ARTÍCULO DE INVESTIGACIÓN / RESEARCH ARTICLE
http://dx.doi.org/10.14482/inde.37.2.1053

Requerimientos de software: prototipado, software heredado y análisis de documentos

Software requirements: prototyping, legacy software, and document analysis

Javier Medina Cruz*
Eliécer Pineda Ballesteros**
Freddy Reynaldo Téllez Acuña***

*Docente Asistente, ECBTI, Universidad Nacional Abierta y a Distancia - UNAD. Magíster en Ingeniería e Informática. javier.medina@unad.edu.co.

**Docente Asociado, ECBTI, Universidad Nacional Abierta y a Distancia - UNAD. Magíster en Informática. eliecer.pineda@unad.edu.co.

***Docente Asistente, ECBTI, Universidad Nacional Abierta y a Distancia - UNAD. Magíster en Potencia Eléctrica. freddy.tellez@unad.edu.co.

Correspondencia: Eliécer Pineda Ballesteros, 3163318839, carrera 27 n.° 40-43, Bucaramanga.

Financiación: Universidad Nacional Abierta y a Distancia

Resumen

Este artículo presenta un rastreo bibliográfico sobre alternativas poco usuales para el levantamiento de requisitos en proyectos de desarrollo de software. La búsqueda bibliográfica parte de un principio fundamental, recabar información sobre métodos de levantamiento de requisitos que no requieran de un alto porcentaje de interacción con los usuarios o clientes. Esta situación obedece a que usualmente los usuarios o clientes no disponen del tiempo requerido, como ocurre cuando se usan métodos tradicionales. La búsqueda de información se realizó teniendo como clave principal la baja interacción con el usuario o cliente, y en este orden de ideas, hubo necesidad de ampliar el horizonte temporal de búsqueda hasta el año 2000 e incluir textos clásicos en el área de estudio. Los resultados de la búsqueda arrojaron luces sobre las técnicas de la Ingeniería de Requisitos, cuya característica principal es la minimizar la interacción con los usuarios, siendo el prototipado, el análisis de documentos y el software heredado las que más se aproximan a la condición de búsqueda.

Palabras clave: análisis de documentos, ingeniería de requisitos, lenguaje natural, prototipos, software heredado.

Abstract

This paper presents a literature review about alternatives for determination of requirements in software development projects. The literature searching begins from a fundamental principle, request information about methods of determination of requirements that do not need a high percentage of interaction with users or customers. This usually happens because users or customers don't have time as it happens when traditional methods are used.

The search for information was made taking into account the low interaction with the user or customer and in this vein, it was needed to expand the search horizon until 2000 and it includes classic texts in the study area. The search results provided insights about the techniques of requirements engineering, whose main characteristic is to minimize the interaction with users, such as, prototyping, document analysis and legacy software which are the closest ones to the search condition.

Keywords: document analysis, legacy software, natural language, prototype, requirements engineering.

Fecha de recepción: 16 de octubre de 2018
Fecha de aceptación: 1 de abril de 2019

1. INTRODUCCIÓN

El poco tiempo, la resistencia al cambio o la falta de interés por parte de los clientes o usuarios de una entidad pueden ser algunos de los factores que impiden que un desarrollador de software identifique adecuadamente las necesidades de una aplicación o realice exitosamente el proceso de levantamiento de requerimientos. En este sentido [1] manifiestan que, en el desarrollo de proyectos de software, el levantamiento de requerimientos es una etapa tan crítica como la misma codificación. Adicionalmente, se puede presentar el problema de haber establecido requisitos que no son viables, en consecuencia, sería necesario replantear los requerimientos en forma iterativa hasta obtener los definitivos; este tipo de situaciones ocurren, como lo plantea [2], si el cliente está equivocado en las necesidades que debe resolver el sistema. Para [3] el cliente, por lo general, intenta plantear un sistema que habitualmente no es claro para él, sin embargo, es necesario que describa los datos, las funciones y el comportamiento del sistema que debe ser implementado. Por tanto, es necesario, por parte de los desarrolladores, mejorar la comunicación con el cliente para establecer los alcances del proyecto.

En concordancia con lo antes expresado, en este artículo se expone un rastreo bibliográfico de algunas técnicas de Ingeniería de Requisitos para llevar a cabo el levantamiento de la información. Las técnicas consideradas en este artículo son: el uso de prototipos [4], [5], [6], el análisis de documentos [7], [8] y el software heredado [9], [10].

2. MÉTODO

La selección de artículos de investigación para su análisis se realizó tomando como referencia el sistema Scopus, Proquest y Google Scholar, contemplando los siguientes criterios de selección:

Fechas de publicación: 2000 en adelante, con excepciones de textos y libros clásicos. El artículo presenta una experiencia relacionada con trabajos de investigación en ingeniería del software, teniendo en cuenta los temas de prototipado, análisis de documentos y software heredado, en razón de que dichas técnicas, a pesar de su utilidad, son poco usadas, consecuencia de su baja visibilidad [11], [12], [13], especialmente en textos de ingeniería del software. Palabras clave utilizadas: software heredado, análisis de documentos, software engineering, ingeniería del software, legacy software, elicitación de requisitos, requirements elicitation, levantamiento de requisitos, survey requirements y document analysis.

