios de software son compendios de artefactos de software. Algunos de estos artefactos; como puede ser la documentación, manuales y discusiones entre los desarrolladores, están escritos predominantemente en lenguaje natural. Para estos suele aplicarse *Procesamiento de Lenguaje Natural* en combinación con técnicas de *minería de textos*. Otros artefactos están escritos en lenguajes de programación o de diseño; para los que se emplean variantes del PLN y minería de datos [32], [44], [46], [49], [57]–[59].
4. Los procesos de minería suelen describirse mediante una secuencia de operaciones que en general incluyen las siguientes etapas: obtención o recolección de los datos, preprocesamiento y limpieza de los datos, procesamiento o exploración de los datos, modelado de los datos y finalmente interpretación [61], [62].
5. El *Procesamiento de Lenguaje Natural* es un enfoque de la lingüística computacional que tiene el objetivo de extraer y sintetizar información textual redactada en algún lenguaje natural [63], [64].
6. La *Extracción de Información* consiste en identificar relaciones clave mediante la búsqueda de estructuras esperadas en los datos a explorar. Las porciones de los datos que se hayan considerado como relevantes se preparan en un formato útil de acuerdo a los objetivos que tenga el ejercicio de la Extracción de Información [42].
7. Los requerimientos de software son descripciones detallas de funciones específicas que debe cumplir un producto de software [2]–[4], [14], [35], [36], [81], [82].
8. Una Especificación de Requerimientos de Software (ERS), es un producto de trabajo del ciclo de vida software que contiene todos los requerimientos de software de un producto en particular; estableciendo un lenguaje específico con el que de describen dichos requerimientos, así como las condiciones, alcances, responsables, especificaciones técnicas, manuales y cualidades de seguridad y calidad que en su conjunto conforman y completan los requerimientos. En pocas palabras, una ERS especifica un producto de software [14], [35], [81], [82].
9. Los estándares y recomendaciones para formatos de ERS no imponen restricciones sobre la estructura de estos documentos; en su lugar, se enfocan en indicar las secciones y el contenido que se espera incluyan los documentos [4], [35].
10. La usabilidad es la suma de características de un producto que lo hacen apto para sus usuarios bajo un contexto determinado [35], [37].

11. Los cuatro aspectos más importantes de la usabilidad son: calidad técnica, funcionalidad, facilidad de uso y experiencia del usuario [2], [5], [14], [34]–[37].
12. Una *Herramienta de Ingeniería de Software Asistida por Computadora* (ISAC) es una aplicación de software que habilita o contribuye a la ejecución de tareas en procesos de software [14], [34].

3 Diseño e implementación del prototipo

En este capítulo se presenta el diseño y especificación del prototipo. Inicia presentando los requerimientos del prototipo; los cuales parten de los objetivos del proyecto y los conceptos presentados en el capítulo anterior. Posteriormente continúa con las consideraciones técnicas de la solución; los entornos de trabajo y estrategias de trabajo. Por último, se presenta la arquitectura general del prototipo.

3.1 Requerimientos

En esta sección, se describen los requerimientos a implementar. Se presentan agrupados por su tipo en el siguiente orden: Funcionales, Usabilidad e Interfaz y por último No funcionales. Los requerimientos se presentan conforme a su prioridad; los de mayor prioridad aparecen primero. La prioridad se representa con valores numéricos enteros del 0 al 10, donde 0 representa la prioridad más alta y 10 la más baja; se indica explícitamente la prioridad de cada requerimiento.

Por último, los requerimientos se expresan como cláusulas, cuyo título contiene el identificador, título y prioridad del requerimiento. En el cuerpo de dichas cláusulas se presenta la descripción, estado, volatilidad, riesgo y motivaciones que conforman cada requerimiento. La volatilidad; de forma similar a la prioridad, se representa con valores numéricos del 0 al 10, donde 0 indica que no se esperan cambios para el requerimiento y 10 que el requerimiento es altamente propenso a cambiar o incluso ser suspendido.

3.1.1 Requerimientos funcionales

1 – Datos de entrada. Prioridad: 1

Debe recibir cuando menos un archivo de texto plano que describe un producto software; describe su desarrollo o mantenimiento. Además, debe recibir el nombre del proyecto descrito en los archivos de entrada.

Motivaciones: El propósito de la herramienta es asistir el proceso de documentación de requerimientos a partir de Tesis y Reportes de Servicio Social que describan proyectos de software.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Los archivos podrían describir más de un software. Su contenido puede contener las características que dificultan la extracción de información de artefactos de software.

2 – Extracción de Información. Prioridad: 0

Debe extraer Requerimientos de Software de los documentos de entrada aplicando PLN y Extracción de Información.

Motivaciones: Los documentos de entrada contienen información respecto al desarrollo o mantenimiento de un producto de software, de los que debe ser posible recuperar requerimientos.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: El proceso de Extracción de Información puede resultar costoso en su diseño, implementación o incluso en tiempo de ejecución.

3 – Análisis asistido de requerimientos. Prioridad: 0

Debe mostrar una lista con los requerimientos extraídos de los documentos al usuario, de forma que puedan ser analizados.

Motivaciones: Los requerimientos identificados de forma automática pueden contener falsos negativos o información incompleta. En el proceso de resolver estas deficiencias en la información, los usuarios pueden analizar los requerimientos que conforman el producto de software que describen.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: La información presentada al usuario puede no ser clara; o puede no identificar las acciones que se espera lleve a cabo.

4 – Documento resultante. Prioridad: 2

Los requerimientos resultantes deben ser exportados al final de la ejecución como un archivo persistente, en forma de una Especificación de Requerimientos de Software.

Motivaciones: El resultado de analizar los documentos extraídos de los datos de entrada, se presentan en una ERS que los desarrolladores podrán consultar en cualquier momento en el futuro como referencia durante el ciclo de vida del software que describe.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: El medio de almacenamiento puede estar lleno o no disponible.

5 – Atributos de la Especificación de Requerimientos a producir. Prioridad: 0

Los atributos a incluir en los documentos de salida que produce el prototipo son: título del documento, fecha de emisión, estado del documento, versión del documento, responsables de emisión, tabla de contenidos, alcances, glosario y referencias. Estos atributos son globales del documento y no son específicos de ningún requerimiento en el documento.

El estado del documento debe ser alguno de los siguientes: {No revisado, Revisión Incompleta, Revisado}

Motivaciones: Los documentos que produce la herramienta deben satisfacer parcialmente estándares y recomendaciones para ERS. Los atributos presentados en este requerimiento, corresponden a los que se presentan en la sección 2.3.4 de este trabajo.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

6 – Atributos de los requerimientos. Prioridad: 0

Los atributos a incluir en cada requerimiento deben ser: identificador, título, prioridad, descripción, estado, volatilidad, tipo, riesgo, cláusula de organización y motivaciones. La prioridad y volatilidad se establecen en intervalos numéricos de enteros del 0 al 10. La prioridad 0 es la mayor, mientras que la volatilidad 0 representa una total estabilidad del requerimiento y 10 indica que se esperan cambios o la suspensión del requerimiento. El estado de un requerimiento puede ser uno de los valores en los siguientes conjuntos; o una combinación de un valor en cada conjunto: {Identificado, Descrito, Implementado, Verificado} y {En progreso, En revisión, Suspendido, Terminado}.

Por su parte, el tipo de los requerimientos debe ser alguno de los siguientes: Funcional, Interfaz, Calidad, No funcional, Usabilidad y finalmente Factor humano. Las cláusulas de organización pueden ser alguno de los tipos de requerimiento o bien alguno de los siguientes: Modo del sistema, Clase de usuario, Objeto, Característica, Estímulo, Respuesta y por último Jerarquía funcional. En el caso de que la cláusula de organización sea alguna de estas últimas, deberá especificarse con al menos 5 caracteres de forma libre. Esto permite especificar el Modo del sistema al que corresponde un requerimiento, el objeto específico que atiende o representa, el estímulo que emite o al que responde; etc.

Los identificadores deben ser asignados de forma automática por la herramienta, usando número enteros de forma incremental a partir de 1. El título y descripción de un requerimiento no pueden tener valores vacíos; el título debe cuando menos tener tres caracteres (permitiendo usar mnemónicos), mientras que la descripción debe poseer al menos tres palabras que en suma deben tener una longitud mínima de 7 caracteres no vacíos.

Motivaciones: Se busca que la ERS que produzca el prototipo satisfaga parcialmente estándares y recomendaciones para este tipo de documentos. Los atributos listados y sus valores son tomados a partir de estos estándares; tal como se discute en la sección 2.3.4 de este trabajo.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

7 – Estructura de la Especificación de Requerimientos a producir. Prioridad: 3

La estructura de los documentos de salida debe contener los siguientes elementos: título, estado del documento, índice de contenido, responsables de emisión, alcances, lista de requerimientos organizada por cláusulas, glosario y referencias. El título del documento debe estar presente en los encabezados de todas las páginas. La fecha de emisión, y versión del documento deben estar presentes al en todos los pie de página; así como el número de página y páginas totales.

Motivaciones: Aunado al requerimiento 3.1.1 – 6, este requerimiento establece una estructura que se apega a las recomendaciones y estándares presentados en la sección 2.3.4. El incluir la fecha de emisión, versión del documento, número de página y páginas totales en todas las páginas permite identificar y organizar copias impresas o distribuidas por páginas que pudieran generarse a partir de la salida original de la herramienta (que es un único documento de acuerdo al requerimiento 3.1.1 – 4); esta es la misma razón por la que se requiere el título en todas las páginas.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

8 – Registro de requerimientos. Prioridad: 1

La lista de requerimientos debe permitir agregar requerimientos no extraídos de los documentos de entrada.

Motivaciones: Ya sea por deficiencias en el proceso de extracción de requerimientos o por su ausencia en los documentos de entrada, es posible que no todos los requerimientos de un proyecto sean encontrados de forma automática.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

9 – Requerimientos descartados. Prioridad: 1

La lista de requerimientos debe permitir quitar registros.

Motivaciones: Falsos positivos en el proceso de extracción, errores o cambios de decisiones al utilizar la lista de requerimientos, pueden resultar en la inclusión de registros no deseados.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

10 – Registros de actividad. Prioridad: 8

Debe producir registros persistentes de texto que indiquen información relevante de su ejecución.

Motivaciones: La información en estos registros debe poder ser utilizada con fines de depuración y futuras mejoras; así como auditar el desempeño de la extracción de requerimientos.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: El medio de almacenamiento puede estar lleno o no disponible.

11 – Asistencia para el análisis de requerimientos. Prioridad: 6

Debe contar con una función de ayuda que permita consultar la información para ERS tomada de estándares y recomendaciones para este tipo de documentos.

Motivaciones: La sección 2.3 resume y presenta en torno a las ERS a partir de las cuales se pueden reflexionar los requerimientos que se estén analizando con la herramienta; de forma similar a cómo se han empleado en esta sección para diseñar el prototipo.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: La ayuda asociada a cada función de la aplicación puede presentar defectos de usabilidad; decrementando su utilidad.

12 – Papelera de requerimientos. Prioridad: 5

Los registros eliminados de la lista de requerimientos son enviados a una papelera de la que pueden ser recuperados.

Motivaciones: El uso de la herramienta en el diseño de un software nuevo, o de cuyo desarrollo se posee poca información; puede derivar en modificaciones que eliminan y retoman requerimientos conforme se consolida el diseño del software.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

13 – Formato de salida aumentable. Prioridad: 4

Los formatos de archivo de la ERS a producir, deben poder ser agregados por medio de módulos opcionales adicionales.

Motivaciones: El formato de archivo por defecto de la herramienta puede ser inadecuado para el proyecto en el que se esté usando, por lo que se permite extender esta función para generar documentos de diferente formato de archivo.

Estado: Implementado. Suspendido. Volatilidad: 0. Riesgo: Ninguno.

3.1.2 Requerimientos de Usabilidad e Interfaz

14 – Entrada de datos necesaria al arranque. Prioridad: 1

Debe poderse proveer los documentos de entrada al ejecutar el prototipo. La entrada de documentos interactiva puede presentarse como una característica opcional.

Motivaciones: Una entrada de datos no interactiva ofrece una implementación centrada en las características funcionales de la herramienta sin necesidad de una IU compleja.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

15 – Edición de los atributos generales de la salida. Prioridad 0

Durante la etapa de análisis de requerimientos, debe permitirse modificar los atributos de la ERS a producir; y dicha información debe poderse contrastar con el estado general de los requerimientos que se analizan.

Motivaciones: De acuerdo con los estándares y recomendaciones revisados en la sección 2.3.4, debe haber cohesión entre los atributos del documento y la información contenida en cada requerimiento que describe dicho documento.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: La organización de información en una interfaz suele ser una tarea no trivial, y puede requerir de varias etapas de prueba.

16 – Inspección rápida de requerimientos. Prioridad: 1

La lista de requerimientos a incluir en la ERS de salida, debe presentar su contenido de forma que sea posible identificar rápidamente aquellos que requieran atención.

Motivaciones: La *lista de requerimientos* presentada en la sección 3.1.1 – 3, puede percibirse como irrelevante si únicamente consiste de identificadores o títulos. Es importante presentar la información de los requerimientos que contiene de forma resumida, pero al mismo tiempo revelando la integridad de la información de cada requerimiento en ella.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: La organización de información en una interfaz suele ser una tarea no trivial, y puede requerir de varias etapas de prueba.

17 – Edición de los atributos específicos de cada requerimiento. Prioridad: 0

Los requerimientos deben poder ser inspeccionados y modificados de forma individual, evitando interferencia de la información de otros requerimientos o de los atributos generales del documento ERS a producir.

Motivaciones: De manera similar a como se presenta en el requerimiento 3.1.2 – 16, es importante ofrecer un espacio ordenado en el que los usuarios puedan inspeccionar la información de cada requerimiento, evitando distracciones que puede causar el exhibir información no necesaria en los requerimientos de forma individual.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: La organización de información en una interfaz suele ser una tarea no trivial, y puede requerir de varias etapas de prueba.

18 – Guías en las restricciones. Prioridad: 2

Las restricciones en los campos que conforman los atributos del documento de ERS a producir y de los requerimientos, deben ser presentadas de forma explícita al usuario; debe indicarse claramente cuando dichas restricciones no sean satisfechas.

Motivaciones: Guiar las acciones del usuario sin limitar el acceso a las funciones de la herramienta, es un aspecto básico usabilidad.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Definir mensajes de error claros y oportunos puede requerir varias etapas de evaluación de usabilidad.

19 – Advertencia de contenido incompleto. Prioridad: 2

Debe verificar que todos los atributos; tanto del mismo documento ERS a producir como de los requerimientos individuales que contendrá, han sido asignados con valores válidos de acuerdo a los requerimientos 5 y 6. En caso de que no estén completos, la herramienta requerirá una confirmación para poder generar el documento, y el atributo *estado del documento* no podrá tener el valor *Revisado*.

Motivaciones: De acuerdo con los estándares y recomendaciones para ERS revisados en la sección 2.3, es importante que la información contenida en este tipo de documentos sea completa puesto que especifica un producto de software.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

20 – Selección de valores restringidos. Prioridad: 4

Los valores limitados a valores específicos descritos en la sección 3.1.1 – 6, deben ser ofrecidos al usuario como opciones; evitando respuestas abiertas.

Motivaciones: Las respuestas abiertas son propensas a errores cuando se espera que tomen valores específicos. Además, esto brinda una guía al usuario y puede facilitar el uso de la herramienta.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

21 – Formato de salida seleccionable. Prioridad: 6

El formato de archivo de la ERS a producir, debe poder ser elegido por el usuario.

Motivaciones: El requerimiento 14 (sección 3.1.1 – 14) especifica la capacidad de la herramienta para permitir la incorporación de formatos de salida adicionales que se adapten a los proyectos con los que se use. Este requerimiento

exige que dichas extensiones puedan ser reconocidas en tiempo de ejecución de forma que los usuarios puedan seleccionar explícitamente el formato de salida deseado.

Estado: Implementado. Suspendido. Volatilidad: 0. Riesgo: Ninguno.

22 – Asistencia interactiva. Prioridad: 4

La ayuda descrita en la sección 3.1.1 – 11, debe centrarse en la actividad que el usuario esté efectuando en el momento en el que la invoque: ingresando documentos de entrada (vea las sección 3.1.2 – 14), revisando la lista de requerimientos (sección 3.1.1 – 3), asignando valores a los atributos generales del documento de salida (sección 3.1.2 – 15), editando los atributos de un requerimiento específico (3.1.2 – 17) o al seleccionar el formato de salida (3.1.2 – 21). También deberá existir ayuda general que permita conocer las funciones de la herramienta; así como consultar el resumen de los estándares y recomendaciones para generar las ERS que se presentan en la sección 2.3.4.

Motivaciones: Este requerimiento busca contribuir a las guías incorporadas en la herramienta, además de asistir en el proceso de análisis de requerimientos al hacer accesible la información respecto a las recomendaciones y estándares en torno a las ERS.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: La respuesta a las actividades del usuario, así como el contenido de los mensajes de ayuda, pueden requerir de varias etapas de diseño.

23 - Interfaz Gráfica de Usuario. Prioridad: 5

Debe contar con una Interfaz Gráfica de Usuario que permita explotar todas las características de la herramienta.

Motivaciones: Uno de los propósitos de esta herramienta es contar con una buena usabilidad. Una interfaz gráfica permite una interacción que ofrezca visualizar la ERS que se está generando más natural que una interfaz basada en comandos.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

3.1.3 Requerimientos No Funcionales

24 – Puntajes de la Extracción de Información. Prioridad: 5

Debe reportar información respecto al proceso de Extracción de Información; en particular, dicha información debe permitir obtener los puntajes de Precisión y Exhaustividad y evaluar el proceso de Extracción de Requerimientos.

Motivaciones: El proyecto tiene dos aspectos principales: implementar mecanismos basados en MRS para extraer requerimientos y por otra parte mecanismos que asistan el análisis de requerimientos y generar ERS a partir de dicho análisis. Los aspectos de usabilidad abarcados principalmente por los requerimientos en la sección 3.1.2 se deben ser complementados con mecanismos para evaluar la calidad de la extracción de los requerimientos.

Estado: Verificado. Terminado. Volatilidad: 0. Riesgo: Ninguno.

25 – Plataformas de ejecución. Prioridad: 10

Debe poderse ejecutar en: entornos de Python, sistemas Windows y sistemas Linux.

Motivaciones: En la sección 3.2.3 se expresa que los proyectos para los que se construye esta herramienta, suelen ser realizados con plataformas Windows y Linux; por lo que es un aspecto que debe satisfacer el prototipo.

Estado: Implementado. Suspendido. Volatilidad: 2. Riesgo: Las dependencias del proyecto requieren archivos de configuración que han probado ser difíciles de empaquetar con la aplicación resultante.

3.2 Consideraciones técnicas de la solución

Comencemos por explorar las herramientas y plataformas de desarrollo del prototipo. Las tareas de PLN serán implementadas a través de bibliotecas terceras para este propósito. Una de las consideraciones de diseño más importante es permitir modificar y aumentar funcionalidades, de forma que pueda ser mejorado económicamente.

3.2.1 Entorno de desarrollo

3.2.1.1 Lenguaje de programación

Dada la alta disponibilidad de herramientas para implementar PLN con Python [58], se elige este lenguaje para realizar la implementación del prototipo. Otro lenguaje con un número significativo de herramientas para PLN es Java; sin embargo, se prefiere Python puesto que cuenta una biblioteca que bajo la configuración por omisión, ofrece mejores análisis textuales que otras bibliotecas para PLN [58].

3.2.1.2 Herramientas de cómputo para PLN

Existen varias bibliotecas para realizar PLN con Python. Algunas de las más populares son: CoreNLP, SyntaxNet y NLTK [58]. Estas herramientas ofrecen funciones que permiten analizar un texto en uno o varios lenguajes naturales mediante procedimientos de PLN; ofreciendo resultados cómo: tokenización y lematización, etiquetamiento de partes de la oración.

De acuerdo con Omran y Treude [58], una biblioteca que ofrece mejores resultados para procesar artefactos de software en su configuración por omisión, es *spaCy*, se elige esta herramienta como base para implementar las tareas de PLN que requiere el proyecto. Otro factor importante para elegir spaCy es que cuenta con amplio soporte en textos en español, incluyendo modelos pre-entrenados con corpus conformados por entradas de Wikipedia y publicaciones periodísticas [88]. SpaCy identifica y etiqueta información morfo-lingüística; como persona, tiempo, género, tipo de constituyente sintáctico (verbo, sustantivo, adjetivo, etc); entre otras.

3.2.1.3 Manejo de versiones

Se elije Git 2.20.1 como herramienta de manejo de versiones. La estrategia del versionamiento del proyecto es la siguiente:

- Existe una rama principal llamada “main” (siguiendo convenciones para trabajar con Git y la tendencia por reemplazar el término *master* para evitar términos que pueden ser ofensivos [89]–[97]). Esta rama principal mantiene un estado del proyecto que representa la última versión de lanzamiento del prototipo.
- Una rama secundaria llamada “testing” se utiliza para acumular todos los cambios realizados por el desarrollo, los cuales deben satisfacer todas las pruebas unitarias y de integración que se definan en el repositorio.
- Los cambios individuales deben ser publicados en ramas específicas para cada colaborador en el proyecto y cada función a implementar, modificar o corregir. La nomenclatura para las ramas de desarrollo es la siguiente:
    - El nombre de cualquier rama del proyecto, debe iniciar con el nombre de usuario del miembro del equipo propietario de dicha rama. El nombre de usuario debe ser seguido por “/”, y este carácter a su vez seguido por una muy breve descripción; de no más de cinco palabras cortas, de las funciones, cambios o tareas que se implementan en dicha rama.
    - Los nombres de las ramas deben seguir la convención *snake_case*.

- Las ramas utilizadas para inicializar tareas de programación o definir las pruebas que debe satisfacer una funcionalidad a implementar, deben ser manejadas por colaboradores del proyecto con facultades para supervisar la implementación. En lugar de llevar el nombre del miembro del equipo que crea una de estas ramas; el nombre de estas ramas debe iniciar con “preprod/*”.
- En el caso de corrección de errores y modificaciones menores, se recomienda usar el nombre genérico “*/hotfix”.

La dinámica para generar versiones se describe a continuación:

- Ningún desarrollador puede publicar en la rama principal. La rama principal; y con ella cada versión de lanzamiento, cambia únicamente al recibir los cambios de la rama secundaria, y únicamente puede recibir dichos cambios cuando el estado de la rama secundaria a integrar satisface todas las pruebas definidas en el repositorio.
- De forma similar, ningún desarrollador puede publicar directamente en la rama secundaria. Esta rama recibe cambios de las ramas de los desarrolladores, y únicamente puede recibir dichos cambios cuando el estado de la rama a integrar satisface sus pruebas.
- Cada vez que se integran los cambios de la rama secundaria en la principal, esta debe ser etiquetada con el número de versión que le corresponda de acuerdo a las reglas para obtener los nombres de versiones.

Las reglas para nombrar las versiones de la herramienta son las siguientes:

- Los nombres de versión consisten en tres números enteros no negativos, separando cada par de números por un punto. Representando cada número con ‘#’, el formato es: #.#.#
- La versión inicial; que representa el estado del proyecto al momento de ser inicializado, es: 0.0.0
- El número en el extremo izquierdo de la versión, representa una versión de lanzamiento. El número en el extremo derecho de la versión, representa el número de cambios individuales desde la última versión.
  - Por cada cambio que se publique por parte de un miembro del equipo, el número en el extremo derecho es incrementado en una unidad.
  - Cada vez que se integran cambios a la rama secundaria, el número en el centro de la versión es incrementado en una unidad; y el número en el extremo derecho reinicia en cero.
  - Cada vez que se integra la rama secundaria en la primaria, se incrementa el número al extremo izquierdo; y se reinician en cero los otros dos números en la versión.

El proyecto se hospeda en un repositorio de gitlab, donde se cuenta con un Sistema de Seguimiento de Incidentes (SSI), el cuál se utiliza en este proyecto para registrar sugerencias de cambios, mejoras, errores; y tareas de diseño o programación. Se debe indicar el número de incidente que atienden usando el identificador numérico que Gitlab genera automáticamente.

3.2.1.4 Metodología de desarrollo

Para desarrollar el proyecto, se utiliza una estrategia de trabajo inspirada en *Programación Extrema* (*Extreme Programming – XP*) [14], [17]. La metodología del proyecto parte de los siguiente puntos:

a) Si bien no se cuenta con un cliente específico, se colabora con los miembros del Grupo de Espacios y Sistemas Interactivos para la Educación (ESIE), del Instituto de Ciencias Aplicadas y Tecnología (ICAT) de la UNAM. El

diseño de la extracción de requerimientos y documentos modelo para el diseño, se toman a partir de documentos de titulación dirigidos por los integrantes del ESIE.

b) El proyecto se basa en tres requerimientos de alto nivel: extraer requerimientos a partir de Tesis y Reportes de Servicio Social, asistir el análisis de dichos requerimientos y generar una ERS tras el análisis.

c) Ocurrirán dos lanzamientos: el primero a más tardar el 16 abril del 2021, y el segundo a más tardar el 15 de junio del 2021. Para el primer lanzamiento, el prototipo debe representar un Producto Mínimo Viable, mientras que para el segundo debe ofrecer funcionalidades completas que puedan utilizarse en los proyectos del ESIE.

d) El proyecto maneja tres tipos de pruebas: unitarias, de integración y de usabilidad. Las pruebas unitarias y de integración se deben ejecutar automáticamente durante el desarrollo de cada iteración. Las pruebas de usabilidad verifican que la salida general del prototipo, contienen la información que se espera de acuerdo a la sección 2.3.4 y serán realizadas tras cada lanzamiento.

3.2.1.5 Estructura del repositorio

La estructura del proyecto tiene como base un entorno virtual de Python3 . La estructura de directorios que conforman el entorno del proyecto se describe a continuación:

<Directorio contenedor del proyecto>

- ↳ <Archivos y directorios del entorno virtual de Python>
- ↳ **setup.py** – Archivo de configuración del proyecto. Se emplea para indicar el directorio donde está contenida la implementación de *Requex* (src), así como configurar la compilación y distribución del prototipo.
- ↳ **make.py** – Archivo de configuración para la compilación y distribución del producto de software del proyecto.
- ↳ **setup.cfg** – Archivo de configuración para la ejecución de pruebas y análisis de código automáticos.
- ↳ **src/** – Contiene el código fuente del prototipo, organizado en directorios de acuerdo a los paquetes (módulos de Python) que lo componen. Estos paquetes se presentan en la sección 3.3.2.
- ↳ **test/** – Contiene las rutinas para ejecutar de forma automática las pruebas del proyecto.
  - ↳ **unitarias/** - Contiene scripts con pruebas que validan el correcto comportamiento de funciones específicas de los distintos componentes de *Requex*.
  - ↳ **integracion/** - Contiene scripts con pruebas que validan el proceso de extracción de requerimientos. No verifican la integridad o calidad de la información extraída, se limitan a verificar que el proceso se ejecuta tal y como se ha diseñado.
- ↳ **dist/** – Contiene paquetes ejecutables y distribuibles generados para la versión correspondiente a la rama de desarrollo en la que se observe el repositorio. Los entornos de ejecución para los que se destinan estos paquetes se describen en la sección 3.2.3.
- ↳ **res/** – Contiene archivos que se pretenden usar como entrada del documento, para entrenar los modelos que utiliza para realizar PLN y los modelos resultantes de tal entrenamiento.
  - ↳ **raw/** – Contiene documentos de Tesis y Reportes de Trabajo Social que describen proyectos de software. Estos documentos están en su formato original y no necesariamente se pueden usar directamente con la herramienta.

