HERRAMIENTA PARA LA ESTIMACIÓN DE HISTORIAS DE USUARIO ASOCIADAS A REQUISITOS NO FUNCIONALES

Liliana González-Palacio[*]
Daniela Villegas-Osorio[**]
Sebastián Luna-Reinosa[***]
Mariana Vásquez[****]
John García-Giraldo[*****]
Elizabeth Suescún Monsalve[******]

Received: 19/09/2023 • Accepted: 08/02/2024
https://doi.org/10.22395/rium.v23n44a6

Resumen

Uno de los primeros pasos para el desarrollo de cualquier producto de *software* es la definición de sus requisitos funcionales y no funcionales, estos últimos también denominados requisitos de calidad. En este proceso participan varias partes interesadas. Sin embargo, estimar el esfuerzo necesario para implantar estos requisitos, y en particular los de calidad, es una tarea compleja. Para superar este reto, presentamos Story Points Predictor (SPP), una herramienta para predecir el esfuerzo necesario en la implementación de los requisitos de calidad mediante técnicas de inteligencia artificial. Basándose en datos históricos, SPP estima el tamaño de los requisitos y los clasifica en tres grupos: pequeños, medianos y grandes. Los equipos de desarrollo pueden utilizar SPP para complementar su proceso de estimación del esfuerzo de los requisitos. Los resultados experimentales muestran que SPP tiene una precisión del 72 %.

[*] Escuela ciencias aplicadas e ingeniería, Universidad EAFIT, Medellín-Colombia (correo: lgonzalez8@eafit.edu.co) Orcid: https://orcid.org/0000-0002-6029-400X

[**] Lendingfront Colombia (dvillegasosorio@gmail.com) Orcid: https://orcid.org/0000-0002-6393-2185

[***] Globant, España (correo: Sebastian.luna@globant.com) Orcid: https://orcid.org/0000-0002-2780-9833

[****] Escuela ciencias aplicadas e ingeniería, Universidad EAFIT, Medellín-Colombia (correo: mvasqueze@eafit.edu.co) Orcid: https://orcid.org/0009-0000-5490-8477

[*****] Facultad de ingeniería, Universidad de Medellín, Medellín- Colombia (correo: jmgarcia@udemedellin.edu.co) Orcid: https://orcid.org/0000-0002-4759-346X

[******] Escuela ciencias aplicadas e ingeniería, Universidad EAFIT, Medellín-Colombia (correo: esuescu1@eafit.edu.co) Orcid: https://orcid.org/0000-0001-7872-7638

Palabras clave: inteligencia artificial, calidad, predicción, desarrollo de software, estimación, proyectos ágiles, priorización, herramientas de estimación.

TOOL FOR ESTIMATING USER STORIES ASSOCIATED WITH NON-FUNCTIONAL REQUIREMENTS

Abstract

One of the initial steps in developing any software product is defining its functional and non-functional requirements, the latter also known as quality requirements. Several stakeholders are involved in this process. However, estimating the effort needed to implement these requirements, particularly the quality ones, is a complex task. To overcome this challenge, we present Story Points Predictor (SPP), a tool for predicting the effort required to implement quality requirements using artificial intelligence techniques. Based on historical data, SPP estimates the size of the requirements and classifies them into three groups: small, medium, and large. Development teams can use SPP to complement their effort estimation process for requirements. Experimental results show that SPP has a 72% accuracy.

Keywords: artificial intelligence, quality, prediction, software development, estimation, agile projects, prioritization, estimation tools.

INTRODUCCIÓN

La calidad del *software* es fundamental para garantizar el éxito de un proyecto de software [1]. Los requisitos de calidad (o requisitos no funcionales) son aquellos que permiten asegurar la calidad del software. Incluyen aspectos como: disponibilidad, seguridad, rendimiento, escalabilidad, portabilidad y usabilidad, entre otros [2].

Los continuos avances tecnológicos, como la computación en la nube, el Internet de las Cosas, la robótica, combinados con los métodos ágiles de desarrollo de software de tendencia (por ejemplo, XP, Scrum, Kanban), abren nuevos retos para asegurar un nivel adecuado de cumplimiento en los requisitos de calidad y su correcta estimación en relación con el esfuerzo que requieren en fase de implementación [3-4].

Las metodologías de desarrollo ágil no ofrecen el soporte necesario para ayudar en estas estimaciones [2, 4]. Para superar este reto, presentamos Story Points Predictor (SPP), una herramienta de software que ayuda a los equipos de desarrollo a predecir el esfuerzo necesario para implementar requisitos de calidad bajo enfoques ágiles.

