NUEVAS REALIDADES PARA LA EDUCACIÓN EN INGENIERÍA: CURRÍCULO, TECNOLOGÍA, MEDIO AMBIENTE Y DESARROLLO
13 - 16 DE SEPTIEMBRE 2022 CARTAGENA DE INDIAS, COLOMBIA
Encuentro Internacional de Educación en Ingeniería ACOFI

Validación y verificación de requisitos de software en proyectos basados en desarrollo continuo usando técnicas de PLN

Adriana Patricia Gómez Ramírez

Universidad Tecnológica de Pereira
Pereira, Colombia

Resumen

El ciclo de vida de un proyecto de software parte de la gestión adecuada de sus requisitos, siendo esta un área de conocimiento que reúne las actividades de levantamiento, análisis, especificación, validación y administración de estos durante todo el ciclo de desarrollo. La especificación básica de requisitos, expresada en lenguaje natural, es transformada a lo largo del ciclo de desarrollo en artefactos que finalmente son desplegados y puestos en funcionamiento como producto.

Un problema identificado en la ingeniería de software es la no correspondencia entre los requisitos funcionales especificados para el producto y las características que son finalmente incorporadas en el producto final. Es decir, la no correspondencia entre lo deseado, lo esperado y lo finalmente alcanzado en el proceso de instanciación.

Entre otros contextos, en la instanciación de arquitecturas de microservicios de software (MSA), se ha extendido el uso de desarrollo continuo (DC), como un proceso iterativo de desarrollo de aplicaciones que incluye Integración continua (IC), entrega continua, pruebas continuas (TC) y despliegue continuo. Este escenario facilita acompañar el ciclo de desarrollo con una verificación y validación continua de la transformación de los requisitos funcionales en cada etapa del proceso, hasta quedar completamente incorporados en el producto entregado.

En la formulación precisa del requisito que lo haga rastreable, comprobable y verificable, es crucial poder hacer el seguimiento de este durante todas las etapas del desarrollo, garantizando el mejoramiento de las condiciones de calidad del producto. Se plantea el uso de técnicas de procesamiento de lenguaje natural (PLN), enriquecidas con modelos de dominio, para hacer el seguimiento, validación y verificación de cada requisito funcional. Asimismo, se han identificado algunos vacíos

en el conocimiento como: el problema de generalización de dominio basado en una frontera lingüística, la limitación de la investigación en contextos específicos de dominio, y el acceso a datos de dominio limitado. Paralelamente, la industria ha presentado la necesidad de hacer rastreo del requisito desde su levantamiento hasta la etapa de testeo.

Las nuevas tendencias de la industria, exigen la eficiencia de los equipos desarrolladores, esta es la razón por la que las metodologías ágiles han cobrado protagonismo a la par del desarrollo continuo; es así como objetivo de este trabajo está direccionado hacia la búsqueda del enriquecimiento de las técnicas estadísticas en el PLN a través de un sistema de reglas de dominio que permita no solo la formulación precisa, minimizando las ambigüedades del lenguaje natural; sino la automatización a través de la creación de artefactos que permitan hacer el rastreo del requisito en todas las etapas del desarrollo aportando de esta manera a una posible solución en el tratamiento de estos en la práctica de la Ingeniería de Software.

Palabras clave: ingeniería de requisitos; desarrollo continuo; PLN

Abstract

The life cycle of a Software Project starts with the proper management of the software requirements. This is a knowledge area which gathers the activities of elicitation, analysis, specification, validation, and requirement management throughout the development cycle. The basic specification, which is expressed in natural language, is transformed along the cycle into artifacts that are deployed and executed as a product.

A problem identified in software engineering is the discrepancy between the functional requirements specified for the product and the features that are finally incorporated into the final product. It means, the dissimilarity between what is desired, what is expected and what is finally achieved in the product instantiation process.