- ↳ **text/** – Contiene archivos de texto que pueden usarse como entrada para entrenar los modelos del lenguaje del prototipo.
- ↳ **models/ –** Contiene modelos estadísticos de características del lenguaje, generados durante un entrenamiento anterior; y que son necesarios para realizar el proceso de extracción de requerimientos.

↳ **util/** - Contiene scripts que explotan herramientas terceras para convertir documentos en archivos de texto plano; de forma que puedan usarse como entrada de la herramienta.

↳ **doc/** - Contiene documentación del proyecto; principalmente diagramas de clase y de secuencias. Los diagramas más relevantes se presentan en la sección 3.3.2.

↳ **README.md** – Ofrece una breve descripción del proyecto que puede ser consultada directamente en el repositorio en línea en el que se encuentre hospedado.

↳ **COPYING.md** – Contiene la licencia bajo la que se distribuye el proyecto.

3.2.1.6 Ambiente de desarrollo

El ambiente de trabajo es `Python 3.7.3` sobre `GNU/Debian 10` con kernel `Linux 4.19.0-14-amd64` (o superior). El proyecto utiliza las siguientes dependencias; las cuáles son administradas utilizando el manejador de dependencias de `Python pip`, en su versión `18.1` y bajo el entorno virtual del proyecto.

- `SpaCy 3.0.1` – Herramienta para Procesamiento de Lenguaje Natural con Python [98].
- `gensim 3.8.3` – Herramienta para modelado estadístico y vectorial de texto; además de análisis de tópicos [99].
- `flake8 3.8.4` – Se trata de una herramienta de calidad de código, verifica que código en Python se apegue a las normas del lenguaje, además de ofrecer evaluar su complejidad ciclomática [100].
- `pytest 6.2.2` – Entorno de pruebas para Python que permite implementar validaciones para el comportamiento del software en desarrollo. Extiende la funcionalidad del módulo de pruebas por omisión de Python; y además acepta extensiones adicionales [101].
- `pyinstaller 4.2` – Compilador que congela un proyecto de Python y lo empaqueta en un ejecutable para la misma plataforma en la que es generado. El paquete generado no requiere dependencias adicionales para su ejecución [102].
- `fpdf2 2.3.0` – Herramienta para crear archivos PDF con Python.
- `bs4 0.0.1` – Biblioteca para el análisis de texto en formato XML. Se utiliza para determinar si un candidato a requerimiento es en mayor proporción código XML o HTML; en cuyo caso el candidato se deshecha.
- `tkinter 8.6.9` – Plataforma portable para construir entornos gráficos de usuario en un sistema de escritorio.
- `distutils 3.7.3` – Paquete de soporte para el manejo de módulos de Python.

3.2.2 Pruebas

En la sección 3.2.1.4 se indicó que el proyecto utiliza tres tipos de pruebas: unitarias, de integración y de usabilidad. Las pruebas unitarias y de integración se ejecutan dentro el entorno de desarrollo; por su parte las pruebas de usabilidad buscan verificar que los documentos de salida que produce el prototipo son ERS que contienen adecuadamente todos

los atributos que se describen en la sección 2.3.4; y verificar que el proceso de uso del prototipo asiste a identificar dichos atributos de forma clara y organizada.

Las pruebas unitarias y de integración son definidas al inicio de cada ciclo de desarrollo [14], [17]. En particular, debe haber al menos una prueba unitaria por cada funcionalidad a implementar, y al menos una prueba de integración que verifique el comportamiento (pero no necesariamente los resultados) del proceso de extracción de requerimientos. Al final de cada ciclo, debe contarse con funciones que satisfacen dichas pruebas.

Las pruebas de usabilidad requieren de la existencia de una versión del prototipo que pueda ser usada por los usuarios; cuando menos para inspeccionar los requerimientos que extraiga de documentos de entrada. Por lo anterior, estas pruebas se ejecutan al final de los ciclos de desarrollo, suponiendo que las funciones implementadas satisfacen las pruebas unitarias y de integración.

Los principales aspectos las evaluaciones de usabilidad se enfocan en descubrir la utilidad que perciben los usuarios para analizar los requerimientos de los proyectos que describen los documentos de entrada. Además de realizar una sesión en la que los usuarios deberán interactuar con el prototipo y poder posteriormente responder una entrevista [37].

En el siguiente capítulo se describen las pruebas de usabilidad con detalle.

3.2.3 Entorno de ejecución

Los proyectos para los que se crea la herramienta, suelen emplear plataformas basadas en Windows y Linux; por lo que se consideran estas dos plataformas como las plataforma objetivo para la distribución del prototipo. Partiendo de la denominación del prototipo *Requex*; las versiones de lanzamiento del prototipo se nombran de la siguiente manera:

- requex-#.#.#_win – Compilaciones generadas para sistemas Microsoft Windows 10.
- requex-#.#.#_linux – Compilaciones generadas para sistemas GNU/Linux.
- requex-#.#.#_py – Paquetes distribuibles y portables que pueden ser utilizados en entornos de Python que satisfagan la configuración del entorno de desarrollo.

En los tres casos, #.#.# debe ser sustituido por el número de versión al que corresponde la compilación. En caso de que una compilación no sea tomada de la rama principal, debe agregarse el nombre de la rama origen como sufijo al nombre del paquete.

El proyecto se encuentra disponible bajo licencia GNU GPL 3.0 en el siguiente repositorio hospedado en Gitlab: https://gitlab.com/nachintoch/requex

3.3 Diseño y arquitectura del prototipo

En esta sección, se presentan los razonamientos y mecanismos con los que se ha diseñado el prototipo, para terminar con diagramas de clases, módulos y secuencias que describen la herramienta; este diseño responde a los requerimientos planteados en la sección anterior. En particular se presta mayor atención al diseño del proceso de extracción de requerimientos. La detección de estructuras lingüísticas en los documentos de entrada se basa en reglas que buscan identificar un patrón gramatical específico.

Dicho patrón gramatical se ha detectado manualmente al explorar ejemplares de documentos de entrada para el prototipo; con el que se ha establecido que la detección de requerimientos puede realizarse con el uso de palabras clave o la presencia de perífrasis verbal.

3.3.1 Secuencia de operaciones para la extracción de requerimientos

Retomemos las cinco operaciones de alto nivel que conforman un procesos de minería descritas en la sección 2.2.1: recolección de datos, preprocesamiento y limpieza de datos, exploración de los datos, modelado de los datos y finalmente interpretación. La primer operación; recolección de datos, es realizada manualmente. En particular, los datos de entrada del prototipo se hacen disponibles a través del directorio *res/text*; tal como se describe en la sección 3.2.1.5. De esta manera, el directorio *res/text* funciona como Lago de Datos en este proyecto, atendiendo también el requerimiento 3.1.1 – 1.

3.3.1.1 Preprocesamiento – Limpieza de datos textuales en lenguaje natural

Esta operación utiliza PLN por medio de spaCy (sección 3.2.1.6). Consiste de las siguientes tareas: tokenización del texto, eliminación de palabras vacías, lematización y extracción de raíces y etiquetado de partes de la oración. Se utiliza el modelo para el español `es_core_news_md` incluido con spaCy; entrenado con artículos periodísticos y de Wikipedia [67], [88], [103], [104]. Se elige por ofrecer un análisis de alta calidad; además de permitir realizar análisis de corta duración (en comparación con los modelos de mayor precisión).

Se buscó enriquecer este modelo general con un corpus formado por Tesis y Reportes de Servicio Social dirigidas por integrantes del ESIE. Sin embargo; debido a problemas de comunicación derivados del trabajo a distancia a raíz de la pandemia por COVID-19, el número de documentos modelos, que contaran con las características deseadas para los documentos de entrada de la herramienta, fue de apenas 8 documentos (que contienen un total de 38,033 palabras).

Por otra parte, para crear un corpus que pueda impactar o equiparase al modelo del lenguaje base utilizado; se esperaría que el corpus de entrenamiento contara con cientos de miles de palabras (o más, por tratarse de un corpus automatizado modelado estadísticamente), contenidas en Tesis y Reportes de Servicio Social que describen proyectos académicos de software [63], [66], [68], [69], [88], [105].

Bajo los alcances de este proyecto; considerando que se busca explorar la viabilidad de crear una prueba de concepto para extraer requerimientos de forma automática, se optó por utilizar el modelo `es_core_news_md` enriquecido con los 8 documentos de entrenamiento, los resultados obtenidos con este modelo del lenguaje permitirán reconocer la viabilidad de realizar la extracción de requerimientos y ofrecer un panorama general de los resultados que pueden esperar de la solución que se propone.

El resultado de analizar un documento con este modelo es un objeto del tipo spacy.tokens.Doc; el cual representa el documento analizado con etiquetas que contienen la información morfolingüística detectada en el documento. Este objeto puede interpretarse como una lista de los tokens que conforman el documento original, los cuáles pueden ser explorados para conocer: el tipo de caracteres que lo conforman (alfanuméricos, espacios, números, signos de puntuación, si es una URL o dirección de correo electrónico), el lema del que proviene el token en el lenguaje, su rol en la oración; así como el tipo de entidad nombrada identificada de ser el caso, entre otras propiedades [67], [106].

A continuación se presentan las tareas de PLN que conforman este preprocesamiento.

3.3.1.1.1 Palabras vacías para el análisis

Considerando la sección 2.3 "*Estándares para ERS*" y los patrones verbales de la sección 2.2.2.9.1, se identifican los siguientes tipos de sintagma como los más relevantes en la descripción de un requerimiento:

- Verbos. De acuerdo con el estándar internacional 29148:2018 (sección 2.3.1), los requerimientos de software presentan un problema u objetivo a satisfacer; los cuales pueden emplear verbos para expresar las acciones o condiciones en torno a dicho problema u objetivo.

- Sustantivos. Tanto la sección 2.3.1 como 2.3.2, indican que los requerimientos de software involucran objetos, fechas, lugares, personas y otras entidades que se expresan directamente como sustantivos.
- Pronombres. Aunado a lo anterior, es posible que en lugar de repetir sustantivos, los requerimientos hagan referencia a ellos utilizando pronombres.
- Adjetivos. En el estándar 29148:2018 también se menciona que los requerimiento deben incluir todas las capacidades, restricciones y factores que deban satisfacer. Los adjetivos se distinguen por calificar o caracterizar sustantivos; por lo que resultan indispensables para identificar estos elementos en un requerimiento.

  Otro motivo por el que los adjetivos resultan útiles, es debido a que también pueden hacer referencia a objetos en la oración de los requerimientos; por ejemplo, empleando los adjetivos *dicho* o *tal*.
- Adverbios. Los adverbios expresan circunstancias, por lo que pueden referir información importante en los atributos de los requerimientos.

Los números también resultan importantes, pues pueden cuantificar aspectos de los requerimientos, por lo que no se consideran palabras vacías.

A partir de lo anterior, consideramos que las preposiciones, conjunciones y otras palabras o caracteres cuya categoría gramatical no corresponda con ninguna de las anteriores (como auxiliares o signos de puntuación), pueden considerarse palabras vacías en el dominio de los requerimientos de software.

3.3.1.1.2 Identificación de Colocaciones mediante N-gramas

Las colocaciones son frases o grupos de palabras contiguos cuyo sentido suele extenderse más allá de lo que pueden sugerir las palabras que las componen. Un grupo de palabras contiguo con una alta frecuencia es candidato a ser una colocación [66]. *Inteligencia Artificial* y *Computadora Personal* son ejemplos de colocaciones. Estas secuencias de palabras pueden arrojar información útil para identificar atributos de requerimientos, la repetición de segmentos de palabras puede usarse para describir aspectos importantes del software.

Podemos identificar colocaciones utilizando n-gramas de palabras con gensim a partir de los documentos etiquetados [67]. Estas colocaciones ofrecen información útil para el proceso de extracción de términos para el glosario del documento de salida; especificado en 3.1.1 – 5. Los n-gramas se calculan utilizando copias de los documentos de entrada que excluyen palabras vacías y utilizan los lemas de las palabras que los conforman.

El texto original puede interpretarse como una secuencia de uni-gramas (palabras individuales). Estos uni-gramas se utilizan para tratar de generar colocaciones de longitud dos (bi-gramas) con gensim; cuyo resultado depende de la frecuencia de pares de palabras no vacías contiguas. De forma sucesiva, con los bi-gramas resultantes pueden generarse tri-gramas; el prototipo tratará de generar colocaciones de la mayor longitud posible; hasta que alcanzar un estado en el que no se detecten colocaciones nuevas.

3.3.1.1.3 Modelo resultante del preprocesamiento

El resultado de estas tareas de PLN deben ser devueltas bajo algún modelo que permita a la siguiente etapa del proceso de extracción utilizar la información identificada en los documentos de entrada. Puesto que los atributos de los requerimientos deben ser legibles por humanos, es importante conservar los textos originales; con sus palabras en forma original y no lematizada e incluyendo sus palabras vacías.

Además de los resultados del preprocesamiento (el texto etiquetado y las colocaciones), se generan también modelos vectoriales de palabras. Estos modelos permiten explorar información en los textos utilizando herramientas algebraicas a partir de vectores que representan los textos a analizar. Los modelos vectoriales de palabras nos permiten determinar

el grado de similitud entre términos del glosario; de manera que el glosario pueda agrupar los términos que se determinen sinónimos [66], [67].

Gensim permite modelar la similitud de palabras mediante un modelo basado en Word2Vec [67], [107]. Word2Vec es un modelo vectorial de texto que utiliza redes neuronales para determinar la similitud de las palabras en el texto; con el entrenamiento adecuado, la similitud puede incluir información semántica en las oraciones que conforman el texto [108].

Por otra parte, también podemos utilizar modelos vectoriales para determinar el grado de relevancia de los términos del glosario, buscando conservar únicamente los términos más relevantes. El índice TF-IDF (Frecuencia de Términos – Frecuencia Inversa de Documentos) estima la relevancia de términos por documento en un corpus. Combina la frecuencia del término en algún documento particular del corpus, con su frecuencia inversa; la cuál se obtiene como el cociente del número de documentos en el corpus entre el número de documentos en que aparece el término [67].

Gensim también ofrece una implementación del modelo TF-IDF que podemos aprovechar para modelar la relevancia de términos. Los dos modelos de palabras que se se han descrito, son construidos filtrando palabras vacías [109].

3.3.1.2 Exploración de los datos – Extracción de Requerimientos

La exploración de los datos se basa en reglas que buscan detectar estructuras morfosintácticas en el texto, que deben contener determinadas palabras clave que sugieren la presencia de un requerimiento. Tras inspeccionar manualmente 8 ejemplares de documentos de entrada para el prototipo; se ha identificado que los requerimientos que describen, poseen una estructura gramatical común.

Se representa esta estructura con un lenguaje formal en el que los símbolos no terminales representan las categorías sintagmáticas del español relevantes para conformar el patrón identificado en los requerimientos. Estos símbolos no terminales y sus reglas de producción se describen a continuación:

- SUST – símbolo no terminal que puede ser sustituido por cualquier sustantivo del español.
- VERB – símbolo no terminal que puede ser sustituido por un verbo del español.
- ADJ – símbolo no terminal que se sustituye por adjetivos del español.
- ADV – símbolo no terminal que se sustituye por adverbios del español.
- P – símbolo no terminal que representa el fin de una oración en español.

Utilizando un meta-símbolo “+” como separador para facilitar la lectura de la cadena de símbolos no terminales, el patrón identificado en los textos de entrada para el prototipo con los que se describen requerimientos es: $(SUST^{*}+VERB^{*}+ADJ^{*}+ADV^{*})^{+}+P$

Este patrón resulta demasiado general para el español; por ejemplo (omitiendo palabras vacías), frases cómo “*Juan fue feliz al parque*” $(SUST^{1}+VERB^{1}+ADJ^{1}+ADV^{0})+(SUST^{1}+VERB^{0}+ADJ^{0}+ADV^{0})+P$ o “*corre alegremente*” $(SUST^{0}+VERB^{1}+ADJ^{0}+ADV^{1})+P$ satisfacen el patrón presentado.

Las palabras clave que se buscan indican actores y acciones; elementos que se espera encontrar en la descripción de una función de un producto de software. Los actores son sustantivos como *sistema*, *usuario* o *aplicación*. Las acciones pueden ser verbos comúnmente empleados en la jerga informática como: *regresar*, *eliminar* o *ejecutar*. Además de estos verbos, podemos aprovechar la perífrasis verbal como se presenta en la sección 2.2.2.9.3; pues nos permite identificar verbos que transmiten acciones acompañados de otra palabra.

Otros patrones verbales descritos en la sección 2.2.2.9 también nos ayudan a interpretar la información en un texto que satisface la estructura gramatical identificada. Las formas verbales no personales (secciones 2.2.2.9.1 y 2.2.2.9.2) nos permiten agrupar verbos como una sola entidad, por lo que no serán separados en el procesamiento del texto para formar el glosario del documento de salida.

Con todo lo anterior, los requerimientos se extraen de la siguiente forma:

1. Se buscan segmentos de texto que tengan la estructura gramatical $\left(SUST^{*}+VERB^{*}+ADJ^{*}+ADV^{*}\right)^{+}+P$
2. Se comprueba que los segmentos de texto que poseen la estructura anterior, satisfagan alguna de las siguientes condiciones:
    1. Palabras clave.
        - Contiene alguno de los siguientes sustantivos: sistema, aplicación, software, usuario, participante, paquete, mensaje, función, módulo, interfaz, interacción, accesibilidad, dato, información, botón, menú, componente, rutina, método, memoria, tarea, estructura, conjunto, arreglo, lista, alta, baja, disco, almacenamiento, dispositivo, dimensión, parámetro, atributo, entrada, salida, requerimiento, formato, plataforma o pantalla.
        - Contiene alguno de los siguientes verbos: deber, contar, poder, aplicar, hacer, mostrar, imprimir, desplegar, regresar, recibir, devolver, salir, incluir, insertar, agregar, eliminar, borrar, suprimir, quitar, producir, consumir, proveer, modificar, alterar, cambiar, limitar, reportar, ejecutar, registrar, cambiar, almacenar, parametrizar o manipular.
    2. Perífrasis verbal.
        - Se toma el primer verbo y se verifica que satisfaga alguna de las siguientes condiciones:
            - El verbo es “ir” conjugado en presente (cualquier persona) y es seguido por “a” en el texto original,
            - El verbo es “tener” conjugado en cualquier tiempo y persona y es seguido por “que” en el texto original,
            - El verbo es “haber” conjugado en cualquier tiempo y en tercera persona; y es seguido por “que” en el texto original; o bien,
            - El verbo es “estar” conjugado en cualquier tiempo y persona pero seguido por otro verbo en gerundio (omitiendo palabras vacías entre ellos).
3. Con cada segmento de texto con el patrón gramatical del punto 1 y cuyo contenido satisface alguna de las condiciones del punto anterior, se construye un requerimiento candidato de la siguiente forma:
    1. El título de los requerimientos se forma con la primer ocurrencia del sub-patrón $SUST^{*}+VERB^{*}+ADJ^{*}+ADV^{*}$
    2. La descripción se forma con la totalidad del segmentos de texto; incluyendo el sub-segmento con el se formó el título.
    3. En caso de que el título o descripción extraídos no cumplan con las condiciones establecidas en el requerimiento 3.1.1 – 6, se descarta el requerimiento.
    4. El resto de los atributos tienen valores vacíos o por omisión que deben ser asignados por el usuario.

4. A la par de la extracción de los requerimientos, se construye también el glosario candidato de la siguiente forma: por cada segmento de texto en el que se identifique un requerimiento candidato, se realiza lo siguiente:

   Se toman los tokens presentes el segmento de texto y se determina su relevancia de acuerdo al modelo TF-IDF descrito en la sección 3.3.1.1.3. Si la relevancia es mayor que un umbral determinado, se busca el token en las colocaciones descritas en la sección 3.3.1.1.2. En caso de que el token en cuestión se encuentre en una colocación, se registra la colocación en el glosario candidato; de lo contrario únicamente se registra el token. En caso de que el índice TF-IDF del token no alcance este umbral, el token es descartado.

3.3.1.3 Modelado de los datos – Compilación de una ERS

En esta última etapa, se presenta al usuario la información extraída de los documentos y se le permite editarla conforme a los requerimientos: 3.1.1 – 3 a 7, 3.1.2 – 16 y 18. En particular, los requerimientos 3.1.1 – 3 y 3.1.2 – 16, indican que la inspección de los requerimientos debe ofrecer una vista general de la ERS que se está generando. Los atributos cuyo valor no es determinado en la etapa de extracción, son asignados con un valor por defecto antes de ser presentados al usuario. Estos atributos y sus valores por defecto son (en concordancia con las secciones 3.3.1.2, 3.1.1 – 5 y 3.1.1 – 6):

- El título del documento es: Especificación de Requerimientos de Software del proyecto “<Nombre del proyecto>”.
- Los alcances llevan la siguiente descripción: El presente documento contiene los requerimientos que describen el proyecto de software “<Nombre del proyecto>”. De acuerdo con Sommerville (2011) y el estándar ISO/IEC/IEEE 29148:2018; una Especificación de Requerimientos de Software describe un producto de software, indicando las funciones y cualidades que debe cumplir y las expectativas que se tienen de ellas. La información contenida en este documento describen estos aspectos de “<Nombre del proyecto>”; pero salvo que se ejemplifique explícitamente, no abarca mecanismos ni implementaciones de sus funciones o cualidades.

  Recordemos que de acuerdo al requerimiento 3.1.1 – 1, además de los documentos de entrada , la herramienta recibe el nombre del proyecto al que corresponden los documentos; este valor se usa para sustituir el <Nombre del proyecto> en la descripción anterior. Esta descripción es una síntesis tomada de la introducción de la sección 2.3 de esta tesis; en la que se presentan ERS.
- La versión del documento se inicializa como “0.1”.
- Las referencias incluyen los siguientes elementos:
  - I. Sommerville, Software engineering. Boston: Addison-Wesley, 2011.
  - “ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering”, ISOIECIEEE 291482018E, pp. 1–104, 2018.
  - International Organization for Standardization y International Electrotechnical Commission, “[ISO/IEC 15288:2008] Systems and software engineering — System life cycle processes”. feb. 2008.
  - “ISO/IEC/IEEE International Standard – Systems and software engineering - Content of life-cycle information items (documentation)”, ISOIECIEEE 152892019E, pp. 1–86, 2019.
  - P. Bourque, R. E. Fairley, y I. C. Society, Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0, 3rd ed. Washington, DC, USA: IEEE Computer Society Press, 2014.
  - Object Management Group, “Essence — Kernel and language for software engineering methods”. oct. 2018.
  - P. Jalote, An Integrated Approach to Software Engineering, 3a ed. Boston, MA: Springer Science+Business Media, Inc., 2005.
  - R. C. Martin y M. Martin, Agile principles, patterns, and practices in C#. Upper Saddle River, NJ: Prentice Hall, 2007.

Estas son las referencias consideradas en la sección 2.3 de este documento para diseñar la salida y contenido de ayuda de la herramienta.

El único atributo del documento que no tiene valor por defecto y que debe ser provisto por el usuario es “Responsables de emisión”. El resto de los atributos son generados automáticamente en el momento en el que el usuario solicite generar el documento de salida y no pueden ser modificados directamente por los usuarios. La forma en la que se asignan estos atributos es la siguiente:

- La fecha de emisión se toma del tiempo del sistema anfitrión.
- Estado del documento – De acuerdo con los requerimientos 3.1.1 – 5 y 3.1.2 – 19, si no se inspeccionan los requerimientos de forma individual y no se completan todos los valores vacíos, se asigna el valor “No revisado”. Si la información se asignan todos los valores vacíos pero no se revisan todos los requerimientos; o bien se inspeccionan todos los requerimientos pero no se completan todos los valores vacíos, se asigna el valor “Revisión incompleta”. Finalmente; se asigna el valor “Revisado” cuando se hayan completado todos los valores e inspeccionado todos los requerimientos.
- La tabla de contenidos depende del atributo “cláusula de organización” de cada requerimiento. La estructura general de la tabla de contenidos es la siguiente (en concordancia con el requerimiento 3.1.1 – 7):

    1. Alcances y responsables de emisión.
    2. Requerimientos.
        - <Cláusulas de organización>.
    3. Glosario.
    4. Referencias.

Respecto a los requerimientos en el documento; en conformidad con el proceso de extracción descrito en la sección 3.3.1.2, los textos recolectados durante proceso de extracción de requerimientos, se muestran como requerimientos en la lista descrita en el requerimiento 3.1.1 – 3 son. Los textos se usan para formar el título y descripción de los requerimientos, mientras que se les asigna el estado por defecto “Identificado”. Estos tres atributos pueden ser modificados libremente por los usuarios.

Por su parte, el Identificador no puede ser editado por los usuarios; se asigna de forma automática al registrar requerimientos en la lista utilizando números naturales de forma creciente y empezando por 1. Finalmente; el resto de los atributos: prioridad, volatilidad, tipo, riesgo, cláusula de organización y motivaciones, no son inicializados con ningún valor y deben ser especificados por el usuario.

La cláusula de organización se utiliza para organizar los requerimientos en la tabla de contenidos del documento. En el requerimiento 3.1.1 – 6 se describen valores de entre los que los usuarios podrán elegir como cláusula de organización; algunos de los cuales permiten ingresar una sub-cláusula personalizada. En la tabla de contenidos, las sub-cláusulas serán agrupadas bajo subtítulos de acuerdo a la cláusula de organización de cada requerimiento; ofreciendo una organización más descriptiva.

3.3.2 Diagramas UML del diseño de Requex

Para concluir esta sección, se presentan los diagramas que representan la herramienta *Requex*; los cuáles reflejan diseño descrito en este capítulo (y correspondientes a la versión 2.0.0 del prototipo). En la Figura 1, se muestra un diagrama de paquetes que revela la organización de sus componentes.