Las referencias recuperadas se organizaron por fecha de publicación, para realizar un proceso de selección y posteriormente por relevancia. Con cada referencia que cumplía con los criterios de selección se procedió a realizar la búsqueda del artículo completo en las bases de datos Academic Search Complete (EbscoHost), Journal Sto-

rage, Compendex, Directory of Open Access Journals, Proquest y Google Scholar; en caso de que no se encontrara el artículo a texto completo se excluía la referencia.

Los artículos seleccionados se revisaron completamente y se procedió a sintetizar los aspectos más relevantes de cada uno de los mismos, registrándolos en un resumen analítico del escrito que contenía los siguientes ítems: Palabras de búsqueda, Base de datos, Palabras claves, Referencia, País, Método, Estrategias de levantamiento y elicitación de requisitos, Conclusiones.

3. RESULTADOS

Ingeniería de requisitos

A continuación se presenta una serie de definiciones que se requieren para una mejor comprensión del texto.

Según [14], la Ingeniería de requisitos se define como una aplicación disciplinada de principios científicos y técnicas para desarrollar, comunicar y gestionar requerimientos. Así mismo [15], definen la Ingeniería de requisitos como todas las actividades relacionadas con la identificación y documentación de las necesidades de clientes y usuarios. La creación de un documento que describa la conducta externa y las restricciones asociadas (de un sistema) que podrá satisfacer dichas necesidades. El análisis y validación del documento de requerimientos para asegurar consistencia y viabilidad. La evolución de las necesidades.

La Ingeniería de requisitos es descrita por [16] como el proceso de estudiar las necesidades del usuario para llegar a una definición de requerimientos de sistema, hardware o software. Refinar los requerimientos de un sistema, hardware o software. Descubrir y comunicar las necesidades de clientes y usuarios y la gestión de los cambios en dichas necesidades.

La Ingeniería de requisitos, en palabras de [17], es definida como el proceso de desarrollar una especificación de software o el proceso de establecer qué servicios son requeridos y qué límites hay en la operación y desarrollo del sistema. Con base en lo anterior, la Ingeniería de requisitos (IR) puede considerarse, de manera general, como un proceso compuesto por etapas ordenadas, objetivos definidos y técnicas establecidas. En cuanto a las características deseables en los requisitos, lo principal es que deben ser alcanzables, necesarios, jerárquicos y concretos. De acuerdo con [18], los requisitos deben tener una alta prioridad dentro del proceso de desarrollo de software.

De otra parte, [19] señala que la especificación de requisitos debe ser posible, necesaria, priorizada, concreta y verificable. En este mismo sentido, la [20] complementa lo

anterior al afirmar que dicha especificación debe ser correcta, completa, consistente, comprobable y modificable.

En este orden de ideas y conociendo el contexto general de la Ingeniería de requisitos, se procede con la descripción de algunas de las técnicas para el levantamiento de requerimientos, en las que se procura evitar al máximo la interacción con el cliente o usuario. Según [21], parte de esas técnicas son:

- Prototipado
- Análisis de interfaces
- Storyboards
- Casos de uso
- Análisis de documentos
- Escenarios

Otra técnica de interés es la Ingeniería Inversa (Software Heredado), descrita por [22], [23].

A continuación se presenta el resultado de la revisión sobre las técnicas: Prototipado, Software heredado y análisis de documentos.

Prototipado

Según lo enuncian [24], [25], [26], los prototipos son medios de comunicación entre analistas, clientes o usuarios, que muestran las decisiones tomadas, con el fin de validarlas y permitir la resolución de los problemas de comprensión presentes en la etapa de levantamiento de requisitos. Los prototipos se pueden considerar como pequeñas implementaciones de un sistema de software, que ayudan al diseño y que pueden ser usadas como una técnica de determinación y validación de requerimientos. Para [27] el uso de prototipos es considerado como una de las técnicas más eficientes del levantamiento de requisitos, en tanto que este método es útil cuando existen dudas acerca de los objetivos de una aplicación y cuando las opiniones del usuario son necesarias en una fase temprana del proceso de especificación.

Dentro del prototipado se pueden mencionar algunas técnicas de diseño que permiten visualizar secuencias de ventanas y acciones relacionadas con el paso de una ventana a otra. Estas técnicas de diseño son descritas por [28], [29], [30], quienes las denominan Sketches y Storyboard, respectivamente. La consideración de su aporte al diseño del software es reforzada por [31], quienes prueban que dichas técnicas se utilizan como medio de comunicación entre desarrolladores y clientes/usuarios,

en la etapa de levantamiento de requisitos. Las técnicas Sketches y Storyboard permiten generar modelos o esquemas visuales como esbozos de interfaces gráficas de usuario (GUI).

De acuerdo con lo anterior, y como lo confirman [32], [33], estas técnicas se pueden plantear como una fuente dinámica de realimentación para el levantamiento de requerimientos, en la medida en que aportan un mejoramiento continuo de los modelos de conocimiento para la generación de los productos requeridos.

Uso de técnicas de prototipado en las fases de análisis y diseño

Según [25], en esta técnica los prototipos son usados para la intermediación entre un modelo de negocio y un sistema. Esta técnica facilita la validación previa de los requerimientos hallados y permite transformar un modelo de negocio en diagramas de casos de uso. De acuerdo con lo descrito por [30], la técnica de storyboard incluye las historias de usuario de los casos de uso y las referencias a elementos de software y hardware, tal como la GUI. Para [30], un caso de uso se asemeja al argumento general de una película, en tanto que el storyboard es el equivalente del guion. La técnica de storyboard tiene ventajas adicionales a las de un prototipo regular, debido a la facilidad que ofrece para presentar interfaces, sin generar expectativas erróneas, pues da la impresión de que el sistema ya está construido.