SPP está basado en inteligencia artificial, usando técnicas de Procesamiento del Lenguaje Natural (PLN). Se requiere como insumo la lista de los requisitos de calidad redactados en forma de historias de usuario, y como salida se obtiene el esfuerzo que puede suponer para un equipo promedio implantar dichos requisitos. Para favorecer su adopción, SPP puede integrarse con herramientas comunes de gestión de proyectos; en su implementación actual, SPP está integrado con Jira. Además, se pueden cargar los requisitos de calidad en un formato determinado (archivo externo) e introducir estos mismos a la plataforma.

Buscando mostrar el proceso de desarrollo de SPP, este artículo se estructura de la siguiente manera: la siguiente sección presenta los problemas y antecedentes que motivan la construcción de esta herramienta. La sección 2 presenta una explicación sobre los materiales y métodos, lo cual incluye detalles del desarrollo de SPP, incluyendo sus requisitos, arquitectura, detalles sobre *frameworks*, lenguajes de programación, modelos de aprendizaje automático y metodología de trabajo del equipo. La sección 3 propone un ejemplo ilustrativo del uso de la herramienta desarrollada. La sección 4 proporciona los detalles sobre el entrenamiento y la validación. Y la sección 5 presenta las conclusiones.

1. PROBLEMÁTICA Y ANTECEDENTES

La definición de los requisitos de calidad o requisitos no funcionales es una tarea durante la fase de ingeniería de requisitos, al inicio de cualquier desarrollo de software. Una buena estimación de los atributos de calidad puede evitar muchas situaciones problemáticas durante un proyecto. Con frecuencia, estos requisitos se ignoran o se

subestiman, lo cual supone un mayor esfuerzo en términos de trabajo y dinero para incluirlos en fase de implementación. Esta situación es más notoria si se usan enfoques ágiles como Scrum en la gestión del proyecto [5-6].

En la actualidad, existen muchas formas de estimar proyectos de software, que se aplican en función de las necesidades y la metodología de desarrollo de cada equipo. Algunas pueden basarse en el juicio de expertos o en la estimación por analogía, mientras que otras pueden utilizar técnicas más sofisticadas como COCOMO, análisis por puntos función, puntos de casos de uso, método Delphi, entre otras [5-6].

Varias herramientas soportan las técnicas mencionadas, permitiendo la interacción dentro del grupo para estimar cada uno de los requisitos del proyecto [7-9]. Algunas opciones populares son:

– SEER-SEM (Software Estimation and Relationship): proporciona soporte para estimaciones utilizando modelos como COCOMO.

– QSM SLIM (Software Lifecycle Management): facilita la estimación de costos y esfuerzo para proyectos adaptándose a diferentes técnicas de estimación, incluyendo Function Point Analysis.

– Price&Cost: permite la estimación de costos de proyectos, incluyendo requisitos no funcionales, y ofrece características para crear estimaciones detalladas en diversos modelos de puntuación.

– Story Point Scrum Poker: se utiliza en la metodología Scrum para estimar cuánto tiempo tomará completar tareas. Permite asignar "puntos de historia" a las tareas, lo que facilita la planificación y la gestión del trabajo en un proyecto ágil. Esto se hace colaborativamente y mejora la precisión de las estimaciones del equipo.

– Pivotal Tracker: Esta herramienta se centra en el desarrollo ágil y facilita la planificación y la estimación de historias de usuario.

– VersionOne: plataforma de gestión ágil que ayuda a estimar el esfuerzo requerido para las historias de usuario utilizando puntos de historia u otras métricas.

Las herramientas listadas anteriormente apoyan la estimación, pero, hasta donde se ha podido verificar, ninguna se especializa en los requisitos de calidad ni tampoco utiliza técnicas de inteligencia artificial.

Story Points Predictor es una aplicación que ayuda en el proceso de estimación de requisitos de calidad mediante un algoritmo de inteligencia artificial para el reconocimiento de texto por medio de palabras clave en una frase, en este caso título y descripción, basados en estimaciones de diferentes conjuntos de datos públicos.

2. MATERIALES Y MÉTODOS

Esta sección explica la metodología seguida por el grupo de trabajo, la arquitectura del Story Points Predictor y su funcionalidad. También se presentan otros detalles de la implementación disponible en un repositorio de control de versiones ampliamente usado.