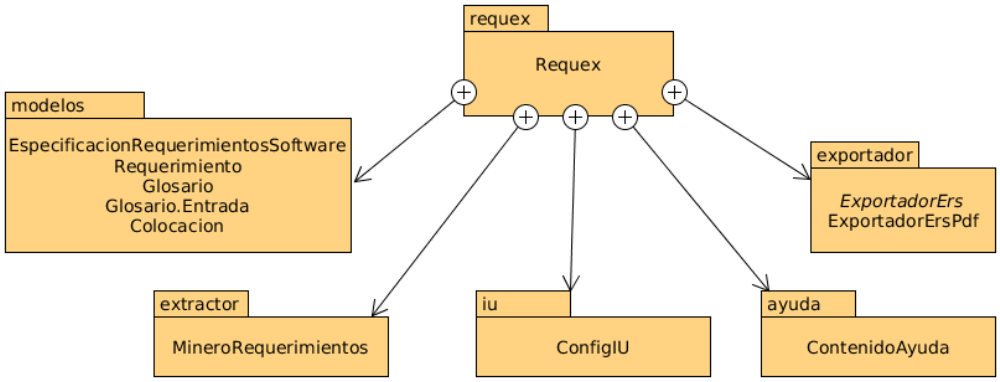

Figura 1: Diagrama de paquetes del prototipo

En el nivel más alto, tenemos el paquete *requex*, el cuál contiene una clase del mismo nombre. Esta clase, ofrece la extracción y análisis de requerimientos de Tesis y Reportes de Servicio Social; así como entrenar modelos del lenguaje a partir de documentos de entrenamiento. El proceso de extracción se vale de estos modelos para generar las colocaciones descritas en la sección 3.3.1.1.2, así como para determinar la relevancia y similitud de términos.

El paquete modelos contiene clases que modelan los objetos con los que trabaja la herramienta, principalmente Especificaciones de Requerimientos de Software y Requerimientos. Se modelan también otros elementos de las ERS; como el glosario y las colocaciones que puede incluir, con la intención de organizar las funciones específicas de estos elementos.

Por su parte, el paquete *extractor* contiene la clase que modela el proceso de extracción, y que a través de la clase Requex entrena los modelos para realizar la extracción de requerimientos o ejecuta el proceso de extracción sobre un corpus dado. Finalmente, el paquete *exportador* permite concentrar clases que a partir del estado de instancias de clases del paquete *modelos*, da algún formato de archivo a la información que contienen dichas clases para finalmente producir un documento ERS persistente. El comportamiento de los exportadores es modelado por la clase abstracta *ExportadorErs*.

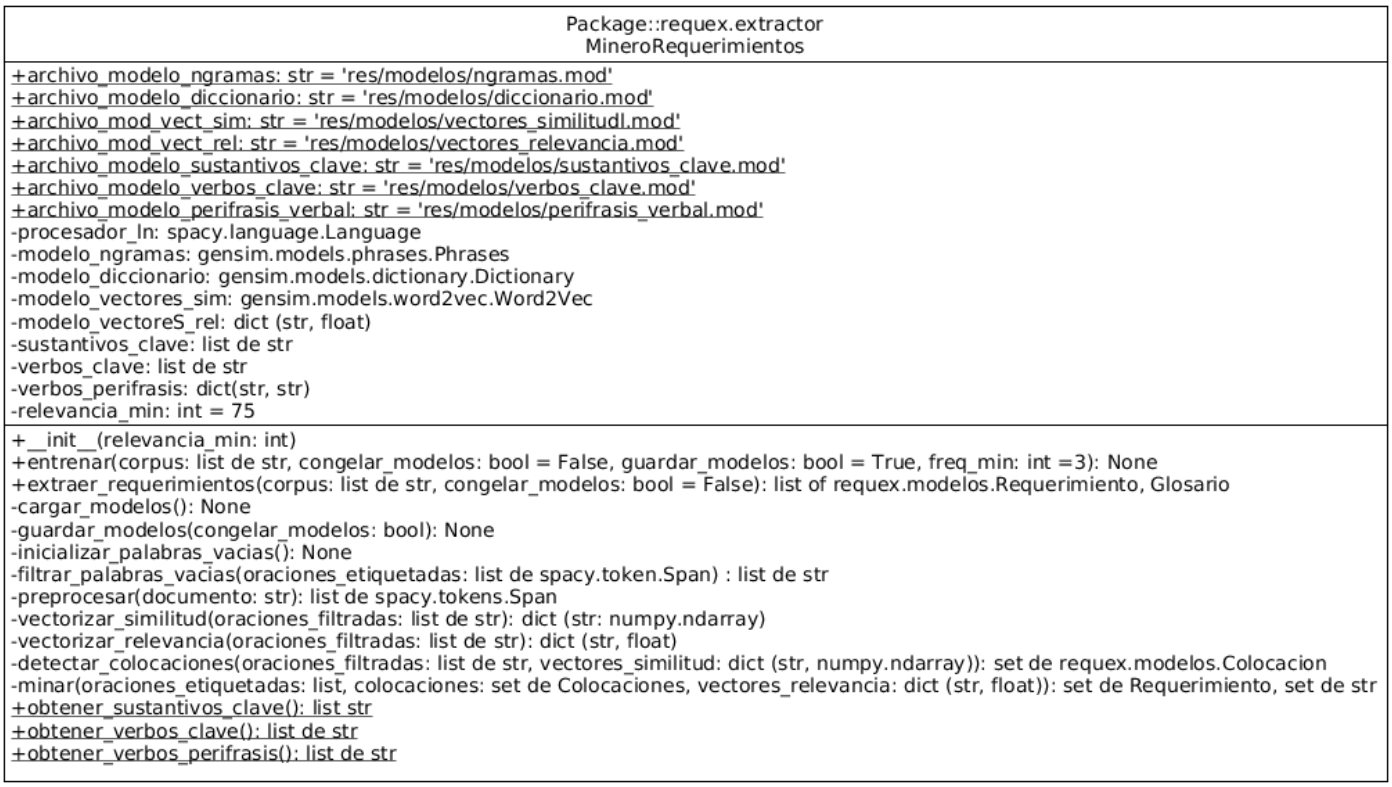

Figura 2: Diagrama de clase que modela el proceso de extracción

Los siguientes dos diagramas de clase modelan los dos procesos centrales del prototipo, el proceso de extracción de requerimientos y el proceso de análisis de requerimientos; a través de las clases *MineroRequerimientos* y *Requex* respectivamente. *MineroRequerimientos* (Figura 2) emplea las bibliotecas spaCy y Gensim para realizar el preprocesamiento y la exploración de los datos descrita en la sección 3.3.1.

La clase *Requex* (Figura 3) se vale del resto de los componentes del prototipo para obtener los requerimientos extraídos, presentarlos al usuario mediante una interfaz gráfica que permiten editar los atributos del documento ERS y de cada Requerimiento

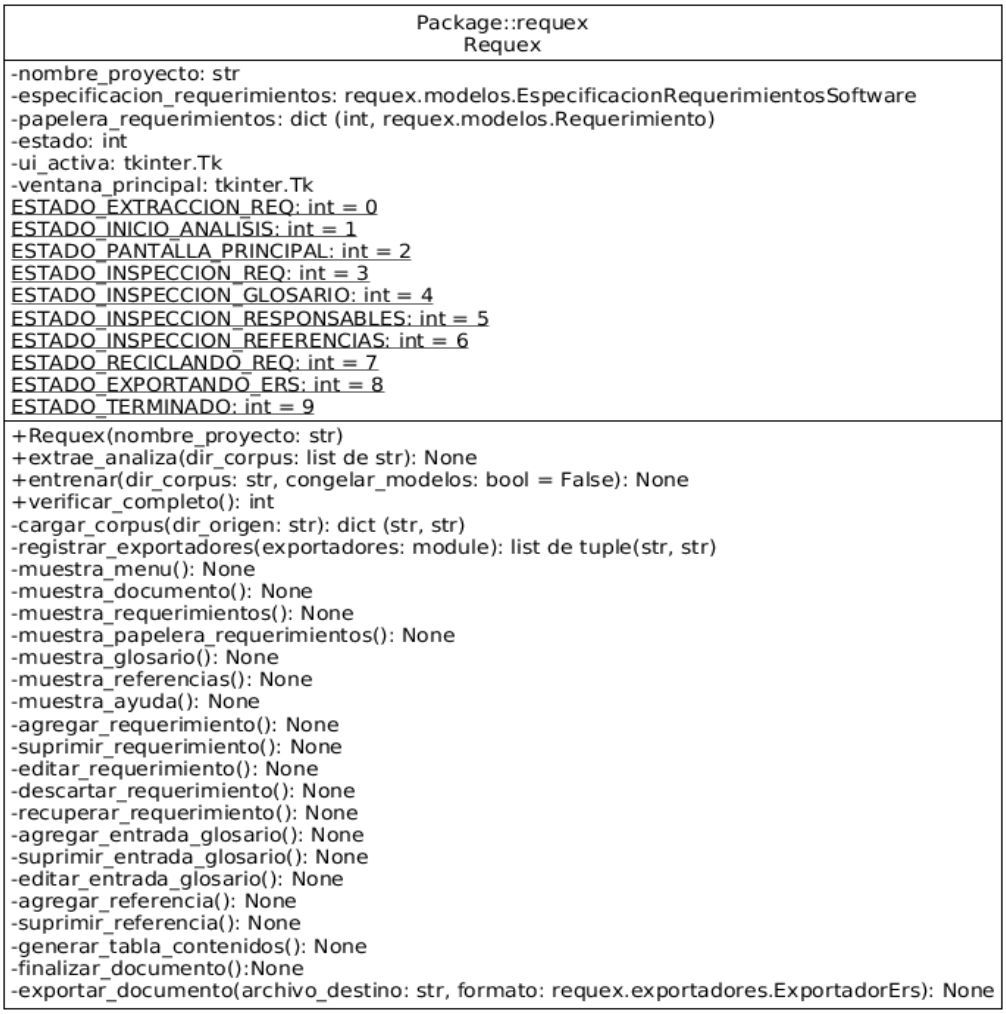

Figura 3: Diagrama de clase que modela el proceso de análisis

recuperado en los documentos de entrada que hayan sido provistos al ejecutar la herramienta (de acuerdo al requerimiento 3.1.1 – 1). Al final permite exportar el documento, validando el estado de los objetos ERS y Requerimientos de acuerdo a la sección 3.3.1.3

De forma similar a los anteriores, los siguientes dos diagramas de clase representan los modelos centrales del prototipo: Requerimiento (Figura 4) y Especificaciones de Requerimientos de Software (Figura 5). Sus atributos y comportamiento se definieron a partir de las secciones 3.1 y 3.3.1

Para terminar, se presentan dos diagramas de secuencias; que en conjunto modelan los procesos de extracción y análisis de Requerimientos. La Figura 6 describe la secuencia de extracción, en la que el usuario inicia la herramienta con un corpus que puede ser la ruta de un archivo de texto plano o de un directorio con archivos de texto plano. El texto de los archivos se provee a *MineroRequerimientos* para realizar la extracción, la cuál está basada en la sección 3.3.1

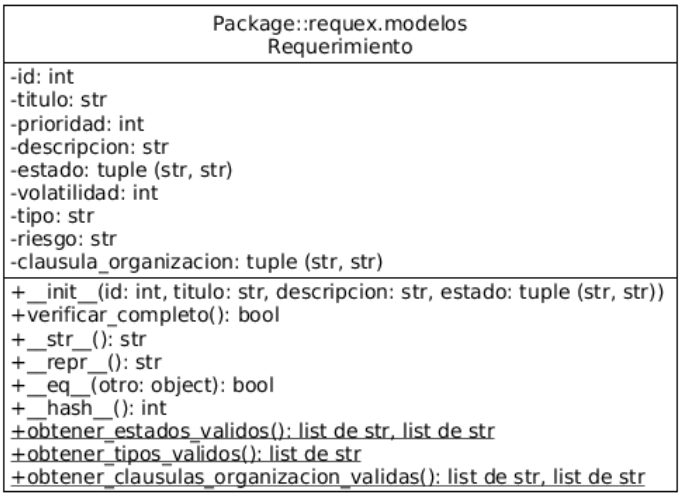

Figura 4: Diagrama de clase que representa un Requerimiento

El resultado del proceso de extracción es una instancia de *EspecificacionRequerimientosSoftware*, la cuál contiene los Requerimientos identificados de acuerdo a la sección 3.3.1 y el glosario de términos relevantes. Estas instancias, juegan un rol significativo en el diagrama de la Figura 7, donde se presenta el proceso de análisis y generación del documento ERS. Este proceso inicia con los resultados de la extracción, permitiendo al usuario inspeccionar los atributos de cada Requerimiento identificado, agregar o eliminar requerimientos, las entradas del glosario y el resto de los atributos de la ERS.

Cuando el usuario solicita finalizar, la clase *TablaContenidos* asiste en la compilación de la información para finalmente proveerla a un *ExportadorErs* que escribe el contenido de la ERS en un archivo persistente; conservando el formato descrito en la sección 2.3.4. Tras exportar el documento, la ejecución de Requex termina.

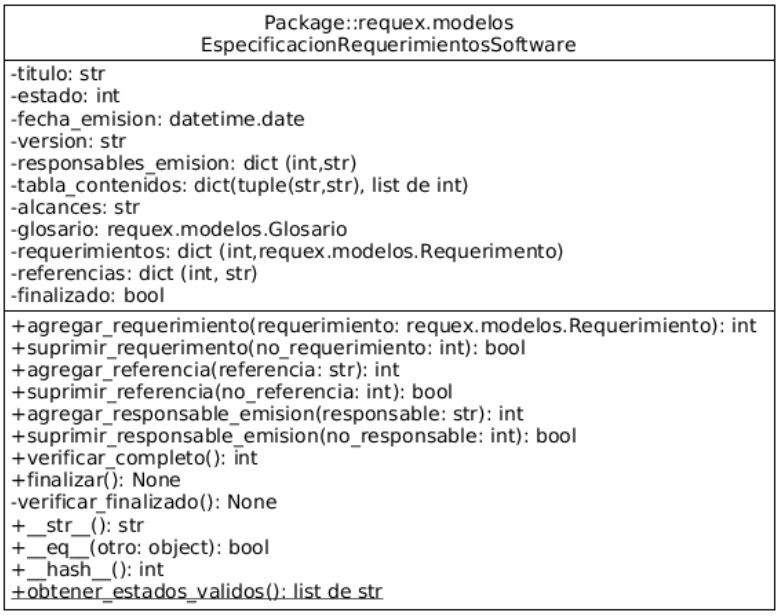

Figura 5: Diagrama de clase que modela una Especificación de Requerimientos de Software

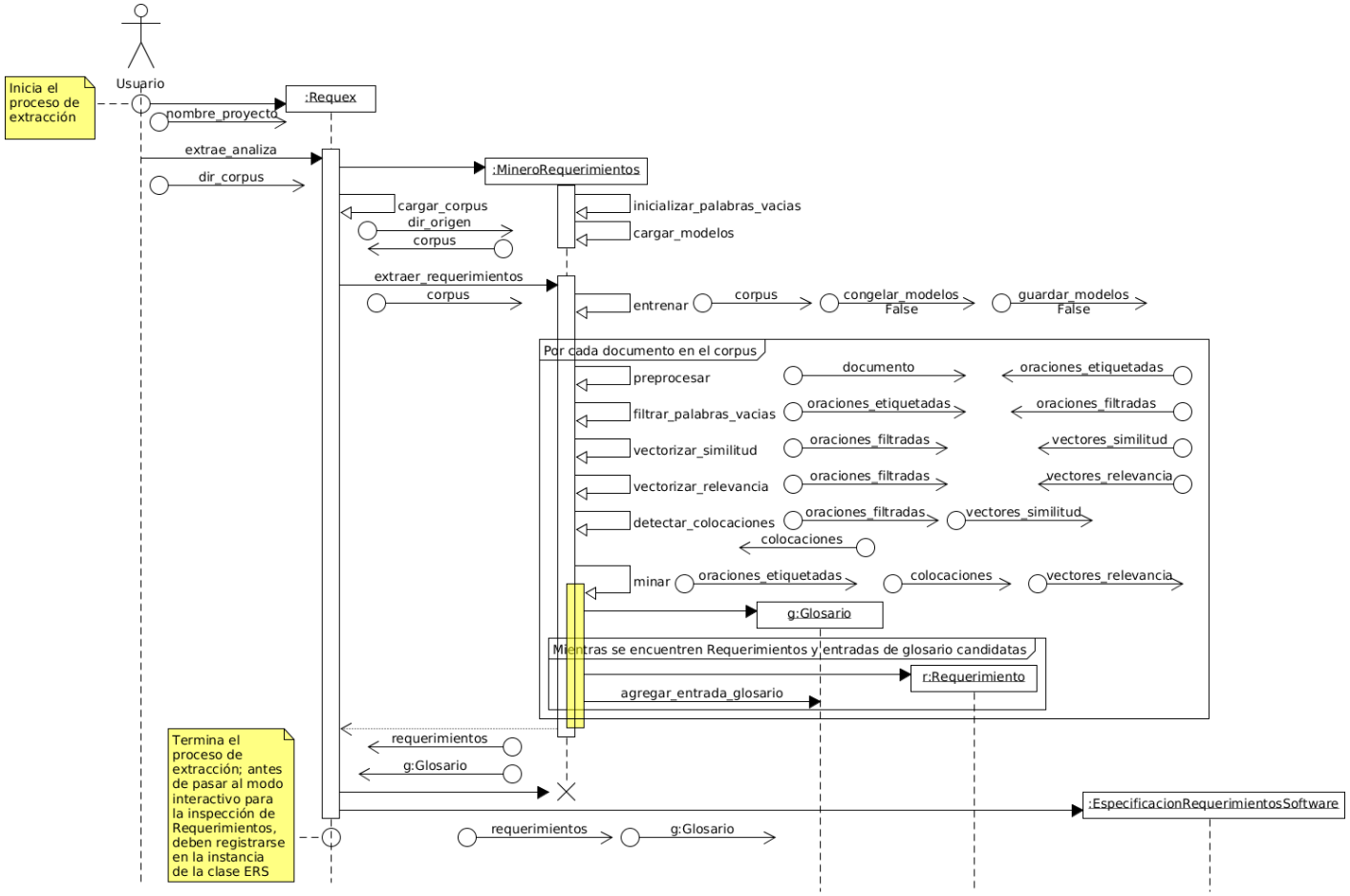

Figura 6: Diagrama de la secuencia de extracción de Requerimientos de Software

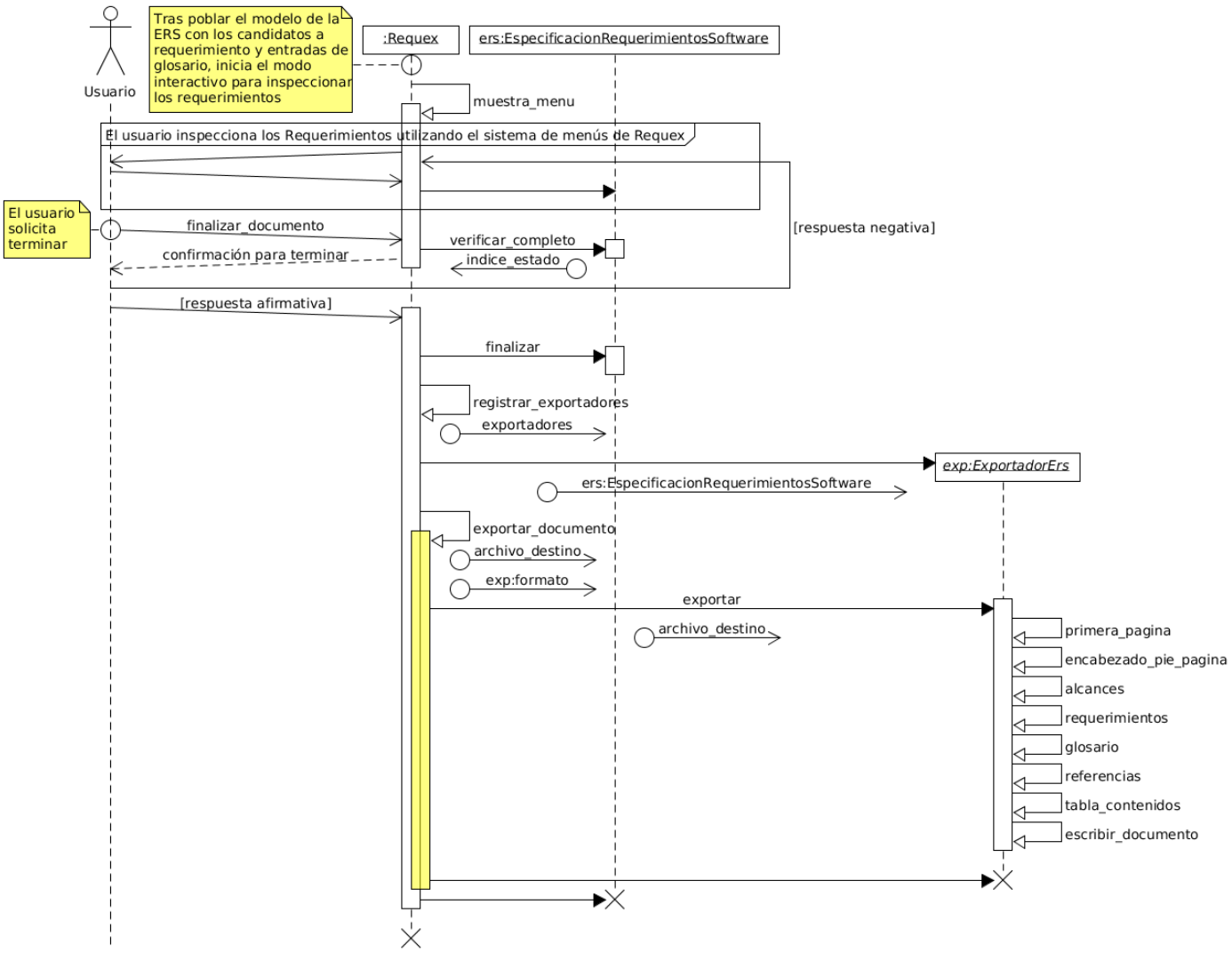

Figura 7: Diagrama de la secuencia de análisis y generación del documento

3.4 Resumen

En este capítulo se han presentado las estrategias y razones creadas a partir del contenido del capítulo 2, para diseñar la estructura y mecanismos del prototipo. El diseño general del prototipo se basa en las cinco operaciones de alto nivel con los que se suelen describir procesos de minería conforme a la sección 2.2.1; las cuáles son:

- **Recolección de los datos**. Esta operación es realizada por los usuarios del prototipo; quienes toman las Tesis y Reportes de Servicio Social que describan proyectos de software cuyos requerimientos deseen analizar.
- **Preprocesamiento**. Los documentos que provea el usuario son analizados con spaCy para etiquetarlo mofo-sintácticamente. Con esta información, se produce una lista de colocaciones; grupos de palabras frecuentes en los documentos. También se generan modelos para la relevancia y similitud de las palabras en los documentos; utilizando los modelos de Gensim con TF-IDF y Word2Vec respectivamente.
- **Exploración de los datos**. Utilizando las etiquetas morfo-sintácticas en los documentos, se buscan oraciones que posean el patrón gramatical $\left(SUST^{*}+VERB^{*}+ADJ^{*}+ADV^{*}\right)^{+}+P$ en las que se busca la existencia de un

sustantivo o verbo clave; o bien, se verifica si la oración presenta una perífrasis verbal. En caso afirmativo, se usa la oración para producir un requerimiento candidato.

El texto que conforma los requerimientos candidatos es explorado para tomar los términos que se determinen como los más relevantes. Estos términos relevantes son agrupados de acuerdo a su similitud semántica estimada o por aparecer en una de colocación.

- **Modelado de los datos**. Los requerimientos se modelan con base en los atributos identificados en la sección 2.3.4. Con el modelo de requerimientos, se define un modelo para el documento a producir; una Especificación de Requerimientos de Software. Este documento también se modela con base en la sección 2.3.4; por lo que incluye un glosario que se construye con los grupos de términos relevantes extraídos en la etapa anterior.

  Las referencias que se citan en este trabajo para determinar los atributos del documento y los requerimientos que contiene; así como los propósitos generales de una ERS, se usan como valores por defecto para los atributos referencias y alcances del documento.

- **Interpretación**. Se le presenta al usuario el documento ERS modelado en una interfaz interactiva, que le permite editar los atributos ingresados por defecto; así como completar la información del documento. Esto incluye, revisar los requerimientos y glosario extraídos, además de crear, editar y borrar registros (requerimientos, entradas del glosario, referencias, responsables de emisión, etc).

  Al solicitar exportar el documento, la herramienta genera automáticamente la fecha del documento y el estado del documento; el cuál depende del grado de completitud que tengan los atributos del documento.

El prototipo “Requex” es una aplicación multiplataforma escrita en Python 3.7, desarrollada bajo prácticas inspiradas en *extreme programming*. Requex ha sido definido a partir de 25 requerimientos, de los cuales 13 son funcionales, 10 son de interfaz y usabilidad; los 2 restantes son no funcionales.

El proyecto se encuentra disponible bajo licencia GNU GPL 3.0 en https://gitlab.com/nachintoch/requex

4 Experimentación

En este capítulo, se presenta el diseño y ejecución de las pruebas con las que se busca responder la hipótesis presentada en la sección 1.3; que en resumen consiste en determinar si es posible obtener documentación de proyectos académicos de software, aplicando PLN y de Extracción de Información para compilar ERS, ofreciendo ventajas en los procesos de software de dichos proyectos.

De acuerdo a la sección 3.2.2, el desempeño general del prototipo será evaluado con tres tipos de pruebas distintas. Puesto que las pruebas unitarias y de integración tienen el propósito de garantizar la cohesión del código y verificar su correcto comportamiento; estas pruebas resultan más útiles dentro del proceso de desarrollo [14], [17] que para validar la hipótesis del proyecto. Por su parte las pruebas de usabilidad ofrecen una vía para determinar la utilidad y usos que los usuarios perciben de la herramienta [37], [110].

Estas percepciones permiten estimar el impacto que tiene el prototipo en la documentación de los proyectos con los que se emplea, así como el potencial percibido de la herramienta en procesos de desarrollo y mantenimiento de software. Por su parte, la extracción de requerimientos se evalúa utilizando métricas para la minería de textos (sección 2.2.3.1); partiendo de precisión y exhaustividad.

4.1 Diseño de las pruebas

Los experimentos se dividen en dos grupos: pruebas de usabilidad y análisis de las puntuaciones de la extracción de requerimientos. En esta sección se presenta el diseño de cada grupo de experimentos de forma individual; en el capítulo 5 se realiza el análisis de resultados de estas pruebas, en las que se integra y correlacionan los resultados de ambos grupos de prueba. Se presenta primero el diseño del análisis del desempeño de la extracción; y después el diseño de las pruebas de usabilidad.

4.1.1 Evaluación del desempeño de la Extracción de Requerimientos

La evaluación del desempeño de la extracción de requerimientos, se realiza partiendo de las métricas precisión y exhaustividad [79]:

1. $precisión = \frac{VP}{VP + FP}$

2. $T_{VP} = exhaustividad = \frac{VP}{VP + FN}$

Puesto que es necesario conocer el número de verdaderos positivos (VP), verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN); para determinar el valor de estos parámetros, se catalogaron los resultados del proceso de extracción de requerimientos ejecutado durante cada una de las pruebas con usuario de la segunda iteración de desarrollo, en una de cuatro categorías: VP, VN, FP y FN. Los VP y FP se toman de los resultados tomados como candidatos a requerimientos por Requex, mientras que los VN y FN se toman de las oraciones rechazadas por la herramienta al extraer información.