El Storyboarding, según [34], permite optimizar el levantamiento de requerimientos. Esta técnica emplea interfaces basadas en formularios. Las características particulares de los formularios permiten que, a partir de una interacción "transicional" entre páginas, se aclare la funcionalidad, asumiendo que el estado del sistema, en general, se representa en cada cambio de interfaz.

La técnica de Prototipado Desechable de Interfaces Gráficas de Usuario (PGUI) y su aplicabilidad en el levantamiento de requisitos fue propuesta por [33]. Según los autores, esta técnica facilita la evaluación del comportamiento y la descripción de las propiedades de un sistema. La técnica sugiere el uso de un esquema preliminar de la interfaz, que ayude a comprender mejor las necesidades del entorno, pues, por lo general, los usuarios no comunican correctamente los requerimientos. En una instancia final del método se hace necesario una sesión para integrar a usuarios y analistas con el fin de realizar una síntesis de los hechos más relevantes a través de un mapa de requisitos. Posteriormente se realizan otras sesiones donde se han de detallar los requerimientos iniciales de alto nivel. Esta tarea, según [35], se puede hacer con una herramienta de prototipado rápido como UIDE (User Interface Development Environment).

El uso de los prototipos de interfaz es definido por [36] como la base para la obtención de un primer modelo entidad/relación. La obtención de un modelo de datos en la

fase de diseño puede valerse de la interfaz de usuario heredada a manera de prototipo. De igual manera [37], proponen una técnica que combina la comunicación con los usuarios, la referencia a artefactos o construcciones GUI y un texto.

La obtención de requerimientos de usuarios finales se puede realizar a través de una herramienta de software denominada Colector Gráfico de Requisitos [38], cuyo propósito es ayudar a los usuarios a crear una vista de la interfaz para la aplicación requerida.

De acuerdo con lo anterior, la información se transmite de dos formas: la primera a través de una construcción de ventanas y la segunda mediante widgets. Estas formas de comunicación permiten proveer información a través del lenguaje de la GUI. Los usuarios tendrán una manera de expresar sus objetivos, según [38], teniendo en cuenta que la argumentación puede ser asociada con los widgets.

Software heredado

Para [39], el software heredado es aquel cuya tecnología es considerada obsoleta, su estructura está deteriorada, contiene reglas del negocio que no aplican a toda la organización y no se puede reemplazar fácilmente. Dentro de los procesos de desarrollo de software los que más frecuentemente aplican esta técnica son los de mantenimiento, evolución y reingeniería. Este proceso se centra en la comprensión del código fuente, utilizando la técnica denominada ingeniería inversa. Adicional a dicha técnica existen otras: análisis de objetos de interacción, análisis de interacciones, obtención de casos de uso a partir de interacciones y obtención de modelos.

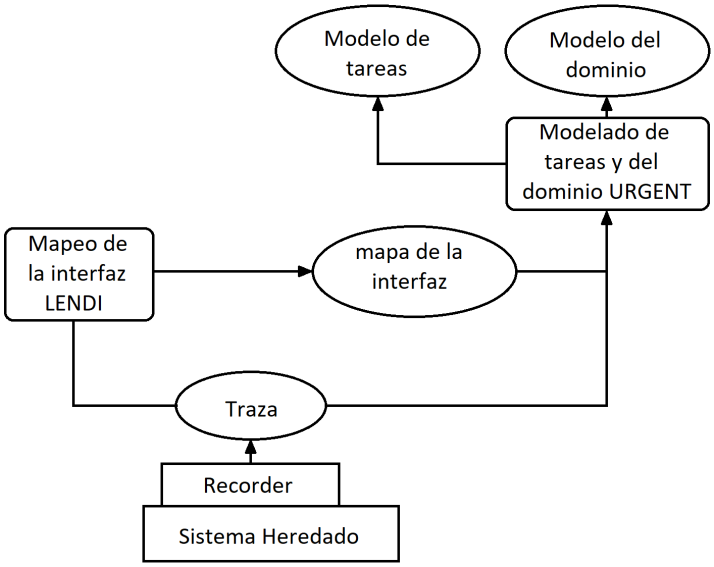
La ingeniería inversa es el proceso de ver un software heredado a un nivel de abstracción alto cuyo principal objetivo es aumentar la comprensión de una aplicación existente, para generar modelos del sistema que representan procesos de mantenimiento, migración o mejora de software. En [20] se recomienda el uso de esta técnica para determinar requerimientos. Se tuvo en cuenta esta recomendación, dado que su propósito no se orienta hacia la duplicación o cambio de software, sino hacia el análisis y la comprensión de este. La ingeniería inversa, según [40], es un proceso que consta de dos pasos ordenados: la extracción y la abstracción de los datos. En el primer paso se hace un análisis de los artefactos de software y en la etapa de abstracción, a alto nivel, se crean documentos y esquemas visuales.