Among other contexts, in the instantiation of microservices software architectures (MSA), the use of continuous development (CD) has become widespread as an iterative process of application development that includes continuous integration (CI), continuous delivery, continuous testing (CT) and continuous deployment. This scenario facilitates to support the development cycle with continuous verification and validation of the transformation of functional requirements at each stage of the process until they are fully incorporated into the delivered product.

Along with the precise formulation of the requirement that makes it traceable, testable, and verifiable, it is crucial to be able to track it during all stages of development, ensuring the improvement of the quality conditions of the product. The use of natural language processing (NLP) techniques, enriched with domain models, is proposed to track, validate, and verify each functional requirement. Furthermore, some knowledge gaps have been identified such as: the problem of domain generalization based on a linguistic boundary, the limitation of research in domain-specific contexts, and the limited access to domain data. In parallel, industry has presented the need to track the requirement from the requirement gathering to the testing stage.

New industry trends demand team efficiency, this is the reason why agile methodologies have gained prominence along with continuous development; this is how the objective of this work is directed towards the search for the enrichment of statistical techniques in PLN through a system of domain rules that allows not only the precise formulation, minimizing the ambiguities of natural language, but also the automation through the creation of a system of domain rules that allows not only the precise formulation, but the automation through the creation of artefacts that allow the tracing of the requirement in all stages of development, thus contributing to a possible solution in the treatment of the requirements of Software Engineering.

***Keywords**: requirement engineering; continuous development; NLP*

1. Introducción

Al hablar de ingeniería de requisitos es necesario resaltar que esta comprende una etapa en el ciclo de vida del software que está determinada por un conjunto de actividades como el levantamiento, análisis, especificación, validación y administración de los requisitos; de ahí que, no se esté hablando de un tema intrascendente, por el contrario, la relevancia de la correcta gestión de estos se ve reflejado en la calidad del producto que es entregado; (Ahmed, 2018) propone que los requisitos presentan un gran impacto en la calidad del software dado que hay dos tipos de conocimiento que es trasmitido, uno explícito y otro tácito, que en ocasiones no se comunica dada la falta de tecnicismos por parte del cliente y un desconocimiento por parte del desarrollador.

El conocimiento de un proyecto esta fragmentado entre la diversidad de documentos y la cantidad de personas que participan, esta situación lleva a que se pueda mal interpretar la información u omitir conceptos relacionados de gran relevancia para el correcto diseño y desarrollo del software. Es fundamental poder generar un sistema que permita conservar los atributos de los requisitos: precisión, integridad, consistencia, claridad y trazabilidad; en la industria el hecho de no cumplir con estos atributos trae como consecuencia errores en el desarrollo que finalmente alteran el alcance en tiempo y costo de los proyectos.

El objetivo de este trabajo está direccionado hacia la búsqueda del enriquecimiento de las técnicas estadísticas en el PLN a través de un sistema de reglas de dominio que se convierta en la base de un lenguaje común que describa las interrelaciones del proyecto pero que sea ajustado a la semántica de un lenguaje específico de dominio; de esta manera se permite conservar los atributos de la formulación de los requisitos, minimizando las ambigüedades del lenguaje natural pero además se propone la automatización a través de la creación de artefactos que permitan hacer el rastreo del requisito en todas las etapas del desarrollo desde su levantamiento hasta el testeo aportando de esta manera a una posible solución en el tratamiento de estos en la práctica de la Ingeniería de Software.

2. Propuesta

(Shankar et al., 2020) establece como crítica la actividad de levantamiento de los requisitos dado que de ahí deriva no solo el diseño sino las funcionalidades y las características que se desean encontrar en el producto terminado; asimismo, este diseño debe contar con objetivos y restricciones, pues toda esta información se documentará como soporte a las especificaciones del software (Serna M. & Serna A., 2020).