Los resultados del proceso de extracción de requerimientos son registrados por la aplicación de forma discreta mientras se ejecuta; conforme a los requerimientos 3.1.1 – 10 y 3.1.3 – 24. Una vez que catalogaron todos los resultados en una de las cuatro categorías, se toma el número de registros en cada una de las categorías que representan los parámetros

VP, VN, FP y FN como el valor del parámetro correspondiente. Finalmente, se sustituyen los valores obtenidos en las ecuaciones de los indicadores precisión y exhaustividad.

Para analizar los valores resultantes de estos cuatro indicadores a lo largo de todas las sesiones de prueba, se calcula el valor-F de cada sesión. El valor-F combina los valores de precisión y exhaustividad de forma que los valores de mayor magnitud y más similares entre sí; resultan en un valor cercano a uno en la escala de valor-F (que se encuentra en el intervalo real [0, 1]) [79], [111]. *Precisión* indica el grado de éxito de identificar la información correcta de los textos de entrada (en este caso, requerimientos); mientras que *exhaustividad* estima la proporción de requerimientos que se pueden recuperar del texto [79], [111].

Si los valores de precisión y exhaustividad difieren significativamente, entonces se recupera una cantidad mínima de información; o bien, se recupera una enorme cantidad de información, cuyo contenido es poco relevante. Los índices de valor-F cercanos a cero, sugieren que se presenta alguno de estos dos problemas; o una combinación de ellos [79], [111].

$$valor-F = \frac{2 \times exhaustividad \times precisión}{exhaustividad + precisión} = \frac{2 \cdot VP}{2 \cdot VP + FP + FN}$$

Las oraciones procesadas durante la extracción de requerimientos se agrupan utilizando K-medias; por medio de los vectores con los que se representa cada oración y la distancia entre ellos. Estos vectores provienen del modelo del lenguaje que se genera con spaCy; la distancia entre ellos es distancia coseno, que se utiliza para representar similitud semántica [67]. De cada grupo de oraciones, se calculan los índices exhaustividad, precisión y valor-F. Este resultado se gráfica para su interpretación en la sección 4.3, en combinación con los resultados de las pruebas de usabilidad.

Conocer los tres índices considerados para evaluar el desempeño de la extracción de requerimientos de cada grupo, permite estimar el grado de éxito en la extracción de información respecto a las clases de oraciones resultantes [79], [111]; lo cuál es importante puesto que el proceso de extracción está basado en la estructura gramatical o la presencia de perífrasis verbal de cada oración en los documentos de entrada.

Una de las estrategias centrales en el diseño de la herramienta propuesta en este trabajo, es la participación activa del usuario para modificar; corregir y completar la información extraída. Esto tiene la intención de reducir el impacto del desempeño del proceso de extracción de requerimientos, en la calidad de la Especificación de Requerimientos de Software que resulta de utilizar la herramienta.

Los resultados de la extracción de requerimientos durante la segunda iteración de desarrollo de la herramienta han sido analizados junto a las pruebas de usabilidad de dicha iteración. Las pruebas de la primer iteración buscan descubrir defectos de usabilidad para ser priorizados como parte de la planificación de la segunda iteración. Las pruebas de la segunda iteración, tienen como objetivo explorar la utilidad percibida por los usuarios en los proyectos de software en los que han participado.

4.1.2 Pruebas de usabilidad

De acuerdo a la sección 2.4, la usabilidad es la suma de características de un producto que lo hacen apto para sus usuarios bajo un contexto determinado. En dicha sección, también se señalan cuatro aspectos centrales de la usabilidad: calidad técnica, funcionalidad, facilidad de uso y experiencia del usuario. De acuerdo con esto, se debe identificar el perfil de los usuarios del prototipo y el contexto en el que se espera sea utilizado para estudiar sus aspectos de usabilidad [37], [112].

Los usuarios de Requex son participantes de proyectos académicos de software que están descritos en Tesis y Reportes de Servicio Social. El contexto bajo el que se espera se utilice la herramienta es en procesos relacionados al desarrollo o mantenimiento de los productos de software de los proyectos en los que participan los usuarios. Los aspectos de calidad técnica son atendidos por los procesos del desarrollo de la herramienta, principalmente las pruebas unitarias y de integración; así como por medio de las herramientas del repositorio remoto como el sistema de seguimiento de defectos y gestión de versionamiento.

El aspecto de la experiencia del usuario de la usabilidad, abarca características que exceden el alcance para el que realizan las pruebas de usabilidad [37], [110]; por lo que las pruebas se limitarán a ofrecer un panorama general de las impresiones de los usuarios al utilizar el prototipo. Por lo anterior, las pruebas de usabilidad se centrarán en determinar la calidad de la funcionalidad y la facilidad de uso de Requex. Estos aspectos se pueden estudiar utilizando diversas técnicas; en este proyecto se combinan los siguientes enfoques:

- **Pruebas con usuarios**. Consisten en observar las interacciones de los usuarios con el prototipo con el fin de descubrir problemas para realizar tareas con la herramienta [37], [110], [112].
- **Entrevistas y cuestionarios**. Se utilizan con fines diversos; como esclarecer casos de uso en etapas tempranas de diseño, elegir candidatos para participar en una prueba de usabilidad con base en sus competencias; o bien conocer las opiniones e impresiones de los usuarios tras utilizar el producto [37], [112].

Las pruebas de usabilidad se realizaron al final de cada iteración de desarrollo. Para determinar el perfil de usuario de los participantes, se definió un cuestionario de entrada en cada serie de pruebas (correspondiente a cada iteración de desarrollo). Las interacciones de los usuarios durante las sesiones de prueba fueron registradas en forma de video, capturando la pantalla del equipo en el que se ejecuta la herramienta durante la prueba. Contar con el registro total de las interacciones, permite analizarlas con detalle; ofreciendo una mejor detección de problemas de usabilidad [110], [112].

Se invitó a los participantes a consentir la grabación de su voz y rostro durante la sesión de prueba. Las expresiones de los usuarios mientras utilizan un software, son indicadores de su estado de ánimo e impresiones durante su interacción. Una estrategia recomendada para analizar estas reacciones, es utilizando heurísticas sobre las emociones expresadas para estimar la reacción emocional del usuario respecto al software. Las grabaciones de las sesiones serán analizadas utilizando este tipo de heurísticas [37], [110], [112].

Las pruebas se realizaron de forma no presencial. Durante el periodo de reclutamiento de usuarios para la prueba; el cuestionario de entrada correspondiente a la iteración de desarrollo para la que se está realizado la prueba, fue publicado por la Web para que pudiera ser llenado por los potenciales usuarios. Las sesiones de interacción con el prototipo se organizaron utilizando la aplicación Zoom[1], la interacción remota se realizó mediante Zoom; u opcionalmente Teamviewer[2] si es que el usuario lo prefirió. Por otro lado, las interacciones se grabaron con OBS[3].

Las sesiones de prueba requirieron dos personas: el usuario y un moderador que recibe al usuario, resuelve sus preguntas y lo guía por la sesión para que el usuario realice todas las acciones que ofrece el prototipo. Al finalizar cada sesión de prueba, se realizó una entrevista de salida con cada usuario; con la que se busca conocer sus impresiones respecto a las funciones del prototipo.

1 https://explore.zoom.us/meetings

2 https://www.teamviewer.com/es-mx/

3 https://obsproject.com/

Las sesiones de prueba se llevaron a cabo conforme a un protocolo de actividades, en el que se describe en orden las tareas que se deben realizar durante la sesión; incluido un guión para realizar la simulación del análisis de Requerimientos con Requex. Puede consultar el protocolo de la segunda iteración de desarrollo en el anexo A.

La obtención de usuarios de prueba; para ambas iteraciones de desarrollo, se realizó mediante divulgación con académicos, estudiantes y recién egresados que hayan o estén participando en un proyecto de software, con cuyo diseño estén familiarizado (aún a un alto nivel), y que además esté descrito en una Tesis o Reporte de Trabajo Social al que tenga acceso. Principalmente se obtuvo respuesta de estudiantes del ESIE.

De acuerdo con Nielsen [112], la mayoría de los problemas de usabilidad de un producto de software pueden ser expuestos con menos de 15 usuarios de prueba; con 5 usuarios es posible reconocer la mayoría de los problemas de usabilidad. Este fenómeno se ha reflejado en experimentos de usabilidad y de experiencia del usuario en diversas aplicaciones de la IPC (Interacción Persona-Computadora); por lo que algunos autores sugieren contabilizar el tiempo de pruebas en lugar de la cantidad de usuarios [113]. Por esto, las pruebas de usabilidad se han diseñado para ser realizadas con al rededor de 15 usuarios.

Los protocolos de las pruebas se basan en prácticas de diseño centrado en el usuario para descubrir defectos de usabilidad [37], [112]; y en las actividades recomendadas por el modelo AMITUDE, el cuál ofrece varias indicaciones para desarrollar productos con enfoque en la usabilidad [37]. De acuerdo con estos protocolos, los interesados en participar deben responder un cuestionario de entrada y agendar una fecha y hora para realizar la sesión. El cuestionario de entrada sirve para identificar el perfil del interesado; y determinar si es apto para participar en la prueba o no. De ser apto, este perfil también ofrece información útil para analizar las sesiones.

Además; en el cuestionario de entrada de ambas series de pruebas, se le pregunta a los usuarios si consienten la grabación de su rostro y/o voz durante la sesión de prueba. Estas grabaciones tienen el propósito de obtener un primer panorama general de las impresiones de los usuarios, se analizan utilizando una serie de heurísticas para la experiencia de usuario desde el enfoque de la usabilidad; ofreciendo información adicional respecto a la interacción que se registra durante la prueba [110].

Debido a las medidas sanitarias durante la pandemia, la comunicación con potenciales usuarios o con otros interesados para promover la participación en ambas rondas de pruebas fue limitada. Durante la segunda iteración, se intentó contactar con usuarios por medio de redes sociales y extender la convocatoria; aún cuando recabara más perfiles de usuario no aptos que los esperados. Sin embargo, las pruebas de la segunda iteración tuvieron lugar al final del semestre; incrementando las dificultades para contactar usuarios aptos, aunado a diversos problemas sociales que opacaron la convocatoria para participar en la prueba.

El cuestionario de entrada para las sesiones de prueba de la segunda iteración, puede ser consultado en el anexo B. A continuación se presentan los detalles específicos de cada ronda de pruebas.

4.1.2.1 Pruebas de la primer iteración – Defectos de usabilidad

El propósito de las pruebas de la primer iteración, fue identificar y priorizar defectos de usabilidad. Esta información resultó vital para la planificación de la segunda iteración de desarrollo. Por esto, en esta primer ronda de pruebas no se calculó el valor de los índices de extracción de información; estas funciones estaban planificadas para ser estudiadas durante la segunda iteración.

Las sesiones de esta serie de pruebas, consistieron en hacer que el usuario utilice Requex de manera guiada por el moderador. Durante la sesión, se le presentan las funciones de la aplicación al usuario y se le solicita realizar tareas muy específicas con ellas; las cuales se espera haría habitualmente durante un análisis de requerimientos con la herramienta.

Las descripciones y acciones solicitadas a los usuarios, se realizaron de manera que no se describen elementos en pantalla; evitando alterar la atención del usuario sobre el software, y con ello evitar alterar los resultados.

La entrevista de salida se limitó a conocer el grado de eficiencia, control y facilidad de aprendizaje del uso de la herramienta, consistiendo de 10 preguntas sobre las funciones de la herramienta que se debían responder en una escala del 1 al 5; donde uno representa "Totalmente en desacuerdo", tres "Opinión neutra" y cinco "Totalmente de acuerdo"; tomando cualquier observación adicional que haga el usuario.

4.1.2.2 Pruebas de la segunda iteración – Utilidad percibida

La ronda de pruebas de la segunda iteración de desarrollo, busca atender los objetivos específicos 1.2.2 de este proyecto (con énfasis en los objetivos 3 y 4); que abarcan la percepción de facilidad de acceso a la información de los proyectos contenida en los documentos de entrada de la herramienta y el valor agregado percibido de la herramienta en procesos de desarrollo y mantenimiento de dichos proyectos.

La entrevista está inspirada en el instrumento para evaluar usabilidad "SUMI" (Software Usability Mesasurement Inventory), que fue creado para medir la percepción de problemas de usabilidad por parte de los usuarios. Consiste en 50 reactivos que se agrupan en 5 dimensiones: *eficiencia*, *afecto*, *ayuda*, *control* y *facilidad de aprendizaje*. Cada pregunta se responde en la siguiente escala: "En desacuerdo", "Indeciso" y "De acuerdo" [110], [114].

Las dimensiones utilizadas para esta entrevista exploran los aspectos de usabilidad que evalúa SUMI, considerando también los objetivos del proyecto. Estas dimensiones; en orden de prioridad descendente son: *efectividad*, *facilidad de uso*, *control del usuario*, *facilidad de aprendizaje* y *afecto*. Los primeros tres aspectos se consideran los más relevantes puesto que de ellos se espera obtener la información más relevante para estimar la percepción de utilidad que tienen los usuarios respecto a utilizar Requex en el proyecto de software descrito por los documentos con los que se realice la sesión de prueba.

La entrevista de salida contiene 31 preguntas, que deben ser respondidas en la escala "En desacuerdo", "Indeciso" y "De acuerdo"; además, se le preguntan o toman las observaciones que haga el usuario respecto a cada grupo de preguntas en cada dimensión en la que se basa la entrevista. Finalmente, se incluyen tres preguntas adicionales para conocer la opinión general del usuario respecto a Requex.

Las sesiones de esta ronda de pruebas, consistieron en simular un análisis de requerimientos más extenso que con la primera ronda de pruebas. En la serie de pruebas anterior se solicitó a los usuarios realizar acciones específicas, haciéndolo utilizar las funciones de la herramienta al menos una vez; por lo que el análisis de requerimientos simulado se limitó a realizar tareas específicas. Por ejemplo: renombrar un requerimiento, cambiar la definición de una entrada del glosario, agregar un responsable de emisión; etc.

En la segunda ronda, se le plantean al usuario seis metas durante la simulación del análisis de requerimientos. En general, estas metas consisten en completar información respecto al proyecto descrito en los documentos de titulación que haya proporcionado para utilizar la herramienta. Para ello, los usuarios deberán realizar tareas específicas; como abrir una ventana o llenar un formulario, pero sin que se lo instruya directamente el monitor. Las metas se describen de forma clara pero sin indicar exactamente al usuario qué elementos en pantalla le permiten cumplir la meta.

El monitor observa la interacción del usuario; quien deberá realizar cada tarea en un lapso de aproximadamente 2 minutos. En caso de exceder este tiempo, el monitor debe guiar el usuario a la siguiente tarea o meta; conservando los cambios incompletos. Estas metas se basan en la información que de acuerdo a los estándares y recomendaciones presentados en la sección 2.3, debe estar presente en una ERS; así como en los objetivos 3 y 4 de este proyecto, los

cuales buscan que el uso de la herramienta ayude a reflexionar el diseño del software en cuestión y aportar conocimiento útil a los proyectos que describen los documentos de entrada.

El análisis de las interacciones e impresiones de los usuarios, permite descubrir problemas de usabilidad en las tareas más importantes que ofrece Requex. La libertad de interacción que estas metas le permiten a los usuarios, también ofrece un mayor beneficio al ser seguido por un cuestionario de usabilidad como SUMI [110], [114]. Por esto, la entrevista de salida de esta ronda de pruebas cumple las características para conocer la utilidad que perciben los usuarios de Requex para analizar los requerimientos del proyecto descrito en los documentos de entrada.

Puede consultar el cuestionario en el que se basa la entrevista de salida de la ronda de pruebas de usabilidad de la segunda iteración en el anexo C. Los resultados se presentan en el siguiente capítulo; en el que se analizan los resultados junto a los índices del desempeño de la extracción de requerimientos, respecto a los objetivos del proyecto para responder a la hipótesis del proyecto.

4.2 Resultados de las pruebas

En la presente sección, se presentan y analizan los resultados de los dos grupos de pruebas.

4.2.1 Primera iteración de desarrollo – Defectos de usabilidad

La primera serie de pruebas contó con cinco usuarios y se realizó del lunes 19 de abril al viernes 30 de abril del 2021. Cada sesión tuvo una duración aproximada de 35 minutos; acumulando 2.9 horas de prueba. En total, se recibieron siete registros de candidatos a usuarios de prueba; sin embargo, dos de estos candidatos no fueron aceptados por no cumplir con el perfil buscado y no completar el proceso de registro.

Cuatro de los cinco usuarios de prueba consintieron la grabación de voz y rostro durante la prueba. Esta prueba fue realizada con la versión 0.8.0 de Requex; que al final de la prueba fue aprobada para ser etiquetada como la versión de lanzamiento 1.0.0. El análisis de las sesiones y las entrevistas realizadas al final, permitieron identificar y priorizar 13 defectos de usabilidad.

En la entrevista de salida, se preguntó a los usuarios su opinión respecto a las funciones de la herramienta y las impresiones que tuvieron al su usarlas. En la siguiente tabla, se presentan los 13 defectos descubiertos tras esta serie de pruebas. La prioridad sigue la convención establecida en la sección 3.1.1 – 6; 0 representa la mayor prioridad y 10 los defectos menos importantes.

No. Defecto	Descripción	Requerimientos involucrados	Usuarios que lo observaron	Prioridad
1	Visibilidad y claridad en la pantalla inicial. La ventana en sí misma es muy pequeña y no recibe al usuario en alguna manera; simplemente espera que empiece a realizar acciones.	14, 22	2 y 4	4
2	Posición aleatoria de ventanas. Cada pantalla de la aplicación se abre en su propia ventana; y aparecen en una posición aleatoria en el escritorio.	23	1 y 5	2
3	Falta de retroalimentación. Al ejecutar acciones o completarse una carga; no hay una clara retroalimentación al usuario.	10, 18, 19, 22, 23	1	*8*

4	Claridad de mensajes. En particular los mensajes de error no son claros al usuario.	11, 18, 19, 22	1	*9*
5	Uso de las listas de inspección. Las interacciones con las listas de inspección no son claras; no todos los usuarios perciben que es posible desplazarse por la lista y su contenido es difícil de leer.	16, 20, 23	1, 2, 5	**0**
6	Información adicional en las listas de inspección. Ofrecer un mejor y breve resumen del contenido de cada entrada en las listas de inspección de atributos de la ERS.	16, 22, 23	1 y 5	3
7	Añadir grado de participación de los responsables de emisión.	5, 15	1	*9*
8	Uso de los selectores en formularios. Las interacciones con los selectores son confusas y ajenas a lo que los usuarios esperan por un selector.	17, 18, 20, 23	1, 4 y 5	**0**
9	Claridad sobre la posibilidad del uso de caracteres especiales en campos de texto.	1	2 y 4	*8*
10	Claridad y orden de elementos en la pantalla principal. Los usuarios perciben desordenada la pantalla principal, por lo que les es complicado explotar las funciones que ofrece.	15, 22	1, 2, 3, 4 y 5	**1**
11	Claridad sobre la división del valor del estado de los requerimientos.	6, 11, 15 y 17	3	*9*
12	Dividir el campo de referencias en sus distintas partes (autores, título, fecha, publicación, etc).	5, 15	5	*10*
13	Expectativas al usar el botón de cerrar ventana como parte de la navegación de la IGU de la aplicación. Los usuarios lo usan al no encontrar un botón de volver, pero con la incertidumbre de cerrar la aplicación.	23	1, 2, 3, 4 y 5	**1**

La prioridad de los defectos se estableció a partir del número de usuarios que lo observaron, la prioridad de los requerimientos con los que se relaciona y el impacto percibido del defecto en la experiencia del usuario. Los defectos con prioridad menor a 2 se atendieron antes que los requerimientos no implementados o cuya implementación debiera ser modificada en la segunda iteración de desarrollo.

Los defectos con prioridad entre 2 y 7 fueron atendidos después de los requerimientos no implementados en la primer iteración y antes que los requerimientos cuya implementación fuera a ser modificada. Aquellos defectos con prioridad mayor a 7 se consideran defectos menores; y únicamente fueron atendidos si compartían una estrecha relación con una tarea dentro del alcance de la iteración.

De estos 13 defectos, se extrajeron 10 tareas; a las que se le suman 7 tareas de corrección de problemas de programación y 8 tareas correspondiente a la implementación de los requerimientos no o parcialmente implementados

durante la primer iteración. En total, se identificaron 25 tareas con las que se conformó el plan de trabajo para la segunda iteración.

4.2.2 Segunda iteración de desarrollo

La segunda serie de pruebas se realizó del lunes 21 de junio al viernes 2 de julio del 2021. Contó con cinco usuarios de prueba. Cada sesión de prueba tuvo una duración aproximada de 55 minutos; acumulando 4.5 horas de prueba. De manera similar a la iteración anterior; las pruebas de la segunda iteración recibieron siete candidatos a usuario de prueba, de los cuales uno no contaba con el perfil necesario para participar y el otro no se presentó a la sesión.

Cuatro de los cinco usuarios de prueba, consintieron la grabación de su rostro y voz durante la sesión. Esta prueba fue realizada con la versión 1.23.0; que al final de la prueba fue aprobada para ser etiquetada como la versión de lanzamiento 2.0.0.

4.2.2.1 Desempeño de la extracción de información

Durante la ronda de pruebas de usabilidad de la segunda iteración, se recolectaron 21 documentos; correspondientes a:

- Dos Tesis (4 documentos).
- Un reporte de actividad docente (1 documento).
- Un reporte de servicio social (1 documento).
- Un reporte de apoyo a la divulgación (15 documentos).

La razón de la multiplicidad de documentos que representan un mismo proyecto de titulación, se debe a que algunos documentos fueron provistos separados por capítulos; además de que algunos usuarios proporcionaron documentos anexos y reportes generados durante las actividades del proyecto descrito (por ejemplo, los anexos de este proyecto generados durante las pruebas de usabilidad).

A partir de estos 21 documentos, el procesador de lenguaje natural de Requex identificó 7,296 oraciones en español. En conformidad con la sección 3.3.1.2, en cada oración se buscó detectar el patrón gramatical que sugiere la presencia de un requerimiento; o bien, una perífrasis verbal. Se han clasificado las oraciones procesadas en los siguientes grupos:

- *Falsos positivos* (FP). Aquellas oraciones que hayan tomado como requerimiento por el prototipo, pero que no describen o sugieren un requerimiento (una descripción de funciones o características específicas que debería satisfacer el software descrito en los documentos de entrada; sección 2.3).
- *Verdaderos positivos* (VP). Aquellas oraciones tomadas como requerimiento que sí describen o sugieren un requerimiento.
- *Falsos negativos* (FN). Agrupa las oraciones rechazadas como requerimiento, que describen o siguieren un requerimiento.
- *Verdaderos negativos* (VN). Oraciones rechazadas que no describen ni siguieren un requerimiento.

El resultado de esta clasificación se presenta a continuación:

	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5
Falsos positivos	234	205	236	567	105
Verdaderos positivos	93	92	120	81	18
Falsos negativos	2	27	25	41	3
Verdaderos negativos	1,223	1,181	1,137	1,811	95
Requerimientos candidatos mostrados al usuario	327	297	356	648	123
Tamaño del corpus (oraciones)	1,552	1,505	1,518	2,500	221

Con estos valores, se han calculados los índices precisión, exhaustividad y valor-F para analizar el desempeño de la extracción de requerimientos. A continuación se presenta una tabla con el valor de estos tres índices para los documentos de cada usuario:

	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5
Precisión	28.440%	30.976%	33.707%	12.5%	14.634%
Exhaustividad	97.894%	77.310%	82.758%	66.393%	85.714%
Valor-F	44.075%	44.230%	47.904%	21.038%	25%
Tipo de documento	Reporte de apoyo a la divulgación	Tesis	Reporte de servicio social	Reporte de actividad docente	Tesis

Las frases se agruparon utilizando el algoritmo K-medias y los vectores que representan las oraciones; generados por el procesador de lenguaje natural utilizando distancia coseno, que estima similitud semántica en el intervalo real [0, 1]. Una similitud con valor 0 significa que las oraciones tienen sentidos totalmente diferentes, mientras que una similitud de 1 indica que las oraciones son sinónimos. Se obtuvieron 20 grupos de oraciones; con más grupos aparecen múltiples grupos con poca o ninguna distinción entre sí y de pocos registros, con menos grupos los registros se acumulan de manera desproporcionada en una fracción de ellos.

Con estos 20 grupos, las oraciones que distribuyen de manera más uniforme. El propósito de generar y presentar estos tópicos, es explorar el desempeño de la extracción de requerimientos desde el enfoque del tipo de oraciones contenidas en los documentos, lo que nos permite criticar las reglas de extracción y esclarecer los resultados generales presentados en las dos tablas anteriores; buscando aprovechar mejor la limitada cantidad de pruebas que se lograron realizar.

Puesto que los grupos se determinan por la similitud entre oraciones; bajo el contexto de minería de textos, estos grupos representan tópicos (sección 2.2.2.8), conjuntos de palabras, textos u oraciones cuyo significado es similar bajo los modelos del lenguaje (en este caso la representación vectorial de las oraciones). De estos 20 tópicos, 8 están

compuestos por completo de oraciones clasificadas como *Verdaderos Negativos*; por lo que los índices de estos tópicos se evalúan en cero. En resumen, estos grupos de VN son los siguientes:

1. **Tópico 0**, tamaño: 40 – Oraciones que contienen series de puntos y espacios " ." y números separados por puntos; básicamente se trata de entradas en una tabla de contenidos.
2. **Tópico 5**, tamaño: 85 – Agrupa oraciones con abundancia de símbolos cómo: $, (,), {, }, [,]; y que en general representan fragmentos de código en algún lenguaje de programación.
3. **Tópico 6**, tamaño: 295 – Se compone de títulos y etiquetas de contenido; como "`84 capítulo 6`", "`figura 12`" y " `25 \x0c● imagen 4`".
4. **Tópico 13**, tamaño: 250 – Es muy similar al tópico 6, con la excepción de ser predominado por títulos de figuras. La principal diferencia con el tópico 6, es que las oraciones en este tópico suelen ser más cortas, contienen menos caracteres especiales y números separados por puntos; como en " `44 figura 3.7`".
5. **Tópico 14**, tamaño: 168 – Se caracteriza por frases que incluyen una cita en formato APA o una expresión corta entre paréntesis.
6. **Tópico 17**, tamaño: 495 – Similar al tópico 13. La diferencia entre estos dos tópicos, es que el tópico 13 suele incluir títulos de figuras enumerados con números; mientras que los títulos de figuras en el tópico 17 poseen una enumeración que incluye letras. Por ejemplo: "`141 figura b.18`".
7. **Tópico 18**, tamaño: 51 – Estas frases se caracterizan por incluir fechas, por ejemplo: "`consultado el 16 de enero de 2020`" y "`del 14 al 18 de octubre 2002`".
8. **Tópico 19**, tamaño: 236 – Contiene enumeraciones de contenido, como "`2.2`".

En suma, las oraciones clasificadas en estos 8 tópicos de VN representa el 22.2% del total del corpus procesado. En la siguiente gráfica (figura 8), se muestra el valor de los índices precisión, exhaustividad y valor-F para cada uno de los 12 tópicos que contienen oraciones de clasificación mixta (y que por ende sus índices pueden evaluarse a un valor distinto de cero).

