Resumen

Esta investigación consiste en un modelo de calificación de deuda técnica de documentación en proyectos de desarrollo de *software* ágiles, mediante una herramienta de análisis de historias de usuario en inglés que utiliza Procesamiento de Lenguaje Natural (*Natural Language Procesing, NLP*).

La herramienta AQUSA+ se encarga de analizar de forma sintáctica y pragmática un archivo con historias de usuario a través de sus tres componentes principales: título, descripción y criterio de aceptación. Al final de la carga del archivo, presenta un registro de errores que permite al creador identificar posibles anomalías en la escritura, que deben ser corregidas antes de llegar al equipo de desarrollo.

Para validar la eficacia de AQUSA+, se compararon sus calificaciones con las realizadas por un grupo de expertos en la misma muestra de historias de usuario; todos ellos utilizaron una rúbrica de evaluación común, de manera que se estandarizaran los valores para calificar cada criterio evaluado.

Cabe añadir que la calificación final de todos los evaluadores se comparó al utilizar métodos estadísticos que permitieran graficar la discrepancia entre la herramienta y los humanos. Asimismo, AQUSA+ fue evaluada mediante un *benchmark* que identificó el comportamiento de la herramienta con grupos de historias de usuario sin anomalías.

Por su parte, los resultados de las evaluaciones y el *benchmark* permitieron identificar falsos positivos y, por ende, determinar la precisión de la herramienta.

Abstract

This research is about a model that identifies and evaluates technical documentation debt in agile software development projects, using a Natural Language Processing (NLP) based tool called AQUSA+.

The tool analyzes a set of user stories syntactically and pragmatically through their three main components: title, description and acceptance criteria. It allows the user to load a file and then display all the errors that need to be corrected in their textual composition, which may lead to technical debt accumulation.

To validate the performance, AQUSA+ scores were compared to the ones of a set of experts, who used the same sample of user stories and the same evaluation rubric, in order to standardize the values of each quality criteria score.

The final score for each evaluator was graphically displayed, in order to statistically compare it to the one from the tool. Also, a benchmark with a set of user stories with no errors was run on the tool to analyze any unexpected behavior.

The evaluations and the benchmark allowed us to identify false positives, and thus to calculate the precision of the tool.

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Programa de Maestría en Computación

MODELO PARA LA IDENTIFICACIÓN DE DEUDA TÉCNICA DE DOCUMENTACIÓN EN AMBIENTES DE DESARROLLO DE *SOFTWARE* ÁGILES

Tesis para optar por el grado de *Magister Scientiae* en Computación

Kenneth Alberto Cascante Saborío

Tutor: Ignacio Trejos Zelaya

Cartago, Costa Rica

Junio, 2017

Aprobación de tesis

www.tec.ac.cr

TEC | Tecnológico de Costa Rica

Programa de Maestría
en Computación
Teléfono: 2550-2402

APROBACIÓN DE LA TESIS

"Modelo para la Identificación de Deuda Técnica de Documentación en Ambientes de Desarrollo de Software Ágiles"

TRIBUNAL EXAMINADOR

MSc. Ignacio Trejos Zelaya
Profesor Asesor

Dr. Cesar Garita Rodríguez
Profesor Lector

Máster/DEA Luis Carlos Naranjo Zeledón
Profesor Externo

Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

TEC | Tecnológico de Costa Rica
Maestría en Computación
Junio, 2017

Dedicatoria

A mi familia, a los que fueron, a los que son y a los que vendrán, porque no hay nada más importante.

Agradecimientos

Agradezco al profesor Ignacio Trejos por concederme el honor de ser mi tutor en esta investigación; su tiempo, su dedicación y su amabilidad fueron claves para una culminación exitosa.

Quiero también agradecer a mi familia por su constante apoyo, especialmente a mis padres y a mi abuela, Sofía, por su esfuerzo y abnegación para permitirme avanzar en mi carrera profesional. No me alcanzará la vida para retribuirles tanto amor.

También manifiesto agradecimiento a Julio Nuñez, Susan Aguilar, Luis Naranjo, Alexander Chaves y Óscar Vásquez, por su asesoría en sus respectivos campos.

Finalmente, un agradecimiento muy especial a los expertos: Oscar Alfaro, Juan Blanco, Kevin Leandro y Esteban Rodríguez, quienes donaron de su valioso tiempo para realizar las evaluaciones: sin ellos esta investigación no habría podido llevarse a cabo.