Para [41], la ingeniería inversa tiene como base el análisis de código de las aplicaciones. La disminución de costos, en el análisis de código, se puede reducir si se tiene en cuenta lo expresado por [42], quienes usan elementos gráficos para la obtención de requisitos, permitiendo la comprensión de la arquitectura de software y el análisis de los programas fuente. Hay dos enfoques para realizar el análisis de interfaces: el análisis estático y el análisis dinámico. Con el primero, según [43], se pueden obtener gráficos de ventanas a partir del código fuente y con el segundo, citando a [44], [45],

[46], se ejecuta un programa para gestionar información relacionada con el comportamiento y la arquitectura de un sistema.

El Análisis de Objetos de Interacción permite obtener objetos, clases o tablas que conforman un nuevo sistema. En [47] se presenta un ejemplo que aplica un análisis estático a una interfaz basada en formularios. Así mismo [48] muestran un análisis basado en formularios y determinan escenarios y modelos del Lenguaje Unificado de Modelado (UML). El Análisis de Objetos de Interacción se realiza por fases: en la fase uno se analiza el uso de formularios, se recopila información que corresponda a la estructura del formulario y a la forma de interacción con el usuario. Seguidamente se realiza el modelado conceptual orientado a objetos, y se obtiene un diagrama inicial de clases de la aplicación. En la fase de diseño de escenarios se parte de la fase uno para obtener un diagrama de procesos. Por último, en la fase de integración se llega a un alto nivel de abstracción, a la consolidación y al mejoramiento de los modelos obtenidos previamente.

El Análisis de Interacciones, descrito por [47], corresponde a una técnica que se fundamenta en la ejecución y grabación de la interacción del usuario con las interfaces de comandos. Según [49] el objetivo del análisis de interacciones es interpretar la información y la lógica del proceso involucrada en el software heredado. En el proyecto CelLEST se aplica la técnica de ingeniería inversa de dos formas. La primera consiste en realizar un mapeo de la interfaz donde se presentan las “pantallas” y las “transiciones” entre ellas, y en la segunda se genera un modelo del dominio y sus tareas (ver figura 1). En este proyecto una “pantalla” hace referencia a un objeto individual donde se presenta información útil para los usuarios, y una “transición” se considera como una serie de actividades de los usuarios que pueden involucrar movimientos del cursor o pulsaciones del teclado.



Fuente: [49].

Figura 1. Subestructura del proyecto CelLEST

De otra parte, [46] describen una técnica para el análisis de interacciones denominada "minería Web", la cual apoya el proceso de identificación de clases jerárquicas durante la ejecución de una aplicación. La minería Web consta de cuatro pasos: inicialmente se definen los escenarios de ejecución, posteriormente se abstraen los eventos a partir de la ejecución de un escenario, el paso siguiente es la minería de datos y por último se hace una interpretación de los resultados.

La Obtención de Casos de Uso a partir de Interacciones, propuesta por [46], profundiza en el tema de las técnicas de análisis de interacciones y analiza cómo a partir de trazas de interacción del usuario es posible conseguir los respectivos Casos de Uso. En palabras de [50], los casos de uso se establecen como modelos operacionales de requerimientos funcionales, de una aplicación heredada, a nivel de abstracción del usuario.

La Obtención de Modelos se describe en [46] como una serie de pasos para identificar patrones de actividades, a partir del análisis de trazas de interacción del usuario con el software heredado. Según [21], estos pasos están relacionados con la creación de una versión inicial de un modelo de clases del dominio del problema, a partir de diagramas de secuencia, proceso y de roles. En este orden de ideas, según [51], esta técnica genera el modelo de requisitos representados con diagramas de casos de uso y diagramas de clases, estableciendo una relación entre las clases del modelo de análisis del sistema y las entidades del negocio.

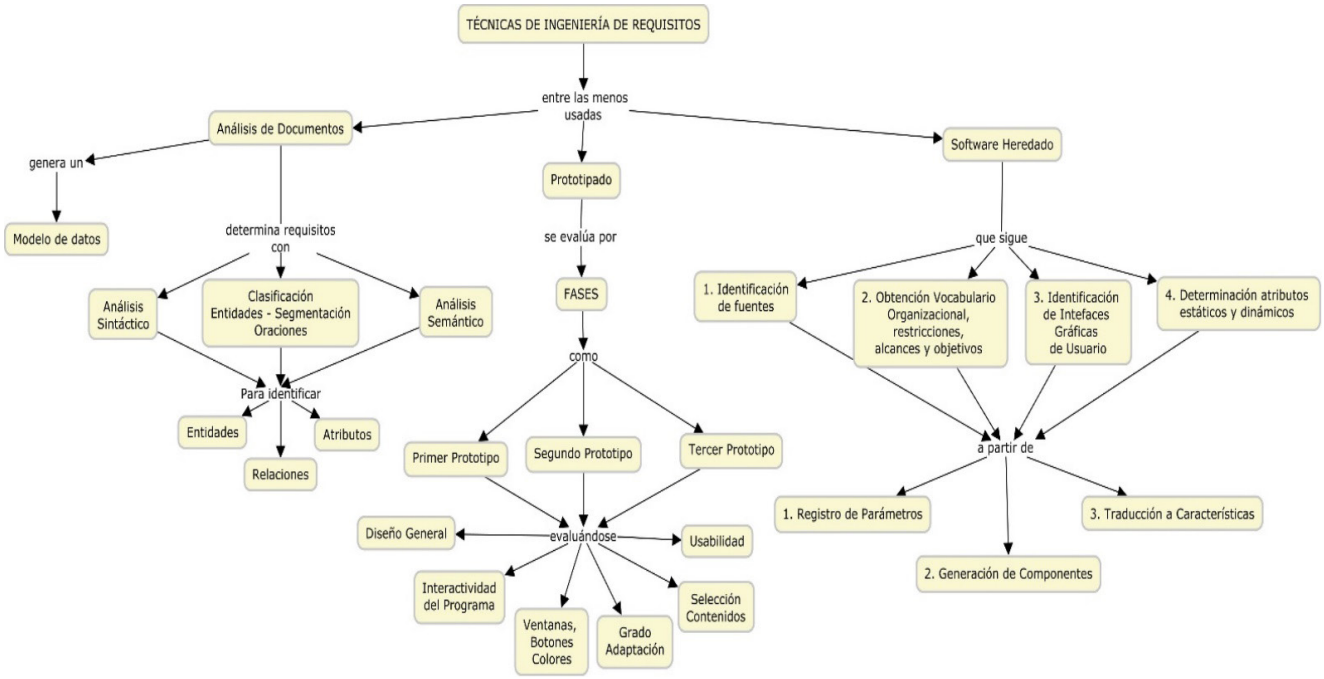
En síntesis y de acuerdo con [21], se pueden recorrer diferentes niveles de abstracción en forma secuencial; por ejemplo, desde una interfaz de usuario hasta un diagrama de clases o modelo conceptual.