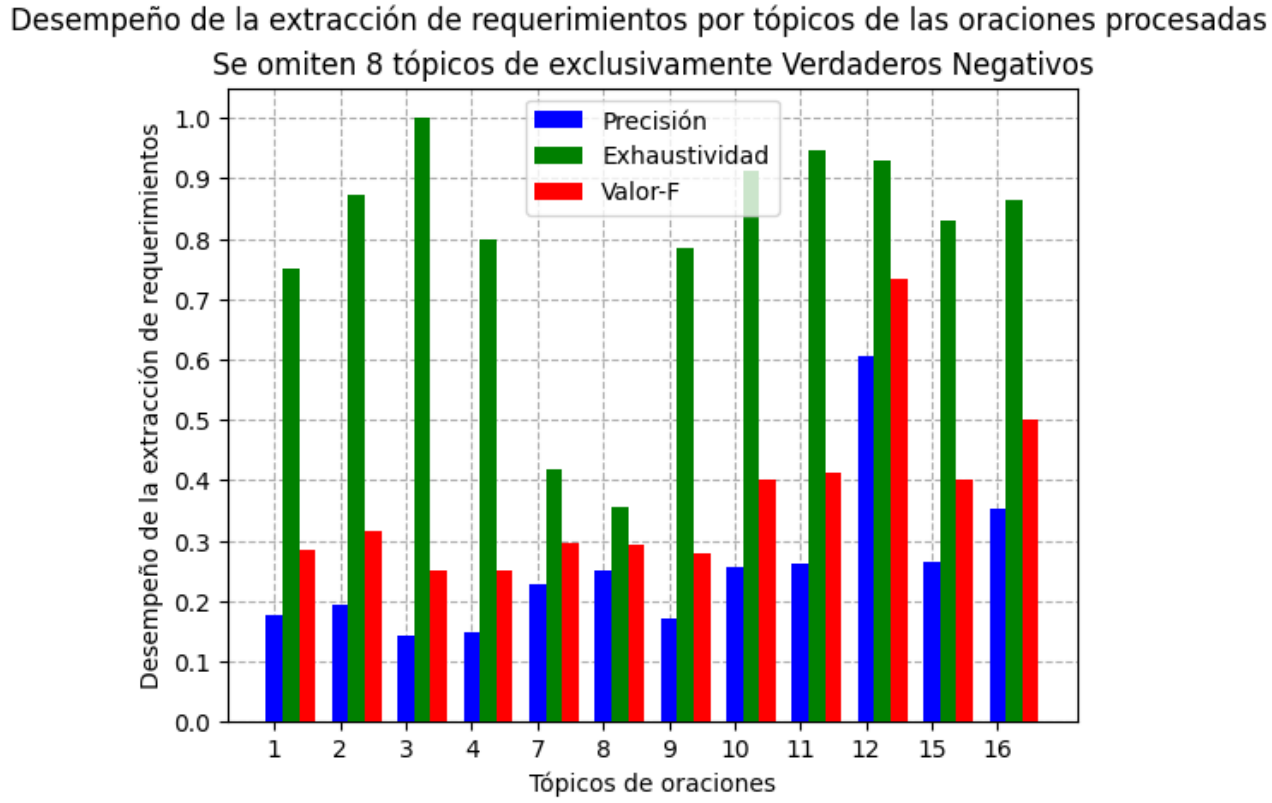

Figura 8: Índices del desempeño de la extracción de requerimientos

A grandes rasgos, estos tópicos se describen con las siguientes características:

1. **Tópico 1**, tamaño: 435 – Contiene oraciones que mencionan aspectos de usabilidad y diseño de software. Algunas oraciones en este tópico son: "cuestionario de usabilidad", "especificación de requerimientos", "patrones arquitectónicos de software" y "diagrama de navegación".
2. **Tópico 2**, tamaño: 383 – Este tópico está formado por oraciones largas que describen el comportamiento que se espera de un software o explicaciones de diversos principios teóricos.
3. **Tópico 3**, tamaño: 198 – Consiste de frases que se caracterizan por alguna Entidad Nombrada (principalmente nombres de personas, institutos y congresos).
4. **Tópico 4**, tamaño: 599 – Agrupa oraciones que describen funciones que se esperan de un producto de software; o las decisiones de diseño entorno a dichas funciones.
5. **Tópico 7**, tamaño: 351 – Se conforma por descripciones de mecanismos y estrategias en torno a funciones de un producto de software. Se distingue del tópico 2 principalmente debido a la precencia de letras griegas y expresiones matemáticas; además de que estas descripciones son más cortas.
6. **Tópico 8**, tamaño: 263 – Es un tanto similar a los tópicos 13 y 17 (sus oraciones incluyen títulos de figuras), pero estas oraciones son más largas y algunas incluyen una URL en lugar de un título de contenido.
7. **Tópico 9**, tamaño: 362 – Las frases en este tópico principalmente describen estrategias y explicaciones de principios empleados en los documentos de entrada. Se distingue del tópico 2 puesto que estas oraciones son menos técnico-teóricas (se trata de explicaciones en una jerga general) y más cortas; se distingue del tópico 7 por carecer de expresiones matemáticas. Ejemplos:
   - cuando se concluyan los cambios deseados, se verificará nuevamente que no existan errores de tipos, y de ser así, se redirigirá al perfil del proyecto editado, mostrando un mensaje de alerta notificando que la tarea se realizó exitosamente.
   - debido a que los proyectos no suelen seguir un flujo secuencial, pues suelen requerirse cambios durante el desarrollo, en este modelo dichos cambios podrían causar confusión y conflictos para los desarrolladores pues requieren terminar con las actividades establecidas en la fase que se realiza.
   - al traducir el modelo entidad-relación a modelo relacional, se generaron algunas tablas no consideradas previamente, como ésta, que contiene las consultas para obtener información de aquellas instituciones o integrantes del grupo, que colaboran en un proyecto.
8. **Tópico 10**, tamaño: 352 – Se caracteriza por oraciones con explicaciones generales breves; similares a las del tópico 9, pero aún más breves. Ejemplos:
   - además permiten anticipar errores en la organización y corregirlos antes de comenzar la construcción de código.
   - algoritmos de clasificación en la minería de texto 21 neuronales de alimentación multicapa pueden modelar la predicción de una clase como una combinación no lineal de las entradas.
   - proceso de evaluación asistida: en la metodología actual, la asignación del nivel de competencia de las repuestas del cuestionario es una tarea manual y puede tomar bastante tiempo.

9. **Tópico 11**, tamaño: 499 – Contiene oraciones que principalmente hablan sobre temas en torno a la minería de textos y a la ingeniería de software. Se distingue del tópico 1 por consistir de oraciones más largas con ideas completas.
10. **Tópico 12**, tamaño: 108 – Agrupa oraciones que describen interfaces gráficas de usuario.
11. **Tópico 15**, tamaño: 484 – Está conformado por justificaciones y decisiones de diseño principalmente. Se distingue del tópico 10 por contener oraciones más largas; siendo más similar al tópico 9.
12. **Tópico 16**, tamaño: 321 – De forma similar al tópico 4, se compone por frases que describen elementos de un producto de software y su comportamiento. La principal diferencia, es que las oraciones en este tópico describen elementos puntuales del software; como elementos gráficos y funciones específicas, mientras que en el tópico 4, las oraciones describen estas características de forma más general.

Podemos notar que el índice de exhaustividad suele tener un valor alto (tanto por usuario como por tópico); mientras que precisión suele tener valores bajos, reduciendo a su vez el valor-F. Esto significa que si bien Requex es capaz de extraer una alta proporción de los requerimientos en los documentos de entrada, los resultados incluyen una proporción considerable de oraciones que no representan requerimientos.

Es importante considerar que por la naturaleza de los documentos de titulación que consume la herramienta, contienen amplias descripciones del proyecto de software que describen y su contenido ahonda en la teoría aplicada en el diseño del producto en cuestión. Muchas de las descripciones en los tópicos caracterizados por decisiones de diseño, la descripción de elementos en una interfaz o los motivos tras una funcionalidad de software; no necesariamente representan o sugieren un requerimiento. Por ejemplo, los siguientes resultados se consideran no requerimientos:

- Del tópico 4 – `prueba tesistest en la ruta test/unit/models/. una vez creado el archivo para los casos de prueba, entonces se definen los métodos con ayuda de aserciones (assertions) de phpunit.`
- Del tópico 12 – `“presentar situaciones cotidianas fácilmente interpretables por los estudiantes...” “susceptibles de ser representadas de diversas formas...” “posibilitar un proceso de reelaboración de explicaciones a lo largo del instrumento...” ([1], página 6).`
- Del tópico 16 – `el procedimiento es similar al anterior, aunque implica colocar un límite superior especificado, en los multiplicadores de lagrange, i. el límite superior se determina mejor experimentalmente según [2].`

En la gráfica de la figura 8, podemos observar que los tópicos 7 y 8 presentan el peor desempeño general. Tras la primer iteración de desarrollo, se detectó que las URL representan un origen importante de *Falsos Positivos*, por lo que la versión 2.0.0 de Requex incluye un filtro que descarta una oración cuando más del 50% de la oración es una URL. El bajo desempeño del índice de exhaustividad, sugiere que este filtro es demasiado estricto; descartando muchos candidatos a requerimientos que incluyen una URL.

En cuanto al tópico 7, la presencia de caracteres especiales puede causar problemas con los modelos del lenguaje que se emplean para analizar los textos de entrada; o bien, pueden ser identificados como código de programación o de marcado, los cuáles también se identificaron como un origen relevante de *Falsos Positivos*. Debido a lo anterior, también se creo un filtro para descartar oraciones cuyo contenido sea dominado por la presencia de código. De forma similar al tópico 8, los índices sugieren que los filtros de código son demasiado estrictos.

En el resto de los tópicos no existe una predominancia de caracteres especiales o secuencias de texto no en español. En particular, el tópico 12 es el que presenta el mejor desempeño general; la forma en la que se describen las interfaces de usuario permite entender casos de uso y las necesidades que atienden los productos de software descritos en cada documento, de donde se pueden tomar candidatos a requerimientos.

Del resto de los resultados es claro que bajo la estrategia empleada, se requieren filtros adicionales que deben ser calibrados de forma que no reduzcan el índice de exhaustividad (mientras incrementan la precisión, y con ella el valor-F). Por último, es importante señalar que los resultados por usuario son pocos como para afirmar alguna diferencia notable del desempeño de la extracción de requerimientos a partir del tipo de documento a procesar (tesis, reporte de servicio social, de actividad docente o de apoyo a la divulgación).

4.2.2.2 Utilidad percibida por los usuarios

Durante las pruebas de la segunda iteración del desarrollo, se recolectaron más de 6 horas de video; de las cuales, 4.3 horas corresponden a interacciones de los usuarios y el resto a la grabación de rostro (de los usuarios que lo hayan consentido) para el análisis de aspectos de usabilidad. Además, se obtuvieron las respuestas completas al cuestionario de entrada y la entrevista de salida de los cinco usuarios.

A partir de las respuestas al cuestionario de entrada, se construyó un perfil para cada usuario. Este perfil se usa como base para analizar los comentarios y observaciones que hizo el usuario durante la sesión o durante la entrevista de salida. Además, se utiliza junto con heurísticas para la experiencia del usuario para analizar las interacciones y reacciones del usuario mientras utilizó la herramienta. Las respuestas a la entrevista de salida de cada usuario, fueron acumuladas en su dimensión correspondiente; de acuerdo a la estructura del cuestionario construido basado en SUMI en que se basa la entrevista.

La escala “En desacuerdo”, “Indeciso” y “De acuerdo”, fue mapeada en el rango entero [-1, 0, 1]. Dependiendo de si la pregunta inquiere un aspecto positivo (por ejemplo “2 – Recomendaría Requex a mis compañeros de equipo”) o negativo (5 – Aprender a usar Requex, al principio, presenta muchos problemas), “En desacuerdo” y “De acuerdo” pueden ser mapeados en (-1,1) o (1,-1), “Indeciso” siempre es cero.

Con esta información se estima el grado general de usabilidad que percibieron los usuarios de la herramienta. La gráfica en la figura 9 muestra el grado de usabilidad percibido por los usuarios respecto a cada dimensión considerada. Se aprovechó la baja cantidad de usuarios con los que se realizaron sesiones de prueba para exhibir la percepción individual dentro de cada dimensión.

Los resultados se encuentran normalizados en la escala real [-1,1]; dónde -1 representa una percepción totalmente negativa de una dimensión de usabilidad, 0 una percepción neutra y 1 una percepción totalmente positiva. Cada usuario puede aportar hasta |0.2| unidades de percepción a cada dimensión.

Podemos observar que la dimensión con mejor percepción es la de *control*; los usuarios encontraron los menús, formularios y presentación de la interfaz útiles, bien ubicados en pantalla, consistentes en su presentación, funcionamiento y con una retroalimentación adecuada. Estos factores se relacionan y reflejan en los resultados de las dimensiones *facilidad de aprendizaje* y *facilidad de uso*.

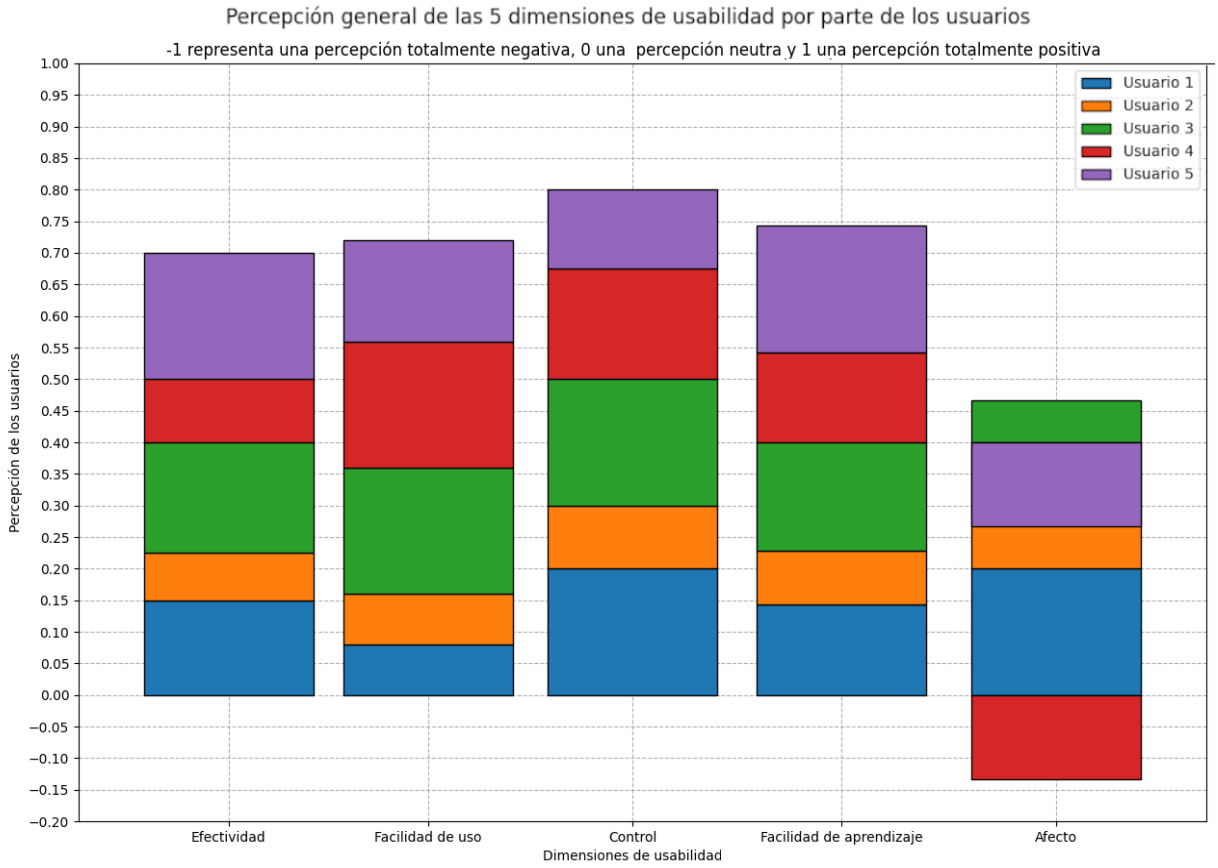

Figura 9: Resultados de las entrevistas de salida por dimensión de usabilidad

Si bien la dimensión de *efectividad* también presenta resultados positivos; las observaciones de los usuarios sugieren que esta dimensión tendría un resultado mejor si la extracción de requerimientos contuviera una porción menor de *Falsos Positivos*. Finalmente, la dimensión de *afecto* tiene el peor resultado; siendo la única en la que un usuario expresó una percepción general negativa. Este resultado es esperado puesto que la dimensión de afecto fue la menos prioritaria durante el diseño de Requex; debido a que tiene una estrecha relación con aspectos de Experiencia del Usuario, los cuales se consideran fuera del alcance de este proyecto.

Al analizar el perfil de los usuarios junto con sus observaciones, interacción con la herramienta y estos resultados de usabilidad, se aprecia que la afinidad de los usuarios con la aplicación está sujeta a su familiaridad con la Ingeniería de Software. Pese a esto; la forma en que los usuarios aprovecharon y criticaron las funciones de ayuda, indica que estas funciones ofrecen un medio para uniformar la utilidad que percibe cada usuario.

En general los usuarios se muestran y dicen muy satisfechos con el documento que resulta de usar la herramienta; además, también exhibieron una actitud reflexiva durante el ejercicio de análisis de requerimientos en las sesiones de prueba, lo que sugiere que se alcanzó el propósito de ofrecer un espacio para reflexionar los requerimientos y el diseño del proyecto descrito en los documentos de entrada.

Al final de las entrevistas, se le preguntó a los usuarios la importancia del prototipo en su vida académica; en una escala del 1 al 5 donde 1 representa que no es nada importante y 5 una total importancia, de donde se obtuvo una importancia promedio de 4.2. Puede consultar un resumen general de los resultados de las pruebas con usuario de la segunda iteración en el anexo D.

En general; los resultados de la segunda ronda de pruebas de usabilidad, sugieren que el prototipo cumple con su objetivo de presentar una prueba de concepto de una herramienta que asiste a recuperar y analizar requerimientos en

documentos de titulación, de manera que los usuarios; aún cuando no son expertos en diseño de software o en lS, pueden entender la actividad que la herramienta le invita a realizar, su propósito y el valor de obtener el resultado.

4.3 Resumen

En este capítulo, se presenta el diseño, ejecución y resultados de los experimentos con los que se busca responder a la hipótesis del proyecto. Los experimentos se diseñaron como pruebas de usabilidad que se realizaron al final de cada iteración de desarrollo; evaluando el desempeño de la extracción de requerimientos y de la usabilidad del prototipo. Estas pruebas se conforman de:

- La síntesis, representación e interpretación de tres índices de extracción de información: precisión, exhaustividad y valor-F. Se utilizan para estimar el desempeño de la extracción de requerimientos.
- Las pruebas de usabilidad incluyen la construcción de un perfil de usuario que se usa para identificar usuarios de prueba, la interacción guiada y observada del prototipo por parte de los usuarios de prueba, y una entrevista de salida que permite conocer la opinión de los usuarios respecto a características específicas de la aplicación.

Los índices se obtienen y representan de la siguiente manera:

1. Se contabilizan manualmente los Verdaderos Positivos, Verdaderos Negativos, Falsos Positivos y Falsos Negativos resultantes durante cada sesión de prueba con usuario de la segunda iteración de desarrollo.
2. Se calculan los tres índices a partir de estos cuatro parámetros; para cada resultado de la extracción en cada sesión de prueba con usuario.
3. Se agrupan los índices del total de oraciones procesadas durante la segunda ronda de pruebas. Estos grupos se obtienen con el algoritmo K-medias y la similitud entre los vectores que representan las oraciones.

Por su parte, cada ronda de pruebas de usabilidad tiene el siguiente propósito:

1. Las pruebas de la primer iteración de desarrollo buscan identificar defectos de usabilidad, los cuáles son priorizados como parte de la planificación de la segunda iteración de desarrollo.
2. Los resultados de las pruebas de la segunda iteración, se analizan con respecto a los índices de la calidad de la extracción de requerimientos para determinar la utilidad percibida de la herramienta; comparado a la calidad de los requerimientos extraídos automáticamente y responder a la hipótesis del proyecto.

Las pruebas de usabilidad se basan en procesos de desarrollo centrado en el usuario y el modelo AMITUDE [37]. En el caso de las pruebas de la segunda iteración, la entrevista de salida se basa en el instrumento SUMI [110], [114]; el cuál es un cuestionario estandarizado que permite conocer los problemas de usabilidad detectados por el usuario. Los resultados de las pruebas de la segunda ronda, se utilizan para estimar la utilidad que percibe el usuario de utilizar la herramienta en el proyecto descrito por los documentos de entrada que se le solicitan al agendar una sesión de prueba.

Para desarrollar las pruebas, se construyó un protocolo para realizar las sesiones, un cuestionario de entrada que permite conocer el perfil del usuario y un cuestionario que se pregunta oralmente durante la entrevista de salida. Se desarrollaron estos tres documentos para cada ronda de pruebas de las dos iteraciones de desarrollo. Anexo a este trabajo, se presentan los tres documentos correspondientes a la segunda iteración:

- El anexo A contiene el protocolo de las pruebas,
- El anexo B contiene el cuestionario de entrada; y finalmente,

- El anexo C contiene el cuestionario para la entrevista de salida.

Con los resultados de las pruebas de la primer iteración de desarrollo, se identificaron 25 tareas con las que se planificó la segunda iteración; de las cuales, 10 tareas corresponden a defectos de usabilidad descubiertos al analizar los resultados de las pruebas. La versión 2.0.0 de Requex; que incluye estas mejoras, fue probada nuevamente al final de la segunda iteración. Con estas pruebas, se realizó un análisis del desempeño de la extracción de información y un análisis de usabilidad; esta vez enfocado a determinar la utilidad que percibieron los usuarios.

Se encontró que la herramienta propuesta logra extraer una alta proporción de los requerimientos contenidos en los documentos que se le den como entrada; pero estos resultados también incluyen una abrumadora cantidad de *Falsos Positivos*. Por su parte; si bien los usuarios señalaron este problema como uno de los más importantes, esto no impidió que pudieran realizar un análisis de requerimientos. Argumentaban los motivos por los que asignaban valores en los editores de la aplicación, pedían opinión sobre sus cambios y lograron identificar con facilidad resultados útiles de los *Falsos Positivos*.

Si bien el perfil y comportamiento de los usuarios, sugiere que los usuarios con mayor afinidad a la Ingeniería de Software son los que entienden mejor el uso de la herramienta; derivado de la terminología y valores que se solicitan al usuario, las funciones de ayuda ofrecen un medio para uniformar el uso de la herramienta entre sus usuarios. La manera en la que los usuarios utilizaron y criticaron estas funciones, sugiere que la ayuda es una de las funciones más relevantes de una herramienta similar a la propuesta.

Por lo anterior, una herramienta práctica similar a Requex; no solamente debería ofrecer mejores resultados en cuanto a la extracción de requerimientos, sino que también será tan importante que ofrezca una ayuda interactiva, completa, fácil de usar y una experiencia de usuario que mantenga la atención del usuario en la actividad del análisis de requerimientos.

Al final de las sesiones de prueba, los usuarios se mostraron muy satisfechos con el documento que resulta de utilizar la herramienta. Los usuarios pudieron apreciar y entender la importancia de la herramienta propuesta durante sesiones de al rededor de 55 minutos, señalando los beneficios que tendría en proyectos de software tanto académicos como en la industria en los que han participado; los cuáles se alinean con las motivaciones de este proyecto.

5 Conclusiones

Partiendo de la hipótesis (sección 1.3) y los objetivos en torno a ella (sección 1.2), se han aplicado técnicas y conceptos de la minería de repositorios de software (sección 2.2), Especificaciones de Requerimientos de Software (2.3) y usabilidad (2.4); con los que se diseñó un prototipo de herramienta de ISAC para compilar de forma semi-automática requerimientos en documentos de titulación que describen proyectos académicos de software.

La solución presentada; Requex, es una aplicación gráfica de escritorio que extrae requerimientos de software y un glosario con términos relevantes detectados en los textos de entrada. Con los resultados de este proceso de minería de texto, la herramienta genera una ERS; y le ofrece a sus usuarios inspeccionar y modificar los atributos de dicho documento mediante un proceso de análisis asistido de requerimientos. Cuando el usuario esté satisfecho con el estado del documento, puede exportarlo como PDF.

El desempeño del prototipo respecto a los objetivos, se evalúa empleando dos enfoques: métricas para evaluar el desempeño de la extracción de información y pruebas de usabilidad. El desempeño de la extracción de información indica la tasa de éxito que tiene el prototipo al extraer requerimientos de los documentos de entrada (secciones 2.2.3.1 y 4.1.1). Por su parte, las pruebas de usabilidad permiten conocer el grado de utilidad y facilidad de uso que perciben los usuarios de la herramienta; así como identificar un panorama general de la experiencia del usuario (secciones 2.4 y 4.1.2).

5.1 Conclusiones respecto a los objetivos e hipótesis

Es importante empezar por considerar que el número de pruebas ejecutadas es muy pequeño como para ofrecer resultados contundentes; pese a esto, estos resultados brindan un panorama general sobre el desempeño de la solución propuesta. Comencemos por analizar los resultados respecto a los objetivos del proyecto:

- **Objetivo 1** – *Desarrollar un prototipo de herramienta que tome Tesis y Reportes de Trabajo Social para realizar Extracción de Información sobre dichas entradas, buscando extraer Requerimientos de Software*.

  La herramienta desarrollada consume documentos de titulación y realiza una extracción exhaustiva de requerimientos; pero los resultados de la extracción incluyen una abrumadora proporción de *Falsos Positivos*.

- **Objetivo 2 –** *La salida del prototipo deberá consistir en documentación textual en español, la cual debe describir los Requerimientos de Software extraídos durante su ejecución*.

  Los documentos generados por la herramienta son altamente valorados por sus usuarios, pueden leerlos sin mayor dificultad puesto que su contenido se forma a partir de oraciones en español que se consideran requerimientos y que además pueden ser afinadas por el usuario antes de generar el documento. El documento es una Especificación de Requerimientos de Software, cuyo formato y contenido se apega a diversos estándares y recomendaciones para este tipo de documentos.

- **Objetivo 3** – *La salida del prototipo deberá aportar conocimiento útil respecto a los Requerimientos que describe*.

  Los usuarios señalaron unánimemente que la principal ventaja de usar la herramienta, es la fácil recuperación de información relativa al diseño y necesidades que busca atender el proyecto de software que describen los documentos de entrada. Además; el documento en formato estandarizado resultante, mantiene la información recolectada y revisada durante el uso del prototipo de manera persistente; permitiendo un fácil acceso a dicha información.

- **Objetivo 4 –** *El uso del prototipo, debería ayudar a los desarrolladores e interesados a entender los Requerimientos de Software extraídos; sin necesidad de realizar otras tareas intensivas*.