La importancia de la ingeniería de requisitos ha sido ampliamente documentada, no obstante, se ha tratado de implementar diferentes métodos, técnicas y herramientas para lograr la forma más apropiada para su gestión, sin embargo, existen algunos desafíos que aún no se logran superar eficientemente. Una de las grandes dificultades es el uso del lenguaje natural como una lengua franca a la hora de levantar y documentar los requisitos (Zhao et al., 2021) y es precisamente este lenguaje el que domina la industria del software (Bruel et al., 2021).

Según (Alzayed & Al-Hunaiyyan, 2021) el uso del lenguaje natural trae consigo problemas como: ambigüedades, inconsistencias, e incompletitud, de ahí que, las técnicas de PLN como: Segmentación de oraciones, etiquetado POS, tokenización, análisis morfológico y análisis sintáctico hayan cobrado importancia como apoyo a la gestión de los requisitos del software, no obstante, (Alzayed & Al-Hunaiyyan, 2021) resalta que estas técnicas han centrado su trabajo solo en las actividades de levantamiento y análisis, y se deben reforzar temas relacionados con: algoritmos que mejoren ambigüedad, evaluar el uso de ontologías como una posibilidad, desarrollar un etiquetado POS con mayor precisión, entre otros; cualquier método exitoso debe posibilitar una manera de reunir y entender las intenciones de los interesados, pero además cumplir con el alto grado de precisión que exige la construcción de software (Bruel et al., 2021).

La mayoría de los modelos estadísticos están entrenados con *corpus* de lenguaje cotidiano como artículos de periódico, lo que según (Zhao et al., 2021) hace poco confiables estos modelos a la hora de trabajar textos de dominios específicos como requisitos de software. Para (Bruel et al., 2021) es claro en que los mecanismos que se utilicen deben permitir la interacción de expertos de domino, que no son expertos en requisitos, con el desarrollador. Según (Serna M. & Serna A., 2020) uno de los problemas de mayor importancia radica en que la escritura de los requisitos en lenguaje natural no son insumos para la verificación y la validación de técnicas automatizadas.

La no correspondencia entre los requisitos funcionales especificados para el producto y las características que son finalmente incorporadas en su instanciación incrementa los costos y tiempos de entrega, los cuales (Serna M. & Serna A., 2020) relaciona directamente con los procesos de reingeniería y el diseño de la arquitectura. (Zhao et al., 2021) resalta una falta de evaluación industrial como apoyo a los modelos y muestra que se está en el momento preciso para darle madurez a esta área de conocimiento, se debe apostar por la realización de estudios y proponer herramientas viables para la industria.

Las nuevas tendencias del mercado exigen la eficiencia de los equipos desarrolladores, esta es la razón por la que las metodologías ágiles han cobrado protagonismo a la par de las prácticas continuas, las cuales permiten acelerar el desarrollo sin comprometer la calidad del software

(Shahin et al., 2017). Este desarrollo requiere rapidez en la determinación de nuevas funcionalidades, evolución y reusabilidad de las arquitecturas, que por supuesto viene ligado de un proceso de verificación y validación antes de ser entregado.

Cada una de estas etapas de desarrollo continuo puede estar soportada por diferentes tecnologías y estas a su vez de una gestión de requisitos para cada uno de los ciclos. La importancia de la ingeniería de requisitos empieza a escalar por la necesidad urgente de optimizar los tiempos de desarrollo, y una correcta gestión de estos durante la integración, entrega y despliegue continuos puede disminuir los errores en el proceso.

(Inayat et al., 2015) expone que la comunidad en general de desarrollo de software aún no identifica el papel de la ingeniería de requisitos en las metodologías ágiles y plantea la necesidad de identificar grupos representativos de los diversos tipos de negocios para asegurar una correcta definición de los requisitos.