El análisis de documentos

Con esta técnica se determinan los requisitos a partir del análisis sintáctico y semántico de los documentos organizacionales, a partir de la observación directa o a través de la aplicación de software especializado que facilita la construcción de frases, en donde dos conceptos se relacionan mediante una acción. La técnica permite identificar las entidades y las relaciones entre estas y sus atributos para la generación de un modelo de datos. El reconocimiento y extracción de los sintagmas de un documento, así como su pre-procesado, hacen parte del análisis de documentos. Este proceso implica la realización de:

- Una segmentación léxica y de oraciones.
- Un análisis y desambiguación morfosintáctica.
- Una detección y clasificación de entidades.
- Un análisis sintáctico.
- Un análisis semántico.
- Una resolución de correferencias.

En un análisis documental se debe tener en cuenta las características cualitativas de los documentos generados; esto permite ampliar la gestión de conocimiento de los procesos en una organización. A continuación, se describen técnicas para la extracción de información de documentos que apoyan la gestión de conocimiento, en el proceso de especificación de requisitos de software.



Fuente: autores (CmapTools).

Figura 2. Técnicas de ingeniería de requisitos de menos uso

En la figura 2 se presenta un esquema que describe las técnicas de ingeniería de requisitos: prototipado, análisis de documentos y software heredado, métodos no convencionales [52], [53], que suelen ser poco aplicados en las fases iniciales de proyectos de desarrollo de software. Estas técnicas se pueden integrar, generando pasos lógicos para el análisis de requerimientos, con el propósito de minimizar el porcentaje de interacción entre el desarrollador y usuarios o clientes, reduciendo así la brecha previamente establecida, en relación con el producto final de software que se requiere. Adicionalmente, es posible combinar las técnicas descritas con otras más tradicionales, aportando al mejoramiento continuo de indicadores de eficiencia, eficacia y efectividad, lo que finalmente redundará en una forma de abordar la denominada "crisis del software" [54].

Técnicas para levantamiento de requisitos a partir de textos en lenguaje natural

En la bibliografía consultada se encontró la descripción de métodos para generar modelos conceptuales, a partir de la abstracción del lenguaje natural.

Existen diversas herramientas que se utilizan para este tipo de abstracciones [43]; los analizadores e intérpretes semánticos y sintácticos, los analizadores léxicos y los

esquemas pre-conceptuales; estos últimos realizan una representación intermedia entre el lenguaje natural y los modelos conceptuales de un módulo de software.

Acorde con [55], [56], la RADD (Rapid Application and Database Development) se considera una "herramienta de diálogo moderado" a partir de una base de datos en lenguaje natural, pues al ingresar un texto en lenguaje natural a un analizador sintáctico, este lo lleva a un nivel de abstracción semántico, identificando el significado de una frase y utilizando roles semánticos asociados con acciones o verbos. El analizador sintáctico identifica los elementos gramaticales de un documento e implementa árboles sintácticos. De otra parte, los roles semánticos asociados con las acciones se utilizan para verificar si una frase, lingüísticamente hablando, está completa (roles semánticos).

Para complementar el proceso anterior [43] propone que, a través de la aplicación de unas reglas heurísticas, se obtengan los elementos que permiten generar un modelo entidad/relación.

Adicionalmente al proyecto RADD, atendiendo lo descrito por [30], existe el proyecto CICO implementado por CIRCLE que usa la inteligencia artificial, específicamente la lógica difusa, para su operación y permite realizar un análisis sintáctico de las frases del documento que están en lenguaje natural. Para [57], cuando CICO es aplicado a la ingeniería de software, apoya las fases de levantamiento y selección de requisitos e identificación de problemas o conflictos entre estos. Este proyecto también está referenciado por [56], quienes describen diagramas que se pueden obtener a partir de especificaciones en lenguaje natural.