  Los usuarios exhibieron un comportamiento analítico durante las tareas de inspección de resultados en las sesiones de prueba. Indicaban sus razones para editar valores; fundamentados en el proyecto descrito en las entradas, cuando tenían problemas para recordar algún aspecto en específico; lo buscaban entre los resultados de la extracción.

Con lo anterior, se procede a responder la hipótesis: "*Es posible obtener documentación de proyectos académicos de software descritos en Tesis o Reportes de Servicio Social en una etapa de su ciclo de vida posterior a la implementación, utilizando técnicas de Procesamiento de Lenguaje Natural y de Extracción de Información; compilando Especificaciones de Requerimientos de Software que satisfacen de forma parcial estándares para este tipo de documentación, redactados en español y ofreciendo ventajas en los procesos de Software de dichos proyectos*".

Los resultados obtenidos sugieren que sí es posible obtener documentación de proyectos académicos de software, compilando ERS utilizando PLN sobre documentos de titulación. Los usuarios señalaron que la herramienta ofrece ventajas en procesos asociados al mantenimiento, operación e integración de participantes en proyectos de software; puesto que por un lado la herramienta ofrece un espacio para reflexionar los requerimientos y el documento resultante permite identificar claramente la información analizada.

Pese a que el número de resultados es pequeño como para ofrecer una respuesta contundente; el diseño y estado actual de la herramienta deberían ofrecer un desempeño de extracción de requerimientos similar al que se exhibe en los resultados por tópicos de oraciones; por lo que la extracción exhaustiva de requerimientos debería apreciarse con otros documentos (al igual que el exceso de *Falsos Positivos*).

Por otra parte, este reducido número de resultados no permite distinguir si la herramienta es más efectiva sobre un tipo específico de documentos de titulación; los resultados entre las dos tesis presentadas difieren entre sí; así como entre dos tipos de documentos presentados en las pruebas no considerados originalmente (reportes de apoyo a la divulgación y de actividad docente).

De acuerdo a los resultados obtenidos; una herramienta basada en MRS que extrae requerimientos, puede emplearse para recuperar Requerimientos de Software en documentos de titulación que describen en español un proyecto académico de software. Además, la herramienta puede ofrecer un espacio para analizar dichos requerimientos y hacer disponible la información recuperada a través de una ERS que se apega a estándares y recomendaciones en la Ingeniería de Software para este tipo de documentos; que además cuenta con una redacción clara en español.

5.2 Trabajo futuro

De los resultados obtenidos durante la segunda ronda de pruebas, podemos notar que el aspecto que más atención requiere; por ser la causa de los bajos puntajes de los índices exhaustividad y valor-F y de las deficiencias en la efectividad y afecto percibidos, es la abundante cantidad de *Falsos Positivos* entre los resultados. Para esto, podrían seguirse las siguientes estrategias:

1. **Crear modelos de lenguaje adhoc al tipo de documentos que consume la herramienta propuesta**.

   La calidad de la extracción de información depende (en principio) de los modelos del lenguaje que se utilicen, ya que es a partir de estos modelos con los que se realiza la división de los documentos de entrada en oraciones y se obtienen los modelos vectoriales del texto. Los resultados obtenidos se lograron con el modelo

proporcionado por spaCy `es_core_news_md`, el cuál se basa en artículos periodísticos internacionales y entradas de Wikipedia en español; enriquecido con 8 documentos de titulación dirigidos por miembros del ESIE.

Un modelo adhoc, construido con un corpus amplio de Tesis y Reportes de Servicio Social que describan proyectos de software, podría ofrecer una separación de los textos de entrada en oraciones más preciso y modelos vectoriales más cercanos al contexto lingüístico de los documentos. Sin embargo; alcanzar un tamaño de corpus suficientemente representativo de los documentos de entrada, podría verse limitado por la disponibilidad de trabajos de titulación de la naturaleza requerida.

Siguiendo la estrategia que presentan autores como Mahadi et. al. [50] y Omran y Treude [58], al corpus de entrenamiento se podrían adicionar textos obtenidos de foros de desarrollo de software en español (los cuales han proliferado en los últimos años); aprovechando el contenido de foros que utilizan una jerga y bloques de código similares a los que se encuentran en estos documentos de titulación.

Además de ofrecer un corpus de mayor tamaño, esta estrategia también puede resultar en un modelo del lenguaje más general; que podría ser utilizado en documentos de titulación de diversas instituciones de educación superior y no limitarse a las características generales del lenguaje presentes en los documentos dirigidos por integrantes del ESIE.

2. **Migrar las reglas de extracción a un modelo de minería basado en estadística**.

   Los mecanismos de extracción de requerimientos del prototipo se basan en reglas, las cuáles fueron definidas a partir de estudiar la estructura gramatical con la que aparecen en los documentos utilizados en el corpus de entrenamiento. Considerando que la extracción basada en reglas es una estrategia que presenta problemas similares en diversos campos de aplicación; en su lugar se prefiere la extracción basada en estadística [66], [68], [78], [115], ya que es más escalable y parametrizable que un conjunto de reglas que puede volverse difícil de mantener.

   Se optó por basar la extracción en reglas puesto que la solución presentada es una primer exploración a la extracción de requerimientos a partir de documentos de titulación en español. Los modelos de extracción basados en reglas, generalmente pueden diseñarse más rápidamente que un modelo estadístico y permiten identificar a bajo costo si los parámetros en los que se basa la extracción ofrecen resultados positivos.

   En el caso de Requex, los índices precisión, exhaustividad y valor-F sugieren que la estrategia planteada en la sección 3.3.1.2 de usar el patrón gramatical $\left(SUST^{*}+VERB^{*}+ADJ^{*}+ADV^{*}\right)^{+}+P$ palabras clave y perífrasis verbal, permite identificar exhaustivamente la mayoría de los requerimientos presentes en los documentos; pero los resultados suelen ser poco precisos al contener una considerable proporción de *Falsos Positivos*.

   Además, como se observó en los tópicos 7 y 8 (sección 5.1.1), los filtros presentes en Requex son demasiado estrictos. De mantener estas estrategias de extracción como base en un modelo estadístico, las palabras clave podrían incluir un peso que sugiera la probabilidad de que su presencia indique que la oración es un requerimiento candidato. Utilizando modelos vectoriales de las oraciones, podrían combinarse oraciones similares de acuerdo a la distancia entre sus vectores y estimar estadísticamente el umbral aceptable de fragmentos de código y URL para aceptar o rechazar las oraciones como requerimientos.

   Por último; también podrían emplearse Resumen Automático para generar el título de los requerimientos. En su estado actual, se toma la primer ocurrencia del grupo en el patrón gramatical que sugiere un requerimiento; $\left(SUST^{*}+VERB^{*}+ADJ^{*}+ADV^{*}\right)$ y se asigna como el título de los requerimientos. Sin embargo, en la

mayoría de las veces que ocurre el patrón buscado, este grupo aparece una sola vez y por ello la mayoría de los requerimientos extraídos presentan un título y descripción con el mismo valor.

Por otra parte, si bien los usuarios consideran que la presentación de la interfaz gráfica de la aplicación es suficientemente buena, presenta algunos defectos derivados de aspectos como la tipografía empleada, márgenes, distancia entre elementos en pantalla; y otros aspectos en torno a su diseño gráfico. Una herramienta similar a la propuesta debería abarcar estos aspectos para poder elevar la percepción de la dimensión de afecto; mejorar la experiencia del usuario puede mejorar el espacio de análisis que ofrece la herramienta. A su vez, un mejor análisis puede ofrecer ERS de mejor calidad.

La función de ayuda fue identificada por los usuarios como una de las más importantes de la herramienta; y en los resultados obtenidos se observa que la ayuda le permite a usuarios con distintos niveles de experiencia en diseño de software, explotar de manera más uniforme las funciones de análisis de requerimientos. En general, los usuarios expresaron desear que la función de ayuda fuese más extensa y proactiva.

Para esto, podría incorporarse un tutorial dentro de la aplicación; utilizando una serie de diálogos (posiblemente narrados) que describan las funciones de la aplicación, los estándares y teorías de Ingeniería de Software en las que se fundamentan; la barra de estado podría extender o complementar información de ayuda. Se podría incorporar un diccionario con la terminología de IS empleada, teoría y mecanismos en torno al análisis de requerimientos y ciclo de vida del software; en el que los usuarios puedan hacer búsquedas dinámicamente.

Finalmente; podría explorarse la aplicación de una herramienta como la propuesta para analizar documentos generados en la industria. Varios usuarios señalaron interés en aplicar la herramienta en sus espacios de trabajo e indicaron que de poder utilizarla efectivamente, representaría una enorme ventaja sobre la forma en la que se estudia el diseño de software y en la que se transmite esa información.

El desarrollo de este tipo de herramientas ISAC ofrece una posible alternativa frente a los problemas derivados de las deficiencias de la documentación sobre el diseño de productos de software; recuperando información dispersa en distintos productos de trabajo y concentrándola en un formato estandarizado de fácil acceso. La disponibilidad de esta información ofrece la posibilidad de tomar mejores decisiones al heredar las características de un software en otro, al modificarlas para una siguiente versión del mismo software; o aprovecharse para capacitar integrantes nuevos a los equipos de trabajo.

6 Anexo A – Protocolo para las sesiones de la segunda ronda de pruebas de usabilidad de “Requex”

Protocolo para las pruebas de usabilidad de la segunda iteración del desarrollo de “Requex”;

del lunes 21 de junio al viernes 2 de julio del 2021.

6.1 Acuerdo de asistencia y registro por medio del cuestionario de entrada

Los interesados en participar en la sesión de prueba deberán responder previamente el cuestionario de entrada. En él se solicita consentimiento para capturar las interacciones del usuario; y opcionalmente su rostro y/o voz durante la sesión de prueba, además de obtener el perfil como participante y conocer la familiaridad que tiene con herramientas de Ingeniería de Software Asistida por Computadora (ISAC).

También se le pregunta a los usuarios, el uso que les han dado a las herramientas ISAC en los proyectos de software en que hayan participado. Además, se le pregunta a los usuarios la plataforma con que prefieren para utilizar Requex de forma remota; pudiendo elegir entre Zoom y Teamviewer. El cuestionario de entrada se encuentra en la siguiente dirección electrónica: https://forms.gle/AZmK8tdBJL4krFfJ7

En el perfil de usuario se establece si el usuario ha participado o no en algún proyecto de software que esté descrito en un trabajo de titulación; y si tiene acceso a alguno de esos documentos de titulación. Finalmente, deben elegir una fecha y hora para llevar a cabo su sesión por medio de la siguiente liga: https://calendly.com/icas/seg-p-requex

Los documentos de titulación nombrados en el cuestionario serán solicitados por correo (a menos que estos ya hayan sido enviados previamente). Se le indicará al usuario que deberá instalar Zoom (y Teamviewer si el usuario elige esta plataforma) y se le proporcionará la invitación para la sesión por Zoom a la hora acordada; confirmando la sesión.

6.2 Preparación del entorno de prueba

El entorno de prueba consiste de los siguientes elementos:

- Sistema anfitrión basado en arquitectura `x86_64` (`amd64`) con mínimo 8 Gb de memoria, 20Gb de almacenamiento libre y conexión a Internet. El espacio de almacenamiento incluye:
  - 1.4 Gb designado para almacenar el repositorio del proyecto en la versión 1.23.0.
  - El resto para almacenar los registros de la aplicación, los modelos del lenguaje generados durante el entrenamiento de Requex, los documentos de los usuarios, archivos de texto plano con el contenido de los documentos de los usuarios y los vídeos que emanan de las sesiones.
- Sistema operativo `GNU/Debian 10` sobre `Linux 4.19.0-13-amd64` o superior.
- `Python 3.7.3` como plataforma de ejecución; en un entorno virtual (`python-venv`) con todas las dependencias del proyecto.
- `OBS 0.0.1` para grabar la pantalla del sistema anfitrión, capturando las interacciones del usuario con la aplicación.

- `Zoom 5.7.0` para realizar la videollamada con la que se desarrolla la sesión, grabar el rostro y/o voz de los usuarios que lo consientan y herramienta de escritorio compartido.
- `Teamviewer 15.19.3` como herramienta de escritorio compartido. Solo se utiliza si el usuario prefiere esta aplicación sobre Zoom.

Una vez que un usuario complete el registro para participar en una sesión y se le envíe la confirmación con la invitación para realizar la reunión por Zoom; se prepara un directorio en el entorno de prueba para concentrar la información resultante de las sesiones. Para esto, cree un subdirectorio que lleve por nombre el correo con el que se registró el usuario en el cuestionario de entrada.

Este subdirectorio se utilizará para almacenar la grabación de las interacciones del usuario, los registros de ejecución de Requex, el chat de la sesión, la grabación del rostro y/o voz del usuario (de haber consentido dichas grabaciones) y los archivos de entrada que resultan tras realizar la sesión con el usuario.

Las siguientes tareas deben completarse antes de iniciar cada sesión de prueba.

1. Debe extraerse el texto de los documentos proporcionados en archivos de texto plano.
2. Estos archivos de texto deben colocarse en una carpeta en el entorno de prueba con el nombre `test-corpus` (dentro del repositorio del proyecto para facilitar su disponibilidad).
3. Se deben actualizar los modelos del lenguaje de Requex; lo cuál se puede lograr utilizando `pytest`. Los modelos generados a partir del corpus se encontrarán en `res/modelos`.
4. Active el entorno virtual de Python que contiene al proyecto.

   ```
   $ source bin/activate
   ```

5. Asegúrese de tener permisos de ejecución sobre el script `src/requex/__init__.py`. Ejecute el siguiente comando para iniciar Requex:

   ```
   (requex)$ src/requex/__init__.py -n “Proyecto de prueba” -c ./test-corpus
   ```

   Este último paso puede tomar varios minutos; dependiendo principalmente del tamaño total de los archivos de entrada. Por ello, se debe procurar realizar este paso al menos 20 minutos antes de la hora acordada con el usuario.

6.3 Bienvenida y presentación

A la hora acordada con cada usuario, la sesión se lleva a cabo de forma remota por medio de Zoom. Se inicia la reunión y se espera hasta 20 minutos a que se conecte el usuario. En caso de que el usuario no se conecte durante esta espera, se cancela la sesión y se contacta al usuario preguntando si le gustaría re-agendar la sesión de haber disponibilidad.

Cuando el usuario se conecte, se le dará la bienvenida y se le explicará el proceso de prueba:

¡Hola, gracias por asistir a esta prueba! Vamos a utilizar un software en desarrollo llamado Requex; el cuál tiene el propósito de extraer requerimientos de tesis y reportes de servicio social. Junto a los requerimientos, recopila un glosario con los términos más relevantes y los presenta para realizar un análisis de requerimientos asistido y generar una Especificación de Requerimientos de Software que se alinea con estándares internacionales y prácticas de Ingeniería de software.

En esta sesión, vamos a realizar seis metas con ayuda de la aplicación. Estas metas requieren que realicemos algunas de las tareas que se esperan sean comunes en el uso proyectado de Requex. Si alguna indicación no es clara o no encuentra cómo realizar la tarea, puede preguntar en cualquier momento. También le invito a realizar observaciones mientras utiliza la aplicación.

Durante la prueba simularemos un análisis de requerimientos; que nos ayudará a reconocer el estado del proyecto como está descrito en los documentos de entrada. También será importante considerar durante la sesión que tendremos un tiempo limitado de 2 minutos máximo para realizar cada tarea. Si no puede completar alguna de las metas en esos dos minutos, conservaremos los cambios incompletos y continuaremos con la siguiente meta.

Si el usuario consintió en el cuestionario de entrada la grabación de voz y vídeo, es conveniente pedir confirmación de dicho consentimiento. En caso afirmativo, se le pedirá que active su cámara y micrófono en Zoom y se graba la sesión de Zoom a partir de este momento.

Por último, quiero recordarle que es indispensable grabar sus interacciones con la aplicación; es decir, los movimientos del ratón y cualquier texto que ingrese mientras la usa, ¿está de acuerdo? Sus comentarios e interacciones serán muy valiosos para mejorar la aplicación.

6.4 Iniciando la simulación del análisis de requerimientos

1. En caso de que el usuario haya indicado que prefiere utilizar Teamviewer para utilizar Requex de forma remota, se le solicitará al usuario iniciar Teamviewer; y se verifica que Teamviewer se ejecuta en el sistema de prueba.
2. Se inicia OBS para grabar el escritorio y los movimientos del cursor para capturar las interacciones del usuario.
3. En caso de ser necesario, se le proporcionan las credenciales de Teamviewer al usuario por medio del chat de Zoom; en otro caso se habilita el control de escritorio remoto. Una vez que el usuario confirme su acceso, se le indica el arranque de la prueba.

   Esta es la aplicación “Requex” que vamos a probar. A partir de este momento comenzaré a indicarle tareas a realizar con esta aplicación; ¿está listo para iniciar?

6.5 Simulación del análisis de requerimientos

El usuario podrá plantear preguntas o hacer observaciones durante la sesión de prueba en el momento que lo considere pertinente. En caso de que la aplicación no haya terminado de extraer requerimientos cuando se ha alcanzado este paso en la sesión, explique al usuario que el proceso de reconocimiento de texto como requerimientos es una tarea intensiva; por lo que es normal que la extracción automática de requerimientos demore algunos minutos.

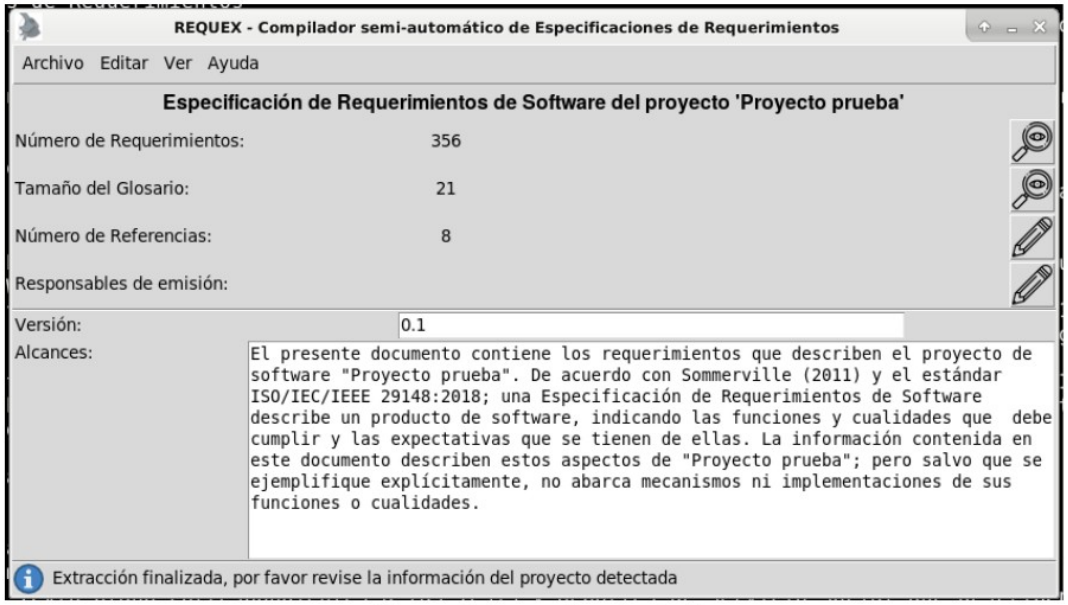

Figura A: Pantalla principal

Inicie la simulación de un proceso de análisis cuando aparezca la pantalla principal (figura A).

Esta es la pantalla principal de la aplicación; los campos que muestra son atributos de la Especificación de Requerimientos de Software representada por la aplicación.

Indique al usuario que cuando aparece esta pantalla (figura A), la aplicación ha terminado la extracción de requerimientos y podemos empezar a analizar la información recuperada.

Antes de iniciar cada meta, pida al usuario describir el contenido de cada pantalla que abra durante la prueba; mientras le presenta las pantallas que puede emplear para alcanzar la meta.

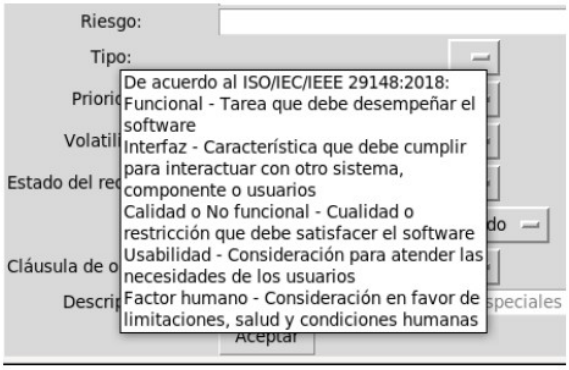

Figura B: Mensaje de ayuda contextual

Indique al usuario que puede obtener descripciones de campos de texto, botones y listas al colocar el cursor sobre los textos a la izquierda de estos elementos (figura B). Finalmente, describa al usuario cómo acceder a las listas de inspección (requerimientos (figura C), glosario (figura D), responsables de emisión (figura E) y referencias (figura F)); de forma que pueda leer la información recuperada y sintetizar información del proyecto.

Cuando el usuario abra una lista de inspección por primera vez, describa el propósito de la pantalla y las funciones que ofrece: presenta todos los miembros de la lista en una pantalla vertical desplazable, permite agregar registros nuevos; así como editar y eliminar los registros existentes.

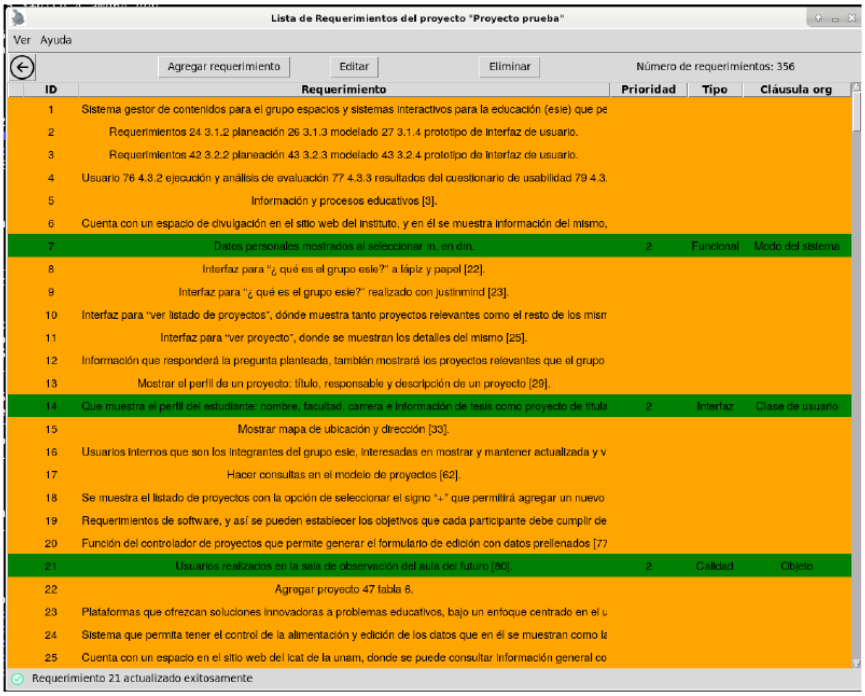

Figura C: Inspector de requerimientos

Figura D: Inspector del glosario

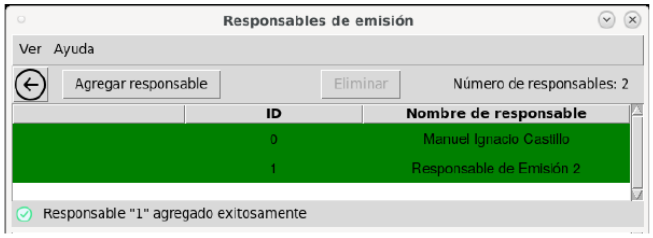

Figura E: Editor de Responsables de emisión

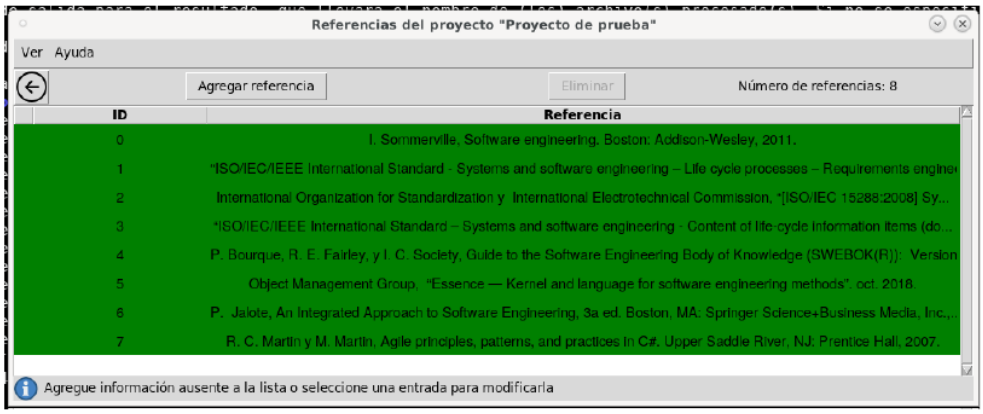

Figura F: Editor de Referencias del proyecto

6.5.1 Meta 1: objetivo del proyecto

La primer meta, consiste en una descripción del objetivo principal del proyecto que se analiza; la cuál debe redactarse como parte de los alcances de la ERS.

6.5.2 Meta 2: atributos generales de la ERS

Solicite al usuario editar los atributos generales de la ERS en la pantalla principal (figura A). Estos atributos son: **versión**, **referencias**, **responsables de emisión** y **nombre del proyecto**.

Recuerde al usuario el propósito y funciones de las cuatro listas de inspección, y que puede apoyarse en ellas para satisfacer la meta.

6.5.3 Meta 3: falsos positivos en requerimientos

Indique al usuario que debe borrar hasta 5 resultados en la lista de requerimientos (figura C) que no correspondan con requerimientos del proyecto. Mencione también al usuario la existencia de la Papelera de Requerimientos (figura G) y cómo acceder a ella desde la lista de requerimientos o desde la pantalla principal.

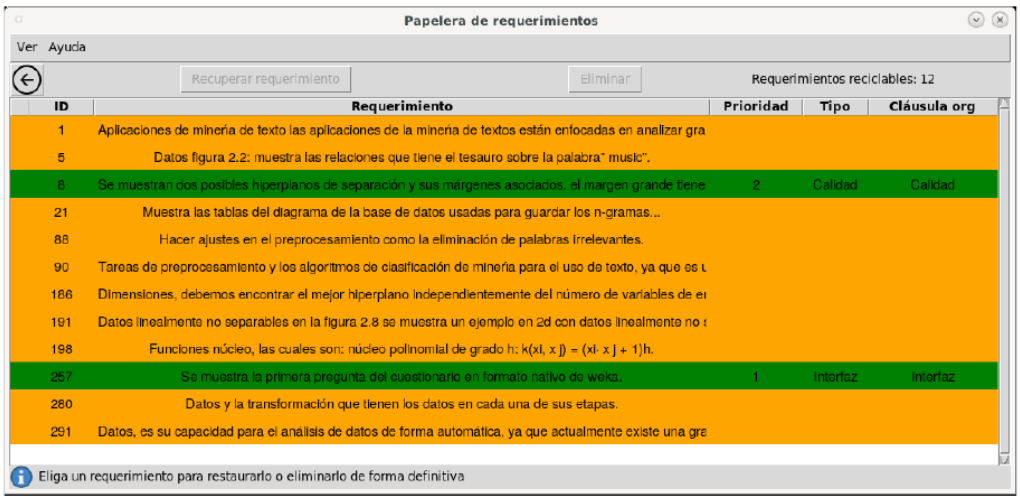

Figura G: Papelera de requerimientos

6.5.4 Meta 4: análisis de requerimientos

Esta meta, requiere que el usuario modifique hasta cinco resultados en la lista de requerimientos (figura C). El tiempo para realizar esta tarea debe considerar las cinco posibles modificaciones en la lista; el usuario tiene 2 minutos para modificar cada requerimiento (incluyendo el tiempo que le tome elegirlo en la lista).