1 Tabla de contenidos

Introducción

Este documento de tesis presenta los resultados de una investigación sobre una evaluación de conjuntos de requerimientos de *software* documentados como historias de usuario, por medio del análisis de tres componentes principales: descripción, criterio de aceptación y título, con el fin de señalar inconsistencias en la forma en que son escritas y, de esta manera, evitar deuda de documentación para el proyecto de desarrollo de *software* al que pertenecen.

Inicialmente, se contextualizará el estado del arte del análisis de historias de usuario, mediante técnicas de procesamiento de lenguaje natural. Posterior a ello, se hará una introducción a la herramienta *Automatic Quality User Story Artisan* (AQUSA) propuesta por (Lucassen *et al.*, 2015).

Luego, se hace una descripción del problema que se desea solucionar, acompañada por la justificación del proyecto, los objetivos que se buscó alcanzar, las limitaciones consideradas para esta propuesta y la hipótesis que respalda a la investigación.

Por su parte, en la metodología se plantea el tipo de investigación por desarrollar, la cual es de tipo cuantitativo, con un enfoque evaluativo de las inconsistencias encontradas por la herramienta en los textos de las historias de usuario. En esta sección se describen los criterios de calidad que fueron analizados, los componentes de la historia de usuario, cómo estos se descomponen para el análisis y, por último, la estrategia para realizar las pruebas, extender el modelo de evaluación y determinar la eficacia de la nueva herramienta AQUSA+; todo esto con base en la comparación de un análisis realizado por expertos sobre el mismo conjunto de historias de usuario.

Posteriormente, se describen y analizan los resultados obtenidos gracias a las pruebas realizadas mediante rúbricas de evaluación de los expertos y el análisis de los archivos de historias

de usuario a través de AQUSA+ con sus respectivas conclusiones, recomendaciones y el trabajo futuro para sugerir campos afines a la investigación.

Finalmente, como parte de los anexos, se muestran las evaluaciones de cada uno de los expertos, la herramienta y las historias de usuario utilizadas como *benchmark*.

1 Generalidades de la investigación

1.1 Antecedentes

1.1.1 La necesidad de un desarrollo ágil

El manifiesto ágil (Beck, Sutherland y Schwaber, 2001) fue escrito en febrero de 2001 por un grupo de practicantes de diversas metodologías de programación. En ese documento, llegan a un consenso sobre sus cuatro principales valores:

1. *Individuos e interacciones* sobre procesos y herramientas.
2. *Software funcionando* sobre documentación extensiva.
3. *Colaboración con el cliente* sobre negociación contractual.
4. *Respuesta ante el cambio* sobre seguir un plan.

Las metodologías ágiles nacen como una variante en el ciclo de vida de las aplicaciones. Previo al modelo cascada, el enfoque metodológico era el más usado. Es llamado de esta forma, pues sus etapas suceden una tras otra, como una analogía al efecto de la gravedad.

1.1.2 El modelo de desarrollo en cascada

El desarrollo en cascada posee 5 fases principales, como se ilustra en la figura 1. Dichas fases se extienden en duración dependiendo de la complejidad y el tamaño del sistema por construir.