Otra herramienta es ASPIN (Automatic Specification Interpreter), implementada por [58], que permite la entrada de modelos definidos y de lenguaje natural. ASPIN realiza la conversión de dichas entradas en grafos conceptuales, mediante un análisis de sintaxis donde se identifican verbos y sustantivos en el texto, filtrando aquellos específicos del dominio, para posteriormente realizar un análisis semántico a partir de la teoría de roles semánticos. Por su parte, NL-OOPS (Natural Language Object – Oriented Product System) está basada en un sistema de procesamiento del Lenguaje Natural llamado LOLITA (Large-scale Object-based Language Interactor Translator and Analyser) [59]; este sistema está conformado por funciones que sirven para el análisis del lenguaje natural. Al nivel de diseño, y de acuerdo con lo expuesto por [60], el NL-OOPS, como salida, entrega las clases candidatas con sus respectivos objetos, atributos y métodos. Esta entrega de resultados se hace en forma de estructura de árbol, teniendo en cuenta los tipos de relaciones, tales como agregaciones, generalizaciones, dependencias, etc. De otra parte, [61] presenta el proyecto KAOS (Knowledge Acquisition in autOmated Specification), en el que se incorporan los requisitos no funcionales, expresados en lenguaje natural y se construye un esquema

conceptual a partir de un lenguaje formal. En [62] se describe cómo el analista debe realizar el proceso de conversión del lenguaje natural al lenguaje formal.

Modelo de espacios vectoriales para levantamiento de requerimientos funcionales a partir del código fuente

La técnica espacios vectoriales, descrita por [63], [64], sugiere realizar consultas a los textos a partir de vectores implementados por términos índice. Los procesos de índices se implementan con base en reglas, en donde los artículos, signos de puntuación, números y otros se obvian, y a cada término índice se le asigna un peso distinto en cada documento y vector de consulta.

El proyecto SMART, descrito por [65], se implementó para apoyar el proceso de gestión de conocimiento de información en dominios problema. Este proyecto establece las bases para la construcción de un modelo de espacio vectorial, a partir de fases de indexación, recuperación y evaluación. A partir de la técnica SMART, con base en el análisis de programas fuente, se referencian situaciones relacionadas con requisitos funcionales donde se utilizan métricas de precisión y recursividad de la ingeniería de requisitos.

Otras técnicas para el análisis de documentos

Se encontraron otras técnicas relacionadas, específicamente, con el análisis de documentos, entre ellas: algoritmos probabilísticos, ponderación automática y algoritmo del vecino más próximo.

La técnica de algoritmos probabilísticos permite determinar la probabilidad de que un término del documento pertenezca a alguna de las categorías de análisis, de acuerdo con ciertos atributos, de los cuales se conoce la probabilidad de que aparezcan en los documentos que pertenecen a la categoría cuestionada. Esta técnica se basa fundamentalmente en el teorema de Bayes.

Entre tanto, la técnica de ponderación automática, planteada por [66], permite expresar en qué medida un documento puede referirse a un tema. Para esto, a los documentos se les asigna un peso entre cero (0) y uno (1), donde el mayor peso lo tienen aquellos documentos cuya ocurrencia de términos sea absolutamente mayor.

Finalmente, la técnica del algoritmo del vecino más próximo permite localizar el documento que más se parezca al elegido para clasificar. Para lo anterior se realiza una consulta sobre una colección de entrenamiento existente categorizada manualmente y donde se conoce la categoría a la que se pertenece y a la que se debe asignar el documento a clasificar.

4. CONCLUSIONES

A través del rastreo bibliográfico se evidenció que es posible, en la etapa de análisis de requisitos dentro del proceso de desarrollo de software, determinar requerimientos sin depender, en un alto porcentaje, de la interacción de un analista o desarrollador con el cliente o el usuario final. Las técnicas revisadas a través de este artículo pueden ser alternativas o complementarias a las técnicas tradicionales de levantamiento de requisitos.

Los materiales bibliográficos revisados permiten afirmar que las técnicas de ingeniería de requisitos descritas no son comunes, por lo que se requiere una profundización, con el fin de apropiar su base conceptual y a partir de allí proponer procedimientos para la consolidación y aplicabilidad de estas técnicas en el proceso de desarrollo de software.

Tomando como referencia cada una de las técnicas descritas es posible formular investigaciones futuras que establezcan sinergias entre ellas para aportar nuevas metodologías para la determinación de requerimientos, teniendo como propósito minimizar el uso de técnicas en las que la interacción desarrollador - cliente/usuario tienen gran relevancia.

Agradecimientos

Los autores expresan su agradecimiento a la Universidad Nacional Abierta y a Distancia (UNAD) por el apoyo brindado durante el desarrollo de la investigación PIE 06-14, titulada "Implementación del Componente e-medios del Nuevo Modelo de e-investigación Unadista".

REFERENCIAS

[1] H. Pérez, C. Salamando & L. Valencia, "Levantamiento de requerimientos basados en el conocimiento del proceso", *Revista Científica*, pp. 42-51, 2012.

[2] A. Martínez, "Calidad en el levantamiento de requerimientos", *Contactos*, pp. 40-42, 2008.

[3] M. d. C. Gómez, *Material didáctico: Notas del curso Análisis de Requerimientos*. Ciudad de México: Publidisa Mexicana S. A. de C.V, 2011.

[4] M. Arias, "La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software", *InterSedes,* pp. 1-13, 2006.

[5] J. Guan, "A Software Prototyping Framework and Methods for Supporting Human's Software Development Activities", in *Bridging the Gaps Between Software Engineering and Human-Computer Interaction,* Portland, Oregon, 2003.