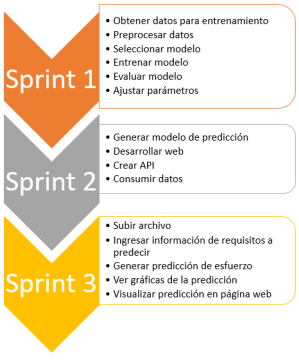
Se eligió Scrum como enfoque de desarrollo. Se trata de un marco utilizado en equipos que gestionan proyectos complejos. Bajo esta filosofía, el propósito es entregar valor en periodos cortos de tiempo, llamados *sprints*. Este proyecto, en concreto, se desarrolló en ciclos de 3 semanas. Una vez finalizado cada sprint, se entregaba un software funcional. Inicialmente, se generó un *backlog*, que luego fue priorizado por todas las partes interesadas. Al final de cada sprint, se evaluaba el progreso, se actualizaba el backlog y se continuaba con el desarrollo.

2.1. Funcionalidades

Requisitos funcionales: i) el usuario puede enviar un archivo CSV (Comma Separated Values) que contenga la lista de requisitos para que el sistema prediga su puntuación; ii) el sistema mostrará gráficos relacionados con la predicción realizada; iii) los usuarios pueden cambiar/personalizar la vista de resultados; iv) el usuario puede buscar/filtrar resultados.

Requisitos no funcionales: i) la aplicación debe ser fácil de usar, con un tiempo de capacitación en su uso no mayor de 30 minutos; ii) la interfaz de la aplicación debe ser compatible con la interfaz de la plataforma Jira; iii) el desarrollo de la aplicación debe ser con el lenguaje Python y el *framework* Flask; iv) la aplicación se despliega sobre navegadores como Chrome y Firefox. Una vez listados los requisitos de la aplicación, en la figura 1 es posible visualizar el backlog construido como una forma de representar dichos requisitos de forma priorizada:

Figura 1. Backlog de producto

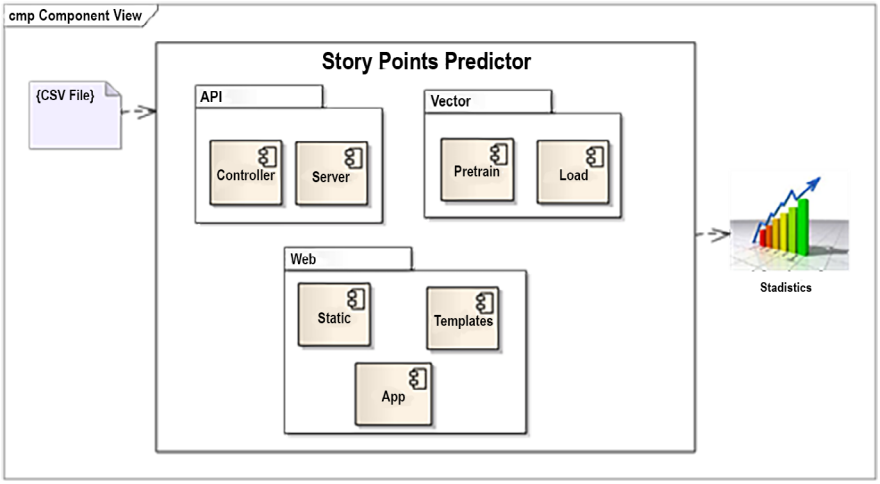


Fuente: elaboración propia.

2.2. Arquitectura del SPP

En la figura 2 se muestra la arquitectura a nivel interno y las interacciones de la aplicación.

Figura 2. Arquitectura de la aplicación



Fuente: elaboración propia.

SPP es una herramienta web que utiliza el framework Flask y Python en el *front-end* y en el back-end; el front-end utiliza SSR (Server Side Rendering) que es el proceso de renderizar la aplicación web en el servidor y luego pasar los datos al navegador [5]. La aplicación no contiene almacenamiento de datos y se alimenta de conjuntos de datos libres de la plataforma Jira.

La aplicación se divide en tres partes, la API (Application Programming Interface), la Web y el Vector. El vector es donde se entrena el modelo y donde almacenamos el archivo entrenado para que pueda ser reutilizado por la API. La API es el back-end de la aplicación, encargada de cargar el modelo entrenado y realizar las operaciones para predecir con base en el fichero de entrada la complejidad de una US (User Story). La web es el front-end de la aplicación encargado de realizar las peticiones a la API y mostrar las predicciones en el navegador.