Esta labor busca establecer las relaciones entre las funciones y restricciones de un sistema con el objetivo de determinar las especificaciones del software y su evolución, esto enmarcado en un conjunto de dominios de ingeniería (Lee & Zhao, 2006). La necesidad de establecer métodos que disminuyan la distancia de comunicación que existe entre los expertos y los desarrolladores ha generado una gran inquietud, (Lee & Zhao, 2006) los expertos de dominio no se comunican como los ingenieros desean, y dado que esta actividad se realiza en lenguaje natural se empiezan a propiciar las inconsistencias con implicaciones directas en el producto; para (Siegemund et al., 2011) los modelos para la ingeniería de requisitos necesitan un alto nivel de abstracción.

Además del método tradicional que se refiere al levantamiento de requisitos en lenguaje natural, existen algunos métodos como: basado en escenarios, en conocimiento, en características y en ontologías (Lee & Zhao, 2006), todos con sus limitaciones, y son los modelos estocásticos usados en el procesamiento del lenguaje natural los más utilizados.

Para (Murtazina & Avdeenko, 2018) abordar la ingeniería de requisitos a través de la construcción colectiva de familias de productos relacionados con un dominio o a partir de ontologías puede proporcionar una interoperabilidad semántica que apoye el razonamiento computacional. (Avdeenko & Pustovalova, 2016) plantea que en la gestión de los requisitos se observa terminología que no es consistente, porque se utiliza diferentes términos para referirse a un concepto o por el contrario se usa el mismo término para referiste a diferentes conceptos.

(Evans, 2004) expone en su libro que el desarrollo de software trae consigo diversas actividades que involucran a los usuarios y es responsabilidad del equipo concebir un cuerpo de conocimiento que esté relacionado con las actividades. Así las cosas, propone un modelo de dominio como una abstracción organizada del conocimiento de un experto y este se puede convertir en la estructura de un lenguaje usado por todos los miembros del equipo permitiéndole a las dos partes tener una comunicación efectiva (Shahin et al., 2017).

Al igual que las prácticas continuas, las arquitecturas de microservicios han cobrado popularidad dadas sus ventajas en tiempo, riesgo y la posibilidad de dividir un gran sistema de software en

partes más pequeñas (Wolff, 2017). Estas arquitecturas favorecen el uso de diferentes tecnologías, teniendo cada microservicio su propio soporte. A pesar de resaltar todas sus bondades es importante tomar en consideración que una arquitectura de este tipo está compuesta por una cantidad de microservicios que deben ser desplegados, controlados y perfectamente operados en una plataforma. Para la ingeniería de requisitos, además de los desafíos implícitos en la especificación y seguimiento de los requisitos de software, estas nuevas arquitecturas escalan los retos a la búsqueda de modelos que permitan una formulación precisa y un adecuado seguimiento y control de estos durante todas las etapas del desarrollo continuo pero además implementado en una arquitectura de microservicios.

Según (Siegemund et al., 2011) es relevante que se pueda validar la integridad del requisito, este el atributo más difícil de lograr y las herramientas adolecen de la capacidad de determinar si hay información relevante omitida por el ingeniero. Los artefactos que se propone desarrollar deben contribuir a la definición precisa del requisito y al rastreo de este en todas las etapas del desarrollo hasta que el software sea aprobado; para (Narayan et al., 2011) la trazabilidad entre artefactos de desarrollo es de gran importancia pues facilita la comprensión y el análisis utilizando la información contenida en ellos.

(Kravari et al., 2020) expone que, a pesar de la importancia y la oferta de herramientas en el mercado, no se ha logrado capturar el significado de los requisitos, lo que nuevamente resalta la necesidad de mecanismos más robustos que muevan los limites lingüísticos y se aproximen desde la semántica para mejorar precisión y exhaustividad en la formulación que a su vez será el insumo para la trazabilidad en el ciclo de vida del producto a través de artefactos en cada etapa.

Este proyecto será probado en el área de la salud donde existen múltiples ontologías de dominio que pueden soportar este trabajo.