[6] S. Rajendra & A. Dani, "Comparative Study of Prototype Model For Software Engineering With System Development Life Cycle", *IOSR Journal of Engineering*, pp. 21-24, 2012.

[7] J. Suaza & E. Serna, "La importancia de documentar la Elicitación de Requisitos", in *7° Congreso Internacional en Ciencias Computacionales Cicomp*, México, 2014.

[8] A. White, *From Comfort Zone to Performance Management*. W. a. M. publishing, 2009, p. 20.

[9] A. Vizcaíno, J. Soto, F. García, F. Ruiz & M. Piattini, "Aplicando Gestión del Conocimiento en el Proceso de Mantenimiento del Software", *Revista Iberoamericana de Inteligencia Artificial*, pp. 91-98, 2006.

[10] A. Jatain & D. Gaur, "Reengineering Techniques for Object Oriented Legacy Systems", *International Journal of Software Engineering and Its Applications*, pp. 35-52, 2015.

[11] L. Antonelli & A. Oliveros, "Fuentes utilizadas por desarrolladores de software en Argentina para elicitar requerimientos". Available: http://www.inf.puc-rio.br/~wer02/zip/Fuentes_utilizadas(8).pdf.

[12] S. Tiwari, S. S. Rathore & A. Gupta, "Selecting requirement elicitation techniques for software projects", in *CSI Sixth International Conference on Software Engineering (CONSEG),* Indore, India, 2012.

[13] S. Besrour, L. B. Rahim & P. Dominic, "Investigating Requirement Engineering Techniques in The Context of Small and Medium Software Enterprises", in *3rd International Conference On Computer And Information Sciences (ICCOINS),* Kuala Lumpur, 2016.

[14] M. Christel & K. Kang, "Issues in Requirements Elicitation", Carnegie Mellon University, Hanscom, 1992.

[15] P. Hsia, A. Davis & D. Kung, "Status report: requirements engineering", *IEEE Software,* pp. 75-79, 1993.

[16] J. Doe, "Recommended Practice for Software Requirements Specifications (IEEE)", IEEE, New York, 2011.

[17] I. Sommerville, *Software Engineering*. Nueva Jersey: Addison-Wesley, 2012.

[18] J. Quintero & R. Anaya, "MDA y el papel de los modelos en el proceso de desarrollo de software", revista *EIA*, pp. 131-146, 2007.

[19] V. Gervasi, "The CICO domain-based parser", Departamento de Informática, Universidad de Pisa, Pisa, 2001.

[20] IEEE, "Especificaciones de los Requisitos del Software IEEE-STD-830-1998", IEEE, New York, 1998.

[21] T. Blain, "Ten Requirements Gathering Techniques", 21 de noviembre de 2006. Available: http://tynerblain.com/blog/2006/11/21/ten-requirements-gathering-techniques/.

[22] Y. Yu et al., "RETR: Reverse Engineering To Requirements", in *Proceedings of the 12th Working Conference on Reverse Engineering*, Pittsburgh, 2005.

[23] I. Rosziati & T. K. Yong, "ReSeT: Reverse Engineering System Requirements Tool", *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, pp. 1877-1880, 2008.

[24] M. El-Ramly, E. Stroulia & P. Sorenso, "Recovering software requirements from system-user interaction traces", Department of Computing Science - University of Alberta, Alberta, 2002.

[25] W. Lloyd, "Tools and Techniques for Effective Distributed Requirements Engineering: An Empirical Study", Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, 2001.

[26] A. Hickey & A. Davis, "Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes", in *Proceedings of the 36th Hawaii International Conference on System Sciences*, Washington, 2003.

[27] M. Manies & U. Nikual, "La elicitacoión de requisitos en el contexto de un proyecto software", *Ing. USBMed*, pp. 25-29, 2011.

[28] A. Brogneaux et al., "Deriving User-requirements From Human-Computer Interfaces," 2005. Available: https://pure.fundp.ac.be/ws/files/226150/IASTED_DBA_05.pdf.

[29] A. English, "Business modeling with UML: Understanding the similarities and differences between business use cases and system use cases", *IBM Developer Works/ Rational*, 2007.

[30] B. Luedke, "Requirements gathering with storyboards", *IBM developer Works,* 2008.

[31] J. Landay & B. Myers, "Sketching Storyboards to Illustrate Interface Behaviors", in *Conference Companion on Human Factors in Computing System*, New York, 1997.

[32] T. Granollers, "MPIu+a. Una metodología que integra la ingeniería del software, la interacción persona-ordenador y la accesibilidad en el contexto de equipos de desarrollo multidisciplinares", Universitat de Lleida. Departament d'Informàtica i Enginyeria Industrial, Lleida, 2004.

[33] M. Pichler and H. Rumetshofer, "Business Process-based Requirements Modeling and Management", in *First International Workshop on Requirements Engineering Visualization*, Minneapolis, 2006.

[34] C. Fillmore, "The case for Case", in *Universals in Linguistic Theory.* Londres, Bach and Harms, 1977, pp. 59-81.

[35] A. Martínez et al., "From Early Requirements to User Interface Prototyping: A methodological approach", in *17th IEEE International Conference on Automated Software Engineering*, Edinburgh, 2002.

[36] B. Nuseibeh & S. Easterbrook, "Requirements Engineering: A Roadmap", in *The 22nd International Conference on Software Engineering*, Limerick, 2000.