El modelo de aprendizaje automático con el que se realizan las predicciones es un modelo supervisado construido con FastText, una herramienta desarrollada por Facebook, que es una extensión de la conocida herramienta de incrustación de palabras Word2Vec creada previamente por un equipo de investigación liderado por Tomás Mikolov en Google [6].

En relación con la implementación, Story Points Predictor emplea un modelo de inteligencia artificial supervisada. A partir de las palabras en un título y su descripción, crea un vector que se utiliza para clasificar nuevos requerimientos. Cuando un usuario ingresa un título y descripción del requerimiento, el sistema clasifica y predice su calificación automáticamente.

En el back-end se utilizó el lenguaje de programación Python con la versión 3.7, y además se utilizó el framework Flask con la versión 1.1.1. Los conjuntos de datos de entrenamiento del modelo se tomaron de la plataforma Jira que proporciona algunos conjuntos de datos gratuitos [5].

Con la librería FastText, se construyó el modelo supervisado de aprendizaje automático para la predicción de requisitos de calidad [6].

En las siguientes tablas es posible encontrar algunos detalles.

Tabla 1. Metadatos del ejecutable actual, versión 1.0

URL de los ejecutables de esta versión: https://drive.google.com/drive/folders/17LQ8i4fPkca5HAUI7V3sJRxE_Vo1x5uu?usp=sharing
Plataforma informática / Sistema operativo: Linux (Ubuntu)
Requisitos de instalación y dependencias: Python:https://www.python.org/downloads/, FastText library (Python): https://fasttext.cc/docs/en/supervised-tutorial.html, Flask (backend y frontend): https://flask.palletsprojects.com/en/1.1.x/

Fuente: elaboración propia.

Tabla 2. Metadatos del código actual, versión 1.

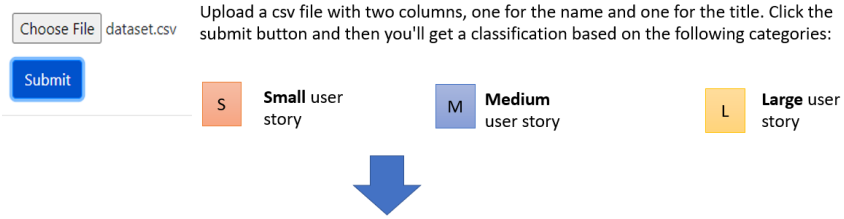
Enlace permanente al código/repositorio utilizado en esta versión del código: https://github.com/DanielaVO/StoryPointsPredictor
Licencia legal del código: https://www.apache.org/licenses/LICENSE-2.0.txt
Sistema de control de versiones de código utilizado: GitHub
Lenguaje de programación utilizado: Python FastText library (Python): https://fasttext.cc/docs/e n/supervisedtutorial.html, Flask (backend and frontend): https://flask.palletsprojects.com/en/1.1.x/

Fuente: elaboración propia.

3. EJEMPLO ILUSTRATIVO

El usuario debe adjuntar un archivo CSV con la siguiente estructura: título, descripción. En este archivo están los requisitos que se quieren predecir. Una vez adjuntado, pulsando el botón Enviar (figura 3), el sistema comenzará inmediatamente el proceso de predicción. Una vez que el sistema arroje una predicción, esta se mostrará en figura 3, la cual contiene la información de la predicción en cuanto al título del requerimiento, descripción, y predicción arrojada por el sistema, la cual puede ser S (pequeño), M (mediano) y L (grande).

Figura 3. Interfaces de la aplicación SPP

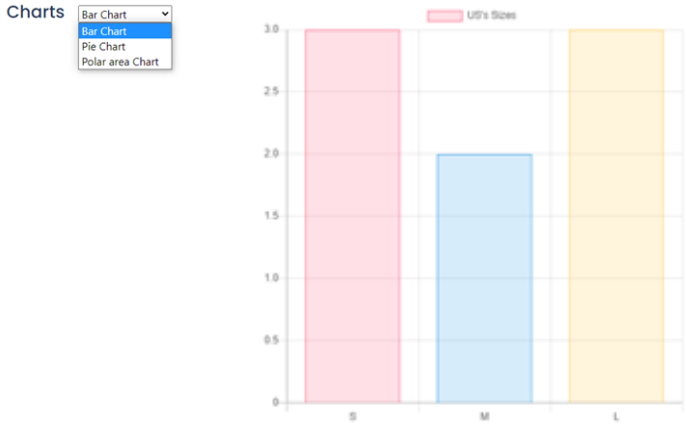


Title	Description	Prediction
Ingresar archivo	Como ingeniero de software deseo ingresar un archivo con historias de usuario para la predicción de su tamaño	S
Hacer login	Como usuario debo hacer login para ingresar a todos los beneficios de la aplicación	M
Login con otras plataformas	Como usuario debo hacer login empleando mi usuario de Google, Facebook, Microsoft	L

Fuente: elaboración propia.