Recuerde al usuario que puede obtener descripciones de campos de texto, botones y listas al colocar el cursor sobre los textos a la izquierda de estos elementos (figura B). Si las descripciones emergentes no son los suficientemente claras, el usuario podrá consultar con el monitor sus dudas.

Las modificaciones a realizar en cada uno de los cinco requerimientos, son las siguientes:

- Ajustar la redacción del título y descripción; si es que el texto extraído automáticamente no refleja un aspecto claro y verificable del software descrito en los documentos de entrada.
- Describir el riesgo de implementar el requerimiento con base en las dificultades asociadas a algún procedimiento para validar el requerimiento.

  Comente con el usuario que de acuerdo con los estándares consultados para crear la aplicación, los requerimientos deben estar asociados a mecanismos para ser validados.
- Elegir una prioridad en una escala del 0 al 10; donde 0 es la mayor prioridad. Dicha prioridad debe reflejar la importancia del requerimiento dentro del proyecto.
- Elegir una volatilidad en una escala del 0 al 10; donde 0 es la menor volatilidad (indicando la mayor estabilidad en la definición del requerimiento). Esta volatilidad debe reflejar la realidad del proyecto como está descrita en los documentos de entrada.
- Elegir un estado para el requerimiento; conforme a las descripciones en los documentos de entrada.
- Elegir un tipo que corresponda a las responsabilidades que atiende el requerimiento.
- Elegir una cláusula de organización conforme a las responsabilidades que atiende el requerimiento.

6.5.5 Meta 5: falsos positivos en el glosario

Indique al usuario que debe borrar hasta 5 resultados en la lista de entradas del glosario (figura D) que no sean términos relevantes en el proyecto. Debe completar esta tarea en un tiempo máximo de 2 minutos.

6.5.6 Meta 6: análisis de entradas del glosario

Solicite al usuario modificar hasta 5 entradas del glosario que si correspondan con términos relevantes para el proyecto. El usuario tendrá 2 minutos para modificar cada entrada del glosario (incluyendo el tiempo que le tome elegirlo de la lista).

Las modificaciones que se espera realice el usuario son las siguientes:

- Modificar el término en la entrada. Los cambios que se piden al usuario dependen del contenido de las entradas: aquellas que poseen un solo término, deben reemplazarlo o agregar un sinónimo; en las entrada con más de un término el usuario debe elegir uno de los términos y eliminar todos los demás.
- Deberá proporcionar una definición para cada entrada del glosario.

6.5.7 Especificación de Requerimientos de Software

El último paso de la sesión es exportar el documento resultante.

Una vez que el usuario haya exportado el documento, solicítele que describa el documento y su relación de su contenido con el análisis realizado durante la sesión. Al final, indíquele revisar las últimas 3 páginas del documento; donde deberá describir la sección de referencias y la tabla de referencias.

A partir de la exportación del archivo, este proceso debe limitarse a 2 minutos.

6.6 Entrevista de salida

Hemos terminado la simulación del análisis de requerimientos. Para finalizar la sesión, voy a decirle algunas afirmaciones respecto a la aplicación que acaba de utilizar. Le pido responder a las afirmaciones utilizando alguna de las siguientes opciones: “En desacuerdo”, “Indeciso” y “De acuerdo”, ¿esta bien?

Se procede a responder el cuestionario de la entrevista de salida; el cuál está disponible en la siguiente URL: https://forms.gle/vZSBg3Mx7wxGrEeH6

6.7 Fin de la sesión

Tras completar en orden los pasos 3 (Sesión de análisis) y 4 (entrevista de salida); termina la sesión de la prueba.

Le agradezco mucho su participación; sus respuestas y acciones serán muy valiosas para mejorar la aplicación. De nuevo le agradezco y le deseo un excelente día.

6.8 Archivo de evidencias

Una vez que se cierre la sesión con el usuario, asegúrese de respaldar los vídeos de las interacciones y del rostro y voz de los usuarios (de estar disponible); así como chats, los registros generados por Requex (`requex/requex.log`) y los archivos de texto utilizados como entrada durante el análisis. Este archivo se coloca en el directorio nombrado con el correo electrónico que el usuario indicó en el cuestionario de entrada descrito en el punto 1 (Preparación del entorno de prueba) de este protocolo.

7 Anexo B – Cuestionario de entrada para la segunda ronda de pruebas de usabilidad de “Requex”

Requex es una aplicación para Windows y Linux que identifica Requerimientos de Software en Tesis y Reportes de Servicio Social, los compila de forma automática y los presenta para ser analizados por el usuario. Requex ofrece un espacio para reflexionar el diseño del software descrito por los documentos de entrada. Al finalizar el análisis, Requex permite exportar los resultados en un documento que corresponde a la Especificación de Requerimientos de Software cuyo formato y contenido se apega a estándares internacionales y prácticas de Ingeniería de Software para este tipo de documentos.

El presente cuestionario de entrada para la segunda prueba de usabilidad de Requex, tiene el propósito de conocer el perfil de los usuarios y establecer las condiciones de la prueba.

Por favor, consulte el consentimiento de manejo de datos para la prueba en: https://bit.ly/3zokQiB
*Obligatorio

1. Correo *

2. Para participar en la prueba, es necesario que consienta el uso de los siguientes datos *

La grabación de interacción consiste en grabar el escritorio de la computadora en la que se ejecuta el software durante la prueba. Los movimientos del cursor, el uso de atajos de teclado y otras interacciones serán analizados para determinar qué aspectos de usabilidad necesitan atención; y qué soluciones son las mejores de acuerdo a las interacciones observadas durante la prueba.

Selecciona todos los que correspondan.

- [ ] Consiento la recolección de mi nombre y correo electrónico con fines de identificación y contacto
- [ ] Consiento la grabación de mi interacción con el software durante la prueba
- [ ] Consiento el registro de opiniones, sentimientos y respuestas emitidas durante la sesión de prueba

3. Opcionalmente, puede consentir el registro de los siguientes datos.

El alcance de la prueba permite prescindir de estos datos, pero aún así resultan útiles para realizar un análisis refinado que puede mejorar los resultados del análisis.

Selecciona todos los que correspondan.

- [ ] Consiento la grabación de voz durante la sesión de prueba
- [ ] Consiento la grabación de mi rostro en video durante la sesión de prueba

4. Plataforma de escritorio compartido *

Debe indicar alguna de las siguientes plataformas para usar la aplicación a probar "Requex" de forma remota. La opciór preferida es Zoom; sin embargo, si lo prefiere la prueba puede realizarse utilizando Teamviewer. Consulte las capacidades de escritorio compartido de cada plataforma en los siguientes enlaces: ZOOM https://support.zoom.us/hc/es/articles/208072316-Sesi%C3%B3n-de-soporte-remoto TEAMVIEWER https://www.teamviewer.com/es-mx/soluciones/escritorio-remoto/

Marca solo un óvalo.

- ◯ Zoom (Requerido para realizar la sesión de prueba, puede requerir configuración adicional)
- ◯ Teamvier (Opcional si prefiere utilizar este control remoto, aún requiere Zoom para la sesión)

Perfil de usuario

5. ¿Ha participado en proyectos académicos que involucren el desarrollo o mantenimiento de un software? *

Marca solo un óvalo.

- ◯ Sí
- ◯ No

6. ¿Cuál de los siguientes roles se apega más a las actividades que ha desempeñado en dichos proyectos? *

Selecciona todos los que correspondan.

- [ ] Programador o Codificador
- [ ] Diseño de la interfaz o interacción con el usuario
- [ ] Diseño de software o ingeniería de procesos
- [ ] Diseño de base de datos o ingeniería de información
- [ ] Control de calidad
- [ ] Consultor o experto
- [ ] Project Manager o Dueño del Producto
- [ ] Cliente o interesado (stakeholder)
- [ ] Otro: _____

7. ¿Conoce algún documento de titulación que describa con atención el software de alguno de estos proyectos? *

¿Conoce un trabajo de titulación que describa algún software en cuyo desarrollo o mantenimiento haya participado? Por favor, seleccione todos los tipos de trabajos que apliquen.

Selecciona todos los que correspondan.

- [ ] Tesis
- [ ] Reporte de Servicio Social
- [ ] Trabajo de divulgación, apoyo a la docencia o a la investigación
- [ ] Trabajo profesional
- [ ] Ninguno
- [ ] Otro: _____

8. Indique el título de al menos uno de los documentos de titulación (de entre los anteriores) que conoce

_____

_____

_____

_____

9. ¿Es autor de alguno de esos documentos? *

Marca solo un óvalo.

- Sí
- No

10. ¿Cuantos años tiene participando en proyectos de software? *

Proyectos de software en general, no necesariamente académicos en este caso.

Marca solo un óvalo.

- Menos de 2 años
- Entre 2 y 5 años
- Más de 5 años

Ingeniería de Software Asistida por Computadora

Las herramientas ISAC (o en inglés Computer-Aided Software Engineering) son aplicaciones que ofrecen algún tipo de ayuda o beneficio para realizar procesos de software. Pueden ayudar a asignar tareas o tickets, a programar o lanzar el producto; o bien, a diseñar y actualizar repositorios con información del diseño.

11. ¿Está familiarizado con alguna herramienta de Ingeniería de Software Asistida por Computadora? *

Marca solo un óvalo.

- Sí
- No

12. Aproximadamente, ¿qué proporción de los proyectos de software en los que ha participad incorporan herramientas ISAC?

Marca solo un óvalo.

- ◯ Muy pocos o ninguno
- ◯ La minoría
- ◯ Aproximadamente en la mitad de ellos
- ◯ La mayoría
- ◯ Todos o casi todos

13. Por favor indique el tipo de herramientas ISAC que han incorporado los proyectos de software en os que ha participado *

Selecciona todos los que correspondan.

- ☐ IDE
- ☐ Sistemas de seguimiento de incidentes o Bugtracker
- ☐ Editores de diagramas o Generadores de código
- ☐ Calendarios, herramientas de seguimiento de tareas o tableros Kanban
- ☐ Herramientas de control de versiones
- ☐ Ninguno

Otro: ☐ _____

14. Aproximadamente, ¿en qué proporción de los proyectos de software en los que ha participado; los procesos de desarrollo son claros, de conocimiento común, seguidos y verificados? *

¿En los proyectos se establecen, siguen y verifican convenciones para nombrar versiones, integrar cambios, estilo de código, trabajo colaborativo, reporte de avances, evaluación de versiones, etc?

Marca solo un óvalo.

- ◯ Muy pocos o ninguno
- ◯ La minoría
- ◯ Aproximadamente en la mitad de ellos
- ◯ La mayoría
- ◯ Todos o casi todos

15. Aproximadamente, ¿en qué proporción de los proyectos de software en los que ha participado, existe documentación suficiente que permita entender y reflexionar el diseño e implementación de los productos de los proyectos? *

No necesariamente documentación escrita, puede tratarse de código de calidad, diagramas, videos; u otros medios qu permitan conocer con facilidad y detalle el diseño, los motivos del diseño y cómo se refleja en el código que integra el producto del proyecto.

Marca solo un óvalo.

- ◯ Muy pocos o ninguno
- ◯ La minoría
- ◯ Aproximadamente en la mitad de ellos
- ◯ La mayoría
- ◯ Todos o casi todos

8 Anexo C – Entrevista de salida para la segunda ronda de pruebas de usabilidad de “Requex”

Una vez terminada la sesión de prueba, recopile la siguiente información. Las respuestas se a cada reactivo debe ser alguna de las siguientes: (1) “En desacuerdo”, (2) “Indeciso” y (3) “De acuerdo”

Agregue cualquier observación adicional que realice el usuario.

*Obligatorio

1. Correo *

_____

2. Recomendaría Requex a mis compañeros de equipo *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

3. Utilizaría Requex con mi equipo de trabajo *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

4. Las instrucciones y ayuda en la aplicación son útiles *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

5. Aprender a usar Requex, al principio, presenta muchos problemas *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

6. Hay tareas que no se cómo realizar con la aplicación *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

7. Disfruto trabajar con Requex *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

8. Los mensajes de ayuda dados por Requex no son muy útiles *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

9. Tarda demasiado tiempo aprender las funciones de Requex. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

10. Observaciones sobre la efectividad de las funciones de Requex

Facilidad de uso

Preguntas respecto a la facilidad de uso de Requex

11. Creo que la aplicación es inconsistente. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

12. Puedo guiarme por la información que proporciona la aplicación. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

13. Las tareas pueden realizarse de forma directa utilizando Requex. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

14. Usar esta aplicación es frustrante. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

15. La aplicación me ha ayudado a solventar cualquier dificultad que haya tenido al usarla. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

16. Observaciones sobre la facilidad de uso

Control del usuario

Preguntas respecto al control del usuario sobre el flujo y operación de la aplicaicón

17. Es obvio que las necesidades del usuario han sido totalmente consideradas. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

18. Al usar la aplicación me he sentido incómodo. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

19. Me parece adecuada la organización de los menús. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

20. Es difícil aprender a usar funciones nuevas. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

21. Se requieren demasiados pasos para hacer cualquier cosa. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

22. Creo que la aplicación me ha provocado dificultades en algunas ocasiones. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

23. Me resulta fácil hacer que la aplicación realice exactamente lo que pretendo. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

24. Nunca aprenderé a usar todo lo que ofrece Requex. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

25. Observaciones sobre la dimensión de control

Facilidad de aprendizaje

Preguntas respecto a la facilidad de aprender a utilizar Requex

26. El software no ha hecho siempre lo que yo esperaba. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

27. La cantidad o calidad de la ayuda varía a lo largo de la sesión de trabajo. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

28. Es relativamente fácil pasar de una tarea a otra. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

29. Es fácil olvidar como se hacen las cosas con la aplicación. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

30. Requex a veces se comporta de forma incomprensible. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

31. La aplicación es realmente muy difícil de usar. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

32. Es fácil ver que opciones hay en cada etapa. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

33. Observaciones sobre la dimensión de facilidad de aprendizaje

Afecto

Preguntas respecto al afecto del usuario por la aplicación

34. Este software parece trastornar la forma en la que normalmente desempeño mi trabajo

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

35. Trabajar con Requex es mentalmente estimulante. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

36. Nunca aparece la suficiente información en la pantalla cuando se necesita. *

Marca solo un óvalo.

	1	2	3	
En desacuerdo	◯	◯	◯	De acuerdo

37. Observaciones sobre la dimensión de afecto

Generales

Preguntas generales respecto a la aplicación

38. En una escala del 1 al 5; donde 1 es nada relevante y 5 es totalmente importante, ¿que importancia tendría Requex en su vida académica? *

Marca solo un óvalo.

1	2	3	4	5
◯	◯	◯	◯	◯

39. ¿Qué considera lo mejor y lo peor de esta herramienta de software? *

40. ¿Qué cree que se debería cambiar en Requex, y porqué? *

9 Anexo D – Análisis de las sesiones de la segunda ronda de pruebas de usabilidad para “Requex”

Usuario 1, lunes 21 de junio a las 12:00.

Perfil:

Programador con entre 2 y 5 años de experiencia, autor de un Trabajo de divulgación, apoyo a la docencia o a la investigación que involucra un proyecto de software. En todos o casi todos los proyectos en los que ha participado, se han empleado herramientas ISAC como: IDE, SSI, editores de diagramas, herramientas de productividad y de control de versiones.

Aproximadamente en la mayoría de los proyectos en los que ha participado se establecieron y evaluaron convenciones para los procesos del desarrollo; así como también en la mayoría de los proyectos se cuenta con documentación que permite entender y reflexionar el diseño e implementación de los productos de software.

Observaciones de las interacciones:

1. Cree que los alcances de la ERS son extraídos con los requerimientos, cuando en realidad se trata de un valor por defecto; y la aplicación no le ayuda a notarlo.
2. Nota y aprecia que los elementos en las listas de inspección cambian de color para indicar su estado.
3. Los gestos y posturas que hace el usuario al interactuar con la lista de requerimientos sugieren que muestra demasiadas entradas en pantalla y el tamaño de letra es pequeño o se pierde en el fondo de color.
4. Trata de leer el texto completo del título de los requerimientos en el inspector; pero en textos largos es cortado y tiene que abrir el editor para verlo completo.
5. Aprovecha la ayuda provista por los tooltips en el editor de requerimientos.
6. El que esté separada la ayuda para el estado de los requerimientos; y que el segundo estado no tiene etiqueta, causa confusión, no es claro para el usuario que puede ver la ayuda en el espacio vacío debajo de la etiqueta.
7. El mensaje de ayuda de la cláusula de organización no es lo suficientemente claro.
8. El usuario olvida rápidamente el propósito del campo “Riesgo” en el editor de requerimientos.
9. El usuario olvida rápidamente el propósito del campo “Descripción” (de la cláusula de organización) en el editor de requerimientos.
10. El usuario quisiera poder seleccionar múltiples entradas en las listas de inspección.

Observaciones del usuario:

1. El usuario se dice muy satisfecho con el resultado del análisis.
2. Considera que la ayuda necesita mejoras.
3. Quisiera que la retroalimentación fuera más detallada.

Importancia de Requex en la vida académica del usuario: 5

Usuario 2, viernes 25 de junio a las 14:00.

Perfil:

Desarrollador full-stack con más de 5 años de experiencia, autor de una Tesis que describe un proyecto de software. En ninguno o casi ninguno de los proyectos en los que ha participado, se han empleado herramientas ISAC.

Aproximadamente en la mitad de los proyectos en los que ha participado se establecieron y evaluaron convenciones para los procesos del desarrollo; mientras que una minoría de los proyectos cuentan con documentación que permite entender y reflexionar el diseño e implementación de los productos de software.

Observaciones de las interacciones:

1. Los gestos y posturas que hace el usuario al interactuar con la lista de requerimientos sugieren que muestra demasiadas entradas en pantalla y el tamaño de letra es pequeño o se pierde en el fondo de color.
2. El usuario tiene dificultades para encontrar elementos en las listas de inspección.
3. El usuario se muestra muy satisfecho con el resultado del análisis.

Observaciones del usuario:

1. Le gustaría que al iniciar la aplicación, se mostrara un documento de salida ejemplo con el que se expliquen las funciones que ofrece el prototipo.
2. Sugiere incluir selección múltiple en las listas de inspección en forma de *checkboxes*.
3. Sugiere incrementar el tamaño de letra.
4. Sugiere resaltar los elementos interactivos.
5. Quisiera que las listas de inspección inviten proactivamente a modificar su contenido.
6. Si bien le parece que el contenido y navegación de la aplicación son adecuados, le gustaría que se mejorará la presentación de la interfaz gráfica.

Importancia de Requex en la vida académica del usuario: 3

Usuario 3, viernes 25 de junio a las 15:00.

Perfil:

Programador con más de 5 años de experiencia, autor de un Reporte de Trabajo Social que describe un proyecto de software. En la mayoría de los proyectos en los que ha participado, se han empleado herramientas ISAC como: IDE, editores de diagramas, herramientas de productividad y de control de versiones.

Aproximadamente en la mitad de los proyectos en los que ha participado se establecieron y evaluaron convenciones para los procesos del desarrollo; así como también la minoría cuenta con documentación que permite entender y reflexionar el diseño e implementación de los productos de software.

Observaciones de las interacciones:

1. Borra manualmente los mensajes de ayuda en los campos de texto.
2. Aprovecha la ayuda provista por los tooltips en el editor de requerimientos.
3. El usuario tiene dificultades para entender el propósito del campo “Descripción” (de la cláusula de organización) en el editor de requerimientos.

4. El usuario tiene dificultades para el valor esperado para el campo “Descripción” (de la cláusula de organización) en el editor de requerimientos.
5. Los gestos y posturas del usuario mientras interactúa con el campo “Descripción” (de la cláusula de organización) en el editor de requerimientos, sugieren que le es frustrante.
6. El usuario cambia los verbos en el glosario a su forma en infinitivo.

Observaciones del usuario:

1. El usuario pierde de vista fácilmente las opciones en la barra de menús.
2. El usuario se dice muy satisfecho con el resultado del análisis.
3. La nomenclatura y datos solicitados; al ser tomados directamente de estándares de Ingeniería de software, no le son familiares y no encuentra su propósito intuitivo.
4. Si bien le parece que el contenido y navegación de la aplicación son adecuados, le gustaría que se mejorará la presentación de la interfaz gráfica.
5. Encuentra difíciles de leer los mensajes de error.

Importancia de Requex en la vida académica del usuario: 4

Usuario 4, domingo 27 de junio a las 12:00.

Perfil:

Arquitecto de software y diseñador de interfaces con entre 2 y 5 años de experiencia, familiarizado con un proyecto de software descrito en un Trabajo de divulgación, apoyo a la docencia o a la investigación. En la mayoría de los proyectos en los que ha participado se han empleado herramientas ISAC como: IDE, SSI, editores de diagramas, herramientas de productividad y de control de versiones.

Aproximadamente en la minoría de los proyectos en que ha participado se establecen y evalúan convenciones para los procesos del desarrollo; así como también la minoría cuenta con documentación que permite entender y reflexionar el diseño e implementación de los productos de software.

Observaciones de las interacciones:

1. La proporción de falsos positivos en la lista de requerimientos desconcierta y distrae al usuario.
2. El que esté separada la ayuda para el estado de los requerimientos; y que el segundo estado no tiene etiqueta, causa confusión.
3. El usuario se muestra muy satisfecho con el resultado del análisis.

Observaciones del usuario:

1. Quisiera que la ayuda fuera más descriptiva y con ejemplos.
2. Quisiera que la ayuda fuera proactiva y que cada ventana se ejemplificaran las funciones que ofrecen.
3. Considera que la herramienta es muy útil para generar ERS y analizar requerimientos, cumpliendo con recomendaciones y estándares.
4. Considera que los problemas de usabilidad rompen el flujo de las tareas del usuario.

5. Considera que la nomenclatura y datos solicitados; al ser tomados directamente de estándares de Ingeniería de software, pueden no ser familiares y no quedar claros para todos los usuarios.
6. Sugiere reducir el número de entradas que muestran las pantallas de inspección; un número gran de resultados abruma al usuario.
7. Considera que completar los datos de los requerimientos en el inspector, ayuda a reflexionar el diseño y necesidades del proyecto.

Importancia de Requex en la vida académica del usuario: 5

Usuario 5, miércoles 30 de junio a las 16:30.

Perfil:

Programador con más de 5 años de experiencia, autor de una Tesis que describe un proyecto de software. En la mayoría de los proyectos en los que ha participado se han empleado herramientas ISAC como: herramientas de productividad y de control de versiones.

Aproximadamente en la minoría de los proyectos en que ha participado se han establecido y evaluado convenciones para los procesos del desarrollo; así como también la minoría cuenta con documentación que permita entender y reflexionar el diseño e implementación de los productos de software.

Este usuario prefirió no realizar la grabación de rostro durante la sesión.

Observaciones de las interacciones:

1. El usuario suele tener dudas respecto a la nomenclatura y datos solicitados; al ser tomados directamente de estándares de Ingeniería de software, no le son familiares.
2. El usuario tiene dificultades para entender el propósito del campo “Riesgo” en el editor de requerimientos.
3. El usuario tiene dificultades para el valor esperado para el campo “Riesgo” en el editor de requerimientos.
4. El usuario tiene dificultades para entender el propósito del campo “Descripción” (de la cláusula de organización) en el editor de requerimientos.
5. El usuario olvida rápidamente el propósito del campo “Descripción” (de la cláusula de organización) en el editor de requerimientos.
6. El usuario olvida rápidamente el propósito del campo “Riesgo” en el editor de requerimientos.

Observaciones del usuario:

1. Desearía opciones redundantes para utilizar las funciones de la aplicación. En particular, quisiera poder modificar uno o varios registros en las listas de inspección directamente.
2. El usuario se dice muy satisfecho con el resultado del análisis.
3. El usuario considera que los resultados de la extracción incluyen un número considerable de falsos positivos.
4. El usuario considera que mejorar la ayuda es necesario para que la aplicación sea efectiva.

Importancia de Requex en la vida académica del usuario: 4

Observaciones generales de los perfiles

- Todos los usuarios han participado como programadores en proyectos de software.

- Todos los usuarios han participado en tareas relacionadas al diseño de varios aspectos de software; cómo bases de datos, interfaz o interacción con el usuario y diseño de software, pero solo uno ha actuado como Project Manager o Dueño del Producto.
- 2 de los usuarios tienen un nivel de experiencia “intermedio”; los otros 3 cuentan con mayor experiencia.
- Las herramientas ISAC más utilizadas son las herramientas de productividad y de control de versiones (4/5); las menos utilizadas son los SSI (2/5).
- Solo uno de los usuarios indicó que en la mayoría de los proyectos en que ha participado se han establecido y evaluado convenciones para los procesos del desarrollo; el resto de los usuarios indicó que esto ha ocurrido en la mitad de los proyectos o menos.
- La mayoría de los usuarios señaló que tan solo la minoría de los proyectos en que ha participado, se cuenta con documentación que permite entender y reflexionar el diseño e implementación de los productos de software.

Análisis general de usabilidad

1. Al utilizar las herramientas de inspección, los gestos y posturas del usuario sugieren que entran en un estado de concentración.
2. Los comentarios que hacen respecto a los valores a asignar en los editores, indican que los usuarios reflexionan estos valores; frecuentemente piden opinión y explican sus motivos para asignar valores (basados en aspectos del proyecto que se analiza).
3. Los usuarios señalaron que la proporción de falsos positivos en los resultados (en especial en la lista de requerimientos), es uno de los principales defectos que presenta el prototipo.
4. Algunos usuarios tienen problema para leer el contenido de las pantallas; lo atribuyen a un tamaño reducido de letra y poco espacio entre elementos en las listas de inspección (3/5).
5. Pese a que los usuarios aprovechan la ayuda contextual; ninguno consultó la ayuda general. La ayuda contextual es más accesible que la general, pues aparece con colocar el cursor sobre un elemento en pantalla; la ayuda general consiste en ventanas con texto. Los usuarios quisieran una ayuda más extensa y proactiva; identifican esta función como una de las más importantes (3/5).
6. Los campos en torno a la “cláusula de organización” de los requerimientos no son claros y frustran a los usuarios (3/5).
7. El Riesgo de los requerimientos también suele ser problemático y frustrante para los usuarios (2/5).
8. Algunos usuarios quisieran modificar simultáneamente múltiples registros en las listas de inspección (2/5).
9. Los usuarios menos familiarizados con terminología de la IS tienen dificultades para entender algunos valores solicitados (3/5).