3. Referencias

Artículos de revista

- Ahmed, U. (2018). A review on knowledge management in requirement engineering. *IEEE Xplore*, *5*.
- Alzayed, A., & Al-Hunaiyyan, A. (2021). A Bird's Eye View of Natural Language Processing and Requirements Engineering. *International Journal of Advanced Computer Science and Applications*, *12*(5), 81–90. https://doi.org/10.14569/IJACSA.2021.0120512
- Avdeenko, T. V., & Pustovalova, N. V. (2016). The ontology-based approach to support the requirements engineering process. *2016 13th International Scientific-Technical Conference on Actual Problems of Electronic Instrument Engineering, APEIE 2016 - Proceedings*, *2*, 513–518. https://doi.org/10.1109/APEIE.2016.7806406
- Bruel, J. M., Ebersold, S., Galinier, F., Mazzara, M., Naumchev, A., & Meyer, B. (2021). The Role of Formalism in System Requirements. *ACM Computing Surveys*, *54*(5). https://doi.org/10.1145/3448975
- Evans, E. (n.d.). Domain-Driven Design: Tackling Complexity in the Heart of Software. In Addison-Wesley (Ed.), *2004*.
- Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature

review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, *51*, 915–929. https://doi.org/10.1016/j.chb.2014.10.046
- Kravari, K., Antoniou, C., & Bassiliades, N. (2020). Towards a Requirements Engineering Framework based on Semantics. *ACM International Conference Proceeding Series*, *72–76*. https://doi.org/10.1145/3437120.3437278
- Lee, Y., & Zhao, W. (2006). An ontology-based approach for domain requirements elicitation and analysis. *First International Multi- Symposiums on Computer and Computational Sciences, IMSCCS'06*, *2*, 364–371. https://doi.org/10.1109/IMSCCS.2006.252
- Murtazina, M. S., & Avdeenko, T. V. (2018). Ontology-Based Approach to the Requirements Engineering in Agile Environment. *2018 14th International Scientific-Technical Conference on Actual Problems of Electronic Instrument Engineering, APEIE 2018 - Proceedings*, 496–501. https://doi.org/10.1109/APEIE.2018.8546144
- Narayan, N., Bruegge, B., Delater, A., & Paech, B. (2011). Enhanced traceability in model-based CASE tools using ontologies and information retrieval. *2011 4th International Workshop on Managing Requirements Knowledge, MaRK'11 - Part of the 19th IEEE International Requirements Engineering Conference, RE'11*, 24–28. https://doi.org/10.1109/MARK.2011.6046559
- Serna M., E., & Serna A., A. (2020). Process and progress of requirement formalization in software engineering. *Ingeniare*, *28*(3), 411–423. https://doi.org/10.4067/S0718-33052020000300411
- Shahin, M., Ali Babar, M., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, *5*(Ci), 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629
- Shankar, P., Morkos, B., Yadav, D., & Summers, J. D. (2020). Towards the formalization of non-functional requirements in conceptual design. *Research in Engineering Design*, *31*(4), 449–469. https://doi.org/10.1007/s00163-020-00345-6
- Siegemund, K., Thomas, E., Zhao, Y., Pan, J., & Assmann, U. (2011). Towards ontology-driven requirements engineering. *Workshop Semantic Web Enabled Software Engineering at 10th International Semantic Web Conference*, 1–15.
- Wolff, E. (2017). Microservices: Flexible Software architecture. In *2016*. addison-Wesley.
- Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E. V., & Batista-Navarro, R. T. (2021). Natural Language Processing for Requirements Engineering. *ACM Computing Surveys*, *54*(3), 1–42. https://doi.org/10.1145/3444689

Libros

- Evans, E. (2004). Domain-Driven Design: Tackling Complexity in the Heart of Software. In Addison-Wesley (Ed.), *2004*.
- Wolff, E. (2017). Microservices: Flexible Software architecture. In Addison-Wesley (Ed.), *2016.*