Una vez finalizada la ejecución de la predicción, el usuario podrá visualizar el consolidado del sistema navegando entre los diferentes gráficos que el sistema genera.

Figura 4. Estadísticas generadas por la aplicación



Fuente: elaboración propia.

4. ENTRENAMIENTO Y VALIDACIÓN

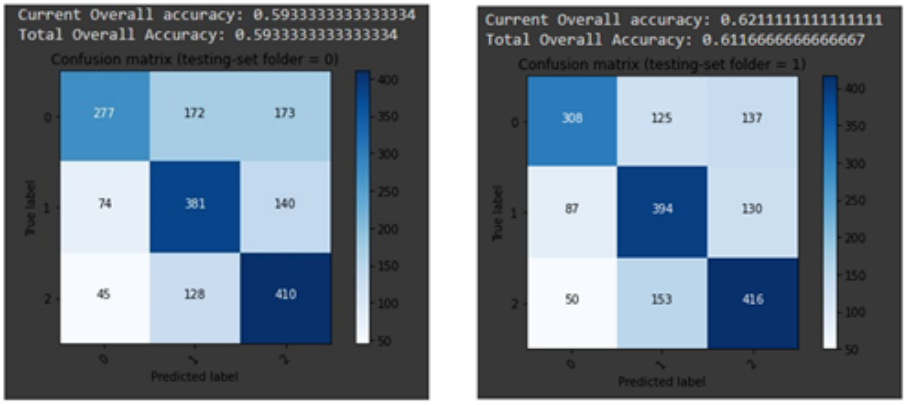
Story Points Predictor fue evaluado mediante conjuntos de datos gratuitos de Jira. Durante la validación, se ingresó en total 16 archivos .csv (conjuntos de datos) de 16 proyectos. Estos proyectos se recopilaron de 9 repositorios de código abierto cuyo archivo se denomina "[nombre del proyecto].csv", por ejemplo, mesos.csv*.

* El enlace es: https://github.com/SEAnalytics/datasets/tree/master/storypoint/IEEE%20TSE2018/dataset

Los proyectos y los repositorios donde se recogieron los datos son Apache, Appcelerator, Duraspace, Atlassian, Moodle, Lsstcorp, Mulesoft, Spring y Talendforge. Cada archivo .csv contiene 3 columnas: título, descripción y punto de historia.

El 70 % de los conjuntos de datos se utilizó para el entrenamiento de mecanismos de inteligencia artificial. Después, el 30 % restante se utilizó para la validación posterior al entrenamiento. La siguiente matriz (figura 5) muestra los resultados:

Figura 5. Resultados durante el entrenamiento (matriz de confusión).



Fuente: elaboración propia.

La matriz de confusión es una forma para describir el rendimiento de un modelo de clasificación (o "clasificador") en un conjunto de datos de prueba cuyos valores verdaderos se conocen. La matriz de confusión es relativamente fácil de entender. Los elementos de la diagonal de la matriz de confusión representan las historias de usuario que la aplicación Story Points Predictor estimó correctamente: 0,0 historias pequeñas; 1,1 historias medianas; 2,2 historias grandes. En otras posiciones (diferentes de la diagonal) de la matriz de confusión, están las historias de usuario que la app Story Points Predictor estimó incorrectamente. Por ejemplo, 0,1 tiene la cantidad de historias pequeñas que la app calculó como historias medianas. Mientras que 1,2 tiene la cantidad de historias medianas que la app calculó como historias grandes. De acuerdo con los resultados, Story Points Predictor ofrece resultados con un alto nivel de confianza, como se puede apreciar en la figura 4.

5. CONCLUSIONES

En este trabajo se explicó el desarrollo de la herramienta Story Points Predictor para la estimación de requisitos de calidad o no funcionales en etapas tempranas de desarrollo de software. La herramienta emplea mecanismos de inteligencia artificial que favorecen la precisión en los resultados.