Los usuarios más afines a la aplicación son aquellos con experiencia “intermedia” o superior en roles en torno al diseño de un producto de software. La forma en que los usuarios aprovecharon las funciones de ayuda; y que hayan señalado que les gustaría que fuese más extensa y proactiva, sugiere que los usuarios con menos experiencia en torno al diseño de software, son capaces de entender el propósito e importancia de editar los valores que se le solicitan durante la sesión de análisis. Se observa que la utilidad de la herramienta está sujeta al dominio de la IS de cada usuario; las funciones de ayuda ofrecen un medio para uniformar la utilidad que percibe cada usuario.

En general los usuarios se muestran y dicen muy satisfechos con el documento que resulta de usar la herramienta; además, la actitud reflexiva durante el uso de la aplicación sugiere que cumple con el propósito de ofrecer un espacio para reflexionar los requerimientos y el diseño del proyecto descrito en los documentos de entrada. La proporción de falsos positivos en los resultados y los problemas de usabilidad (principalmente en el editor de requerimientos), son los factores principales que restringen la efectividad de la herramienta.

El promedio de la importancia del prototipo en la vida académica de lo usuarios; en una escala del 1 al 5 donde 1 representa que no es nada importante y 5 una total importancia, es de 4.2. Este resultado indica que los usuarios perciben la herramienta como útil y relevante en los proyectos académicos de software que han participado.

Defectos de usabilidad en la implementación de los requerimientos

Requerimientos impactados por los defectos de usabilidad descubiertos:

- 2 – Extracción de información (prioridad 0).
- 11 – Asistencia para el análisis de requerimientos (prioridad 6).
- 15 – Edición de los atributos generales de la salida (prioridad 6).
- 17 – Edición de los atributos específicos de cada requerimiento (prioridad 0).
- 18 – Guías en las restricciones (prioridad 2).
- 22 – Asistencia interactiva (prioridad 4).
- 23 – Interfaz gráfica de usuario (prioridad 5).

Todos estos requerimientos fueron impactados por problemas de usabilidad descubiertos en las pruebas de la primer iteración; pero en este caso, se identificaron problemas en únicamente 7 requerimientos; mientras que la ronda anterior identificaron problemas en 15 (de un total de 25). Además, el número de requerimientos de alta prioridad (menor a 3) con problemas de usabilidad detectados; se redujo de 9 a 3 (conforme a la priorización establecida al inicio de la segunda iteración).

A partir de las prioridades de los requerimientos impactados, el número de usuarios que detectaron el defecto y su severidad; es posible determinar tareas para una siguiente etapa de desarrollo que además de corregir defectos de programación identificados a lo largo de la segunda iteración de desarrollo, incluir correcciones para estos defectos de usabilidad y con ellos mejorar la percepción de la herramienta por parte de los usuarios.

10 Referencias

[1] Software Engineering Institute, “CMMI for development”. Carnegie Mellon University, 2010.

[2] P. Bourque, R. E. Fairley, y I. C. Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd ed. Washington, DC, USA: IEEE Computer Society Press, 2014.

[3] Object Management Group, “Essence — Kernel and language for software engineering methods”. oct. 2018.

[4] “ISO/IEC/IEEE International Standard – Systems and software engineering - Content of life-cycle information items (documentation)”, *ISOIECIEEE 152892019E*, pp. 1–86, 2019.

[5] International Organization for Standardization y International Electrotechnical Commission, “ISO/IEC 12207:2008 Systems and software engineering — Software life cycle processes”. ene. 31, 2008. [En línea]. Disponible en: https://www.iso.org/standard/43447.html

[6] G. Booch, “The History of Software Engineering”, *IEEE Softw.*, vol. 35, núm. 5, pp. 108–114, 2018.

[7] P. Naur y B. Randell, Eds., “Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division”. ene. 1969.

[8] D. A. Tamburri *et al.*, *Success and Failure in Software Engineering: a Followup Systematic Literature Review*. 2020.

[9] M. R. Basirati, M. Otasevic, K. Rajavi, M. Böhm, y H. Krcmar, “Understanding the relationship of conflict and success in software development projects”, *Inf. Softw. Technol.*, vol. 126, p. 106331, 2020, doi: https://doi.org/10.1016/j.infsof.2020.106331.

[10] C. Tam, E. J. da C. Moura, T. Oliveira, y J. Varajão, “The factors influencing the success of on-going agile software development projects”, *Int. J. Proj. Manag.*, vol. 38, núm. 3, pp. 165–176, 2020, doi: https://doi.org/10.1016/j.ijproman.2020.02.001.

[11] C. W. Butler, L. R. Vijayasarathy, y N. Roberts, “Managing Software Development Projects for Success: Aligning Plan- and Agility-Based Approaches to Project Complexity and Project Dynamism.”, *Proj. Manag. J.*, vol. 51, núm. 3, pp. 262–277, 2020.

[12] G. Jin Xiu, “Measuring Information System Project Success through a Software-Assisted Qualitative Content Analysis.”, *Inf. Technol. Libr.*, vol. 38, núm. 1, pp. 53–70, 2019.

[13] J. L. O. Coscia, C. Mateos, M. Crasso, y A. Zunino, “Refactoring code-first Web Services for early avoiding WSDL anti-patterns: Approach and comprehensive assessment”, *Sci. Comput. Program.*, vol. 89, pp. 374–407, 2014, doi: https://doi.org/10.1016/j.scico.2014.03.015.

[14] I. Sommerville, *Software engineering*. Boston: Addison-Wesley, 2011.

[15] M. Visconti y C. R. Cook, “An overview of industrial software documentation practice”, en *12th International Conference of the Chilean Computer Science Society, 2002. Proceedings.*, 2002, pp. 179–186.

[16] V. S. Chomal y J. R. Saini, “Software Project Documentation - An Essence of Software Development”, *Int. J. Adv. Netw. Appl.*, vol. 6, núm. 6, pp. 2563–2572, 2015.

[17] R. C. Martin y M. Martin, *Agile principles, patterns, and practices in C#*. Upper Saddle River, NJ: Prentice Hall, 2007.

[18] P. Clements *et al.*, *Documenting software architectures: views and beyond*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2011.

[19] R. Castañeda M, G. De la Cruz M, A. L. Eslava C, y J. Ramirez O, “Desarrollo de un sistema interactivo multimedia a través de la intercomunicación de sus componentes: Caso Laboratorio Musical”, Sn. Francisco de Campeche, Campeche; México, 2013, pp. 1–11.

[20] J. M. Mondragón Cruz y F. Lara Rosano, “Coma (colaboración musical aumentada) un sistema para la creación musical en un esquema colaborativo co-localizado, a través de superficies táctiles y objetos tangible.”, Universidad Nacional Autónoma de México, Ciudad de México, 2014. [En línea]. Disponible en: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat02029a&AN=tes.TES01000723377&lang=es&site=eds-live

[21] G. De la Cruz M, R. Castañeda M, A. L. Eslava C, J. Ramirez O, y C. R. Ma. Alvarado Z, “Centro de recursos didácticos de realidad aumentada para la enseñanza de las ciencias experimentales a nivel bachillerato”, presentado en SOMI XXXII Congreso de Instrumentación, Acapulco, Guerrero; México, 2016.

[22] J. R. Rosas Bocanegra y G. de la Cruz Martínez, “Interacción con realidad aumentada utilizando elementos tangibles.”, Universidad Nacional Autónoma de México, Ciudad de México, 2018. [En línea]. Disponible en:

http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat02029a&AN=tes.TES01000775077&lang=es&site=eds-live
[23] R. Castañeda M, A. L. Eslava C, J. Ramirez O, J. M. García C, y G. De la Cruz M, "Diseño de un Ambiente de Aprendizaje de Realidad Aumentada para la Creación de Rallys Didácticos", presentado en SOMI XXXII Congreso de Instrumentación, Acapulco, Guerrero; México, 2016.
[24] O. Ruiz Gutiérrez y G. de la Cruz Martínez, "Modelo computacional para calcular enlaces químicos utilizando realidad aumentada.", Universidad Nacional Autónoma de México, Ciudad de México, 2018. [En línea]. Disponible en: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat02029a&AN=tes.TES01000780915&lang=es&site=eds-live
[25] J. Ramirez O *et al.*, "Software Interactivo Didáctico como apoyo para la comprensión de lectura en nivel básico. 'El mundo del Bla-Bla-Bla'", presentado en IX Encuentro Internacional Virtual Educa Zaragoza 2008, Zaragoza España, 2008.
[26] M. I. Castillo López y G. de la Cruz Martínez, "Diseño de un videojuego para el desarrollo de habilidades lectoras, basado en el enfoque del diseño de la experiencia del usuario", Universidad Nacional Autónoma de México, Ciudad de México, 2019. [En línea]. Disponible en: http://132.248.9.195/ptd2019/enero/0784830/Index.html
[27] F. de la Cruz, "Universitarios desarrollan vehículo para distribución de insumos médicos | Comisión UNAM COVID-19", *Comisión universitaria para la atención de la emergencia Coronavirus*, jul. 16, 2020. https://covid19comisionunam.unamglobal.com/?p=87484 (consultado sep. 11, 2020).
[28] J. Solórzano Herrera, "Evalúan expertos escenarios de riesgo en ductos petroleros", *Ciencia desde la UAM*, vol. 2, pp. 12–13, jul. 2008.
[29] Instituto Politécnico Nacional, "Estudiante del IPN desarrolla sistema para detectar fallas cardiacas", Ciudad de México, Comunicado 145, ago. 2020. Consultado: sep. 11, 2020. [En línea]. Disponible en: https://www.ipn.mx/assets/files/ccs/docs/comunicados/2020/08/c-145.pdf
[30] Universidad Nacional Autónoma de México, "Convocatoria sublínea Desarrollo de Software", 2020. http://becas.colaboracion-vinculacion.tic.unam.mx/servicio.html (consultado sep. 11, 2020).
[31] Universidad Nacional Autónoma de México y Secretaría de Economía, "NMX-I-059/02-NYCE-2005 Tecnología de la Información — Software — Modelos de procesos y evaluación para desarrollo y mantenimiento de software — Parte 2: Requisitos de Procesos (MoProSoft)". ago. 15, 2005.
[32] L. L. Minku, E. Mendes, y B. Turhan, "Data mining for software engineering and humans in the loop", *Prog. Artif. Intell.*, vol. 5, núm. 4, pp. 307–314, nov. 2016, doi: 10.1007/s13748-016-0092-2.
[33] R. Martin C., *UML for Java developers*. Englewood Cliffs, N.J: Prentice Hall, 2002.
[34] S. McConnell, *Code complete*, 2nd ed. Redmond, Wash: Microsoft Press, 2004.
[35] "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering", *ISOIECIEEE 291482018E*, pp. 1–104, 2018.
[36] P. Jalote, *An Integrated Approach to Software Engineering*, 3a ed. Boston, MA: Springer Science+Business Media, Inc., 2005. Consultado: oct. 19, 2020. [En línea]. Disponible en: http://0-dx.doi.org.fama.us.es/10.1007/0-387-28132-0
[37] N. O. Bernsen y L. Dybkjaer, *Multimodal usability*. Springer, 2009. [En línea]. Disponible en: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat02025a&AN=lib.MX001001273120&lang=es&site=eds-live
[38] A.-H. Tan, "Text Mining: The state of the art and the challenges", 1999.
[39] A. Usai, M. Pironti, M. Mital, y C. Aouina Mejri, "Knowledge discovery out of text data: a systematic review via text mining.", *J. Knowl. Manag.*, vol. 22, núm. 7, pp. 1471–1488, 2018.
[40] I. H. Witten, E. Frank, M. A. Hall, y C. J. Pal, "Chapter 1 - What's it all about?", en *Data Mining (Fourth Edition)*, Fourth Edition., I. H. Witten, E. Frank, M. A. Hall, y C. J. Pal, Eds. Morgan Kaufmann, 2017, pp. 3–41. doi: 10.1016/B978-0-12-804291-5.00001-5.
[41] I. H. Witten, E. Frank, M. A. Hall, y C. J. Pal, "Chapter 3 - Output: Knowledge representation", en *Data Mining (Fourth Edition)*, Fourth Edition., I. H. Witten, E. Frank, M. A. Hall, y C. J. Pal, Eds. Morgan Kaufmann, 2017, pp. 67–89. doi: 10.1016/B978-0-12-804291-5.00003-9.
[42] S. A. Salloum, M. Al-Emran, A. A. Monem, y K. Shaalan, "Using Text Mining Techniques for Extracting Information from Research Articles", en *Intelligent Natural Language Processing: Trends and Applications*, K. Shaalan, A. E.

Hassanien, y F. Tolba, Eds. Cham: Springer International Publishing, 2018, pp. 373–397. doi: 10.1007/978-3-319-67056-0_18.
[43] C. Ma, D. C. Chou, y D. C. Yen, "Data warehousing, technology assessment and management.", *Ind. Manag. Data Syst.*, vol. 100, núm. 3/4, p. 125, 2000.
[44] V. A. Luzgin y I. I. Kholod, "Overview of Mining Software Repositories", en *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2020, pp. 400–404.
[45] T. Xie, S. Thummalapenta, D. Lo, y C. Liu, "Data Mining for Software Engineering", *Computer*, vol. 42, núm. 8, pp. 55–62, 2009.
[46] U. Chakravarty, R. Pimpale, R. Sharma, y L. Ramanathan, "Applications of Artificial Intelligence and Data Mining in Optimizing Software Engineering", *Int. J. Syst. Softw. Eng.*, vol. 5, núm. 1, p. 1, ene. 2017.
[47] A. E. Hassan y T. Xie, "Mining software engineering data", en *2010 ACM/IEEE 32nd International Conference on Software Engineering*, may 2010, vol. 2, pp. 503–504. doi: 10.1145/1810295.1810451.
[48] M. Ellmann, "Natural Language Processing (NLP) Applied on Issue Trackers", en *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, New York, NY, USA, 2018, pp. 38–41. doi: 10.1145/3283812.3283825.
[49] P. Ardimento y A. Dinapoli, "Knowledge Extraction from On-Line Open Source Bug Tracking Systems to Predict Bug-Fixing Time", New York, NY, USA, 2017. doi: 10.1145/3102254.3102275.
[50] A. Mahadi, K. Tongay, y N. A. Ernst, "Cross-Dataset Design Discussion Mining", en *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2020, pp. 149–160.
[51] M. Hirsch, A. Rodriguez, J. M. Rodriguez, C. Mateos, y A. Zunino, "Spotting and Removing WSDL Anti-pattern Root Causes in Code-first Web Services Using NLP Techniques: A Thorough Validation of Impact on Service Discoverability", *Comput. Stand. Interfaces*, vol. 56, pp. 116–133, 2018, doi: https://doi.org/10.1016/j.csi.2017.09.010.
[52] W. J. Brown, Ed., *AntiPatterns: refactoring software, architectures, and projects in crisis*. New York: Wiley, 1998.
[53] W3C, "Web Services Glossary", feb. 11, 2004. https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/ (consultado jul. 17, 2020).
[54] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*, 1a ed. USA: Prentice Hall PTR, 2008.
[55] S. Tiwari, D. Ameta, y A. Banerjee, "An Approach to Identify Use Case Scenarios from Textual Requirements Specification", New York, NY, USA, 2019. doi: 10.1145/3299771.3299774.
[56] L. Westfall, "Software Requirements Engineering: What, Why, Who, When and How", *Softw. Qual. Prof.*, vol. 7, núm. 4, sep. 2005.
[57] M. P. Arthur, "Automatic Source Code Documentation using Code Summarization Technique of NLP", *Procedia Comput. Sci.*, vol. 171, pp. 2522–2531, 2020, doi: https://doi.org/10.1016/j.procs.2020.04.273.
[58] F. N. A. A. Omran y C. Treude, "Choosing an NLP Library for Analyzing Software Documentation: A Systematic Literature Review and a Series of Experiments", en *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 187–197.
[59] L. Pollock, K. Vijay-Shanker, E. Hill, G. Sridhara, y D. Shepherd, "Natural Language-Based Software Analyses and Tools for Software Maintenance", en *Software Engineering: International Summer Schools, ISSSE 2009-2011, Salerno, Italy. Revised Tutorial Lectures*, A. De Lucia y F. Ferrucci, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 94–125. doi: 10.1007/978-3-642-36054-1_4.
[60] J. S. Thakur y A. Gupta, "Identifying Domain Elements from Textual Specifications", en *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, New York, NY, USA, 2016, pp. 566–577. doi: 10.1145/2970276.2970323.
[61] C. Li, "Preprocessing Methods and Pipelines of Data Mining: An Overview", 2019, p. 7. Consultado: nov. 02, 2020. [En línea]. Disponible en: https://arxiv.org/abs/1906.08510v1
[62] A. Raj, J. Bosch, H. H. Olsson, y T. J. Wang, "Modelling Data Pipelines", en *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2020, pp. 13–20. doi: 10.1109/SEAA51224.2020.00014.
[63] D. Crystal, *A dictionary of linguistics and phonetics*, 6th ed. Malden, MA ; Oxford: Blackwell Pub, 2008.
[64] S. N. Galicia Haro y A. Gelbukh, *Investigaciones en análisis sintáctico para el español*. México, DF: Inst. Politécnico Nacional, Dirección de Publ, 2007.

[65] W. D. O'Grady, M. Dobrovolsky, y M. Aronoff, *Contemporary Linguistics: An Introduction*. St. Martin's Press, 1997. [En línea]. Disponible en: https://books.google.com.mx/books?id=MUu3QgAACAAJ
[66] C. D. Manning y H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.
[67] B. Srinivasa-Desikan, *Natural Language Processing and Computational Linguistics : A Practical Guide to Text Analysis with Python, Gensim, SpaCy, and Keras*. Packt Publishing, 2018. [En línea]. Disponible en: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1841858&lang=es&site=eds-live
[68] D. Jurafsky y J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, vol. 2. 2008.
[69] S. Bird, E. Klein, y E. Loper, *Natural Language Processing with Python*. 2009.
[70] A. Di Tullio, *Manual de gramática del español: desarrollos teóricos, ejercicios, soluciones*. Buenos Aires: Edicial, 1997.
[71] A. R. Salama y W. Menzel, "Learning Context-Integration in a Dependency Parser for Natural Language", en *Intelligent Natural Language Processing: Trends and Applications*, K. Shaalan, A. E. Hassanien, y F. Tolba, Eds. Cham: Springer International Publishing, 2018, pp. 545–569. doi: 10.1007/978-3-319-67056-0_26.
[72] T. McEnery y A. Hardie, *Corpus linguistics: method, theory and practice*. Cambridge ; New York: Cambridge University Press, 2012.
[73] P. Hanks, "Lexical patterns: from Hornby to Hunston and beyond", Barcelona: Universitat Pompeu Fabra, 2008, pp. 89–129.
[74] A. S. Hornby, H. Wakefield, y E. V. Gatenby, "The advanced learner's dictionary of current English / edited by A. S. Hornby, E. V. Gatenby and H. Wakefield". Oxford University Press London, 1963.
[75] M. Á. Muñoz Lobo, "Patrones de gramática léxica en español: Realidades y posibilidades didácticas", *marcoELE*, vol. 15, pp. 124–143, 2012.
[76] R. Alonso Raya, A. Castañeda Castro, P. Martínez Gila, L. Miquel López, J. Ortega Olivares, y J. P. Ruiz Campillo, *Gramática básica del estudiante de español*, Revisada. Barcelona: Difusión, 2011.
[77] K. Adnan y R. Akbar, "Limitations of information extraction methods and techniques for heterogeneous unstructured big data.", *Int. J. Eng. Bus. Manag.*, vol. 11, p. N.PAG, 2019.
[78] J. Tang, M. Hong, D. Zhang, B. Liang, y J. Li, "Information Extraction: Methodologies and Applications", en *Emerging Technologies of Text Mining: Techniques and Applications*, China, 2008.
[79] I. H. Witten, E. Frank, M. A. Hall, y C. J. Pal, "Chapter 5 - Credibility: Evaluating what's been learned", en *Data Mining (Fourth Edition)*, Fourth Edition., I. H. Witten, E. Frank, M. A. Hall, y C. J. Pal, Eds. Morgan Kaufmann, 2017, pp. 161–203. doi: 10.1016/B978-0-12-804291-5.00005-2.
[80] M. Bravo, A. Montes, y A. Reyes, "Natural Language Processing Techniques for the Extraction of Semantic Information in Web Services", en *2008 Seventh Mexican International Conference on Artificial Intelligence*, 2008, pp. 53–57. doi: 10.1109/MICAI.2008.50.
[81] "IEEE Guide for Software Requirements Specifications", *IEEE Std 830-1984*, pp. 1–26, 1984.
[82] "IEEE Recommended Practice for Software Requirements Specifications", *IEEE Std 830-1998*, pp. 1–40, 1998.
[83] "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering", *ISOIECIEEE 291482011E*, pp. 1–94, 2011.
[84] International Organization for Standardization y International Electrotechnical Commission, "[ISO/IEC 15288:2008] Systems and software engineering — System life cycle processes". feb. 2008. [En línea]. Disponible en: https://www.iso.org/standard/43564.html
[85] D. F. Burrows, "The use of computer-aided software engineering technology in systems and software design", en *Proceedings of the European Design Automation Conference, 1990., EDAC.*, 1990, pp. 434–438. doi: 10.1109/EDAC.1990.136687.
[86] J. Iivari, "Why Are CASE Tools Not Used?.", *Commun. ACM*, vol. 39, núm. 10, pp. 94–103, 1996.
[87] K. Łukasz y G. Krzysztof, "Artificial intelligence for software development — the present and the challenges for the future.", *Biul. Wojsk. Akad. Tech.*, vol. 68, núm. 1, pp. 15–32, 2019.
[88] ExplosionAI GmbH, "Spanish", *spaCy Models Documentation*, 2019. https://spacy.io/models/es (consultado may 18, 2020).
[89] S. Chacon y B. Straub, *Pro Git*, 2a ed. Apress, 2020.

[90] Twitter Engineering, “We’re starting with a set of words we want to move away from using in favor of more inclusive language”, *@TwitterEng*, jul. 02, 2020. https://twitter.com/TwitterEng/status/1278733305190342656 (consultado ene. 05, 2021).
[91] C. Wright, “Making open source more inclusive by eradicating problematic language”, jun. 30, 2020. https://www.redhat.com/en/blog/making-open-source-more-inclusive-eradicating-problematic-language (consultado ene. 05, 2021).
[92] D. Miller, “some language improvements”, *GitHub*, jul. 05, 2020. https://github.com/openbsd/src/commit/5bde2954c180034a27b079acaff46073dc75139b (consultado ene. 05, 2021).
[93] K. Gryp, “MySQL Terminology Updates”, *MySQL High Availability*, jul. 01, 2020. https://mysqlhighavailability.com/mysql-terminology-updates/ (consultado ene. 05, 2021).
[94] L. Torvalds, “Merge tag ‘inclusive-terminology’”, jul. 10, 2020. https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=49decddd39e5f6132ccd7d9fdc3d7c470b0061bb (consultado ene. 05, 2021).
[95] P. Petersen, C. Minden, M. Wenzel, A. Pasic, y S. Cai, “Bias-free communication”, sep. 13, 2910. https://docs.microsoft.com/en-us/style-guide/bias-free-communication (consultado ene. 05, 2021).
[96] Python Software Foundation, “Avoid master/slave terminology”, sep. 07, 2018. https://bugs.python.org/issue34605 (consultado ene. 05, 2021).
[97] The Chromium Projects, “Cleanup of potentially offensive terms in codebase”, jul. 26, 2019. https://bugs.chromium.org/p/chromium/issues/detail?id=981129#c16 (consultado ene. 05, 2021).
[98] ExplosionAI GmbH, *spaCy*. Alemania, 2020. Consultado: ene. 11, 2021. [En línea]. Disponible en: https://spacy.io/
[99] R. Řehůřek, *Gensim*. UK, 2020. Consultado: ene. 11, 2021. [En línea]. Disponible en: https://radimrehurek.com/gensim/
[100] Python Code Quality Authority, *flake8*. 2020. Consultado: ene. 11, 2021. [En línea]. Disponible en: https://gitlab.com/pycqa/flake8
[101] H. Krekel, *pytest*. pytest-dev, 2020. Consultado: ene. 11, 2021. [En línea]. Disponible en: https://docs.pytest.org/en/latest/
[102] PyInstaller Development Team, *PyInstaller*. 2020. Consultado: ene. 11, 2021. [En línea]. Disponible en: https://www.pyinstaller.org/
[103] ExplosionAI GmbH, “Top-level Functions”, *spaCy API Documentation*, 2021. https://spacy.io/api/top-level (consultado ene. 18, 2021).
[104] ExplosionAI GmbH, “Language”, *spaCy API Documentation*, 2021. https://spacy.io/api/language (consultado ene. 18, 2021).
[105] G. E. Sierra Martínez, “Introducción a los Corpus Lingüísticos.”, *Anu. Let.*, vol. 6, núm. 2, pp. 237–242, 2018.
[106] ExplosionAI GmbH, “Token”, *spaCy API Documentation*, 2021. https://spacy.io/api/token (consultado ene. 18, 2021).
[107] R. Řehůřek, “Word2vec embeddings”, *models.word2vec – Word2vec embeddings – gensim*, nov. 04, 2020. https://radimrehurek.com/gensim/models/word2vec.html (consultado feb. 01, 2021).
[108] T. Mikolov, K. Chen, G. Corrado, y J. Dean, “Efficient Estimation of Word Representations in Vector Space”. 2013.
[109] R. Řehůřek, “TF-IDF Model”, nov. 01, 2019. https://radimrehurek.com/gensim_3.8.3/models/tfidfmodel.html (consultado mar. 13, 2021).
[110] E. A. Montenegro Gaviria y L. Vivas Cerón, “Métodos de evaluación de experiencia de usuario (UX)”, Métodos de evaluación de experiencia de usuario (UX)Erika Alejandra Montenegro GaviriaLuisa Vivas CerónUniversidad de San Buenaventura, Santiago de Cali, 2012. [En línea]. Disponible en: http://hdl.handle.net/10819/1335
[111] P. Christen, *Data matching : concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer, 2012. [En línea]. Disponible en: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat02025a&AN=lib.MX001001578943&lang=es&site=eds-live
[112] J. Nielsen, *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994.
[113] A. Drachen, “Behavioral Telemetry in Games User Research”, en *Game User Experience Evaluation*, R. Bernhaupt, Ed. Cham: Springer International Publishing, 2015, pp. 135–165. doi: 10.1007/978-3-319-15985-0_7.

[114] J. Kirakowski, “The Use of Questionnaire Methods for Usability Assessment”, *SUMI Background Reading*, 1994. https://sumi.uxp.ie/about/sumipapp.html (consultado jun. 18, 2021).
[115] R. Grishman, “Information Extraction”, en *The Handbook of Computational Linguistics and Natural Language Processing*, John Wiley & Sons, Ltd, 2010, pp. 515–530. doi: 10.1002/9781444324044.ch18.