[37] M. Ryan & A. Doubleday, "Evaluating 'Throw Away' Prototyping for Requirements Capture", 2000. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.3791&rep=rep1&type=pdf.

[38] M. Moore & F. Shipman, "A Comparison of Questionnaire-Based and GUI-Based Requirements Gathering", *IEEE Conference Publications*, Grenoble, 2000.

[39] R. Ibrahim & T. Yong, "ReSeT: Reverse Engineering System Requirements Tool", *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, pp. 1877-1880, 2008.

[40] V. Rajlich, "Comprehension and Evolution of Legacy Software", in *(19th) International Conference on,* Boston, 1997.

[41] G. Canfora & M. Di Penta, "New Frontiers of Reverse Engineering", in *Future of Software Engineering, 2007. FOSE '07*, Minneapolis, 2007.

[42] B. Weide, W. Heym & J. Hollingsworth, "Reverse Engineering of Legacy Code Exposed", in *7th International Conference on Software Engineering (ICSE'95),* Ohio, 1995.

[43] M. Moore, "Communicating Requirements Using End-User GUI Constructions with Argumentation", in *18th IEEE International Conference on Automated Software Engineering*, Montreal, 2003.

[44] S. Staiger, "Reverse Engineering of Graphical User Interfaces Using Static Analyses", in *14th Working Conference on Reverse Engineering (WCRE 2007),* Vancouver, 2007.

[45] A. Siochi and R. Ehrich, "Computer analysis of User interfaces Based on Repetition in Transcripts of User Sessions", *ACM Transactions on Information Systems*, pp. 309-335, 1991.

[46] E. Stroulia, M. El-Ramly, L. Kong, P. Sorenson & B. Matichuk, "Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach", in *Reverse Engineering, 1999. Proceedings. Sixth Working Conference on,* Atlanta, 1999.

[47] P. Sorenson, M. El-Ramly & E. Stroulia, "Mining System-User Interaction Traces for Use Case Models", in *10th International Workshop on Program Comprehension,* Paris, 2002.

[48] J. Choobineh, M. Mannino & V. Tseng, "A form-based approach for database analysis and design", *Communications of the ACM,* pp. 108-120, 1992.

[49] H. Lee & C. Yoo, "A form driven object-oriented reverse engineering methodology", *Information System*, pp. 235-259, 2000.

[50] A. Zaidman, T. Calders, S. Demeyer & J. Paredaens, "Applying Webmining Techniques to Execution Traces to Support the Program Comprehension Process", in *Ninth European Conference on Software Maintenance and Reengineering*, Washington, 2005.

[51] J. García et al., "De los Procesos del Negocio a los Casos de Uso", *Técnica Administrativa*, vol. 6, n.° 4, 2007.

[52] S. Khan, A. Dulloo & M. Verma, "Systematic Review of Requirement Elicitation Techniques", *International Journal of Information and Computation Technology*, vol. 4, n.° 2, pp. 133-138, 2014.

[53] S. Tiwari, S. S. Rathore & A. Gupta, "Selecting requirement elicitation techniques for software projects", in *CSI Sixth International Conference on Software Engineering (CONSEG),* Indore, India, 2012.

[54] B. Fitzgerald, "Software Crisis 2.0", *Computer*, vol. 45, n.° 4, pp. 89-91, 2012.

[55] C. Zapata & F. Arango, "Los modelos verbales en lenguaje natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de software: Una revisión crítica", *Revista EAFIT*, pp. 77-95, 2006.

[56] E. Buchholz & A. Düsterhöft, "Using Natural Language for Database Design", in *Proceedings Deutsche Jahrestagung für KI 1994 - Workshop "Reasoning about Structured Objects: Knowledge Representation meets Databases*, 1994.

[57] J. Krebs, "Form feeds function: The role of storyboards in requirements elicitation", *The Rational Edge*, 2005.

[58] W. Cyre, "A requirements sublanguage for automated analysis, International Journal of Intelligent Systems", *Journal of intelligent systems*, pp. 665-689, 1995.

[59] R. Garigliano & L. Mich, "NL-OOPS: A Requirements Analysis tool based on Natural Language Processing", in *3rd International Conference on Data Mining,* Wessex, 2002.

[60] V. Ambriola & V. Gervasi, "On the parallel refinement of NL requirements an UML diagrams", in *2001 Workshop on Trasformations in UML*, Génova, 2001.

[61] L. Mich, "NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA", *Natural Language Engineering*, pp. 161-187, 1996.

[62] A. Dardenne, A. Lamsweerde & S. Fickas, "Goal-directed requirements acquisition", *Science of Computer Programming*, pp. 3-50, 1993.

[63] A. Lamsweerde & E. Letie, "Integrating Obstacles in Goal-Driven Requirements Engineering", in *The 1998 International Conference on Software Engineering*, Kioto, 1998.

[64] G. Salton & M. Lesk, "Computer Evaluation of Indexing and Text Processing", *Journal of the ACM*, pp. 8-36, 1968.

[65] G. Salton, *The SMART Retrieval System - Experiments in Automatic Document Processing*. Upper Saddle River: Prentice-Hall, 1971.

[66] L. Codina, "Teoría de recuperación de información: modelos fundamentales y aplicaciones a la gestión documental", *Revista internacional de Información y Comunicación,* 1995.