SPP se destaca por su enfoque en la estimación de requisitos de calidad, lo que la diferencia de otras herramientas de estimación de proyectos de software que generalmente se centran en aspectos más generales. Esto la convierte en una herramienta valiosa para equipos que ponen un énfasis significativo en la calidad de sus productos.

SPP utiliza técnicas de inteligencia artificial, específicamente el Procesamiento del Lenguaje Natural (PLN) para analizar y reconocer texto en las historias de usuario. Esta capacidad de procesar texto en busca de palabras clave en títulos y descripciones es una ventaja significativa, ya que puede automatizar y mejorar el proceso de estimación.

La integración con herramientas comunes de gestión de proyectos, como Jira, facilita la adopción de SPP en los equipos de desarrollo de software. Esto permite a los equipos aprovechar la funcionalidad de estimación de calidad sin cambiar drásticamente sus flujos de trabajo existentes.

Se empleó el 70 % de los conjuntos de datos para entrenar los mecanismos de inteligencia artificial, lo cual garantiza que el modelo haya sido expuesto a una amplia variedad de datos para aprender patrones y características. Esto contribuye a la confiabilidad de los resultados de SPP.

Story Points Predictor ofrece resultados con un alto nivel de confianza. Esto sugiere que la herramienta es precisa en sus estimaciones de esfuerzo para implementar requisitos de calidad, lo que puede ayudar a los equipos de desarrollo a tomar decisiones más informadas y realistas sobre la planificación de proyectos. Como trabajo futuro se propone: mejorar la precisión de las estimaciones mediante la expansión de conjuntos de datos y características, ampliar la integración con más plataformas de gestión de proyectos, explorar modelos de aprendizaje automático avanzados, recopilar retroalimentación de usuarios y realizar validaciones continuas, abordar la seguridad y privacidad de datos de manera proactiva, mejorar documentación y recursos de capacitación para facilitar la adopción y uso efectivo de SPP.

AGRADECIMIENTOS

A los investigadores de la Universidad Politécnica de Madrid, por su asesoría en este proyecto. A las universidades donde laboran los autores de esta contribución, por favorecer espacios para la investigación y las alianzas interinstitucionales.

REFERENCIAS

[1] Abbas N, Gravell AM, Wills GB. The Impact of Organization, Project, and Governance Variables on Software Quality and Project Success. In: *Proceedings 2010 Agile Conference*, Nashville, TN, USA, 2010 Aug 9-13.

[2] Howarth T, Greenwood D. *Construction Quality Management. Principles and Practice*. Routledge, 2017.

[3] Oriol M, Seppänen P, Behutiye W, Farré C, Kozik R, Martínez-Fernández S. *et al.* Data-Driven Elicitation of Quality Requirements in Agile Companies. In: *International Conference on the Quality of Information and Communications Technology —QUATIC—*, 2019: 49-63. https://doi.org/10.1007/978-3-030-29238-6_4

[4] AbdElazim K, Moawad R, Elfakharany E. A Framework for Requirements Prioritization Process in Agile Software Development. *Journal of Physics: Conference Series*, 2020; 1454, 012001. https://doi.org/10.1088/1742-6596/1454/1/012001

[5] Salamea Bravo MJ, González Palacio L, Oriol Hilari M, Farré Tost C. Estimación y priorización de requisitos no-funcionales para desarrollo de software: estado del arte. En: *Conferencia Internacional de Ingeniería. "Desarrollo e innovación en ingeniería"*. Medellín, Antioquia: Instituto Antioqueño de Investigación, 2020: 150-157.

[6] Remón CA. *Estimación de esfuerzo en el desarrollo de software a partir de una especificación de requerimientos* [doctoral dissertation]. Universidad Nacional de La Plata, 2017.

[7] Abdukalykov R, Hussain I, Kassab M, Ormandjieva O. Quantifying the impact of different non-functional requirements and problem domains on software effort estimation. In: *2011 Ninth International Conference on Software Engineering Research, Management and Applications*, 2011 Aug. IEEE: 2011: 158-165.

[8] Chung L, Nixon BA, Yu E, Mylopoulos J. *Non-functional requirements in software engineering*, Vol. 5. Springer Science & Business Media: 2012.

[9] Ramos F, Costa, A, Perkusich M, Almeida H, Perkusich A. A Non-Functional Requirements Recommendation System for Scrum-based Projects. In: *International Conference on Software Engineering and Knowledge Engineering*, 2018 Jul.