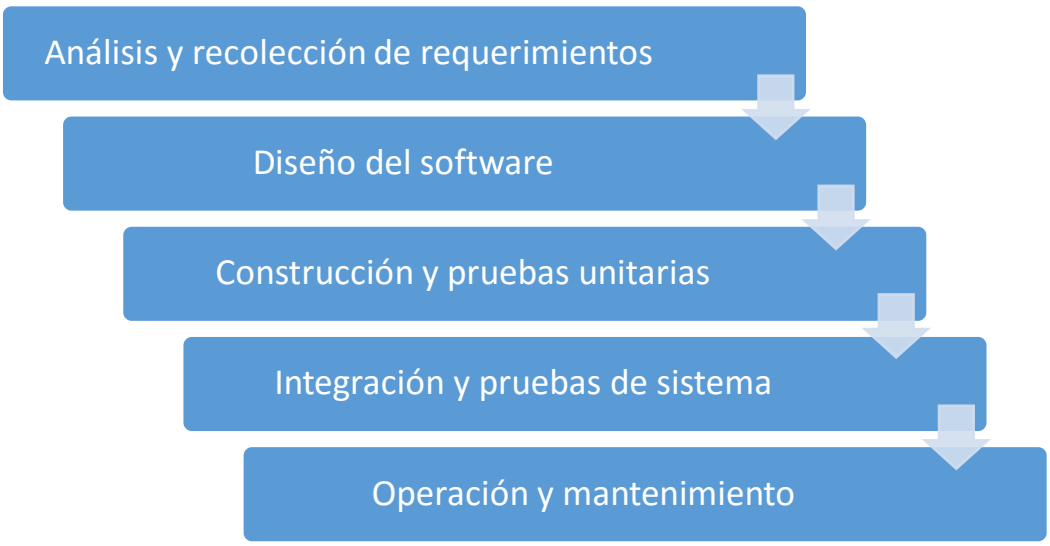

Figura 1. Modelo de desarrollo en cascada.

Las cinco fases mencionadas corresponden al "esqueleto" del ciclo de vida del proyecto; sin embargo, se pueden identificar algunas subfases, como se indica en Scacchi (2001):

En la fase de requerimientos se enlistan:

- Planeamiento o iniciación del sistema: qué sistemas o procesos anteceden al *software* que se pretende construir.
- Análisis de requerimientos y especificación: identifica el problema que el nuevo *software* resolverá.
- Especificación funcional o prototipaje: identifica objetos, sus atributos, relaciones y las operaciones realizadas sobre ellos.

El producto de esta fase corresponde a una serie de documentos, los cuales, después de ser revisados por las partes, se convierten en el contrato de lo que se espera sea el sistema al ser entregado.

Posteriormente se procede al diseño del *software*. Algunas etapas de esta fase son:

- Partición y selección: dados los requerimientos y las especificaciones funcionales, se identifican las piezas que conforman subsistemas lógicos. Se toman decisiones sobre la construcción, la compra o bien, la reutilización de componentes específicos.
- Diseño arquitectónico y especificación de la configuración: define las interfaces entre los diferentes subsistemas.
- Especificación detallada del diseño de los componentes: indica los procedimientos a través de los cuales los datos de entrada requeridos son transformados para producir datos de salida.

Durante esta fase es común el emplear la diagramación de ciertos componentes arquitectónicos del sistema con *Unified Modeling Language (UML*) y, al final, entregar uno o varios documentos de especificación de diseño al equipo de desarrollo.

Con los documentos de diseño, el equipo de desarrollo transforma las especificaciones en código fuente y valida su operación básica (Scacchi, 2001). Esto se conoce como la fase de construcción del *software.*

Una vez construido el sistema, se continúa con la fase de integración y pruebas, la cual vela por la integridad de la configuración arquitectónica del sistema, verificando la consistencia y completitud de sus módulos, las respectivas interfaces, así como el rendimiento del *software* contra los requerimientos solicitados.

También, en esta fase se revisa la documentación del proyecto para asegurar que incluya todas las guías necesarias para la entrega del sistema.

Finalmente, en la fase de operación y mantenimiento, se identifican los siguientes pasos por llevar a cabo:

- Instalación y puesta en marcha: se proveen guías para la instalación del sistema, parámetros en el sistema operativo, permisos de usuario y, en general, se realizan pruebas para garantizar el funcionamiento básico del *software.*
- Capacitación: se imparte un conjunto de indicaciones y guías a los usuarios para que sean capaces de entender las capacidades y limitaciones del sistema.
- Mantenimiento del *software*: se brinda soporte de la aplicación para hacer las reparaciones necesarias o las mejoras de rendimiento.

Conviene señalar que, aparte del enfoque de desarrollo en cascada, existen otros antecedentes de los enfoques ágiles, como el modelo en espiral, el prototipaje rápido, los ciclos de vida iterativos, evolutivos e incrementales y el desarrollo con 'cajas de tiempo' del RAD de James Martin[1].

Por su parte, los enfoques antes mencionados se describirán brevemente en términos de sus requerimientos en las siguientes secciones.

1.1.3 Modelo en espiral de Boehm

El modelo en espiral es un modelo bidimensional secuencial que representa las fases de un método lógico de desarrollo de *software* con atención al manejo del riesgo (Mooz y Forsberg, 2001).

Las fases tempranas involucran a los usuarios para enfocarse en la comprensión de los requerimientos, la viabilidad técnica y los conceptos operacionales, seguidos por las fases de desarrollo en cascada.

[1] RAD = Rapid Application Development, propuesto por James Martin en 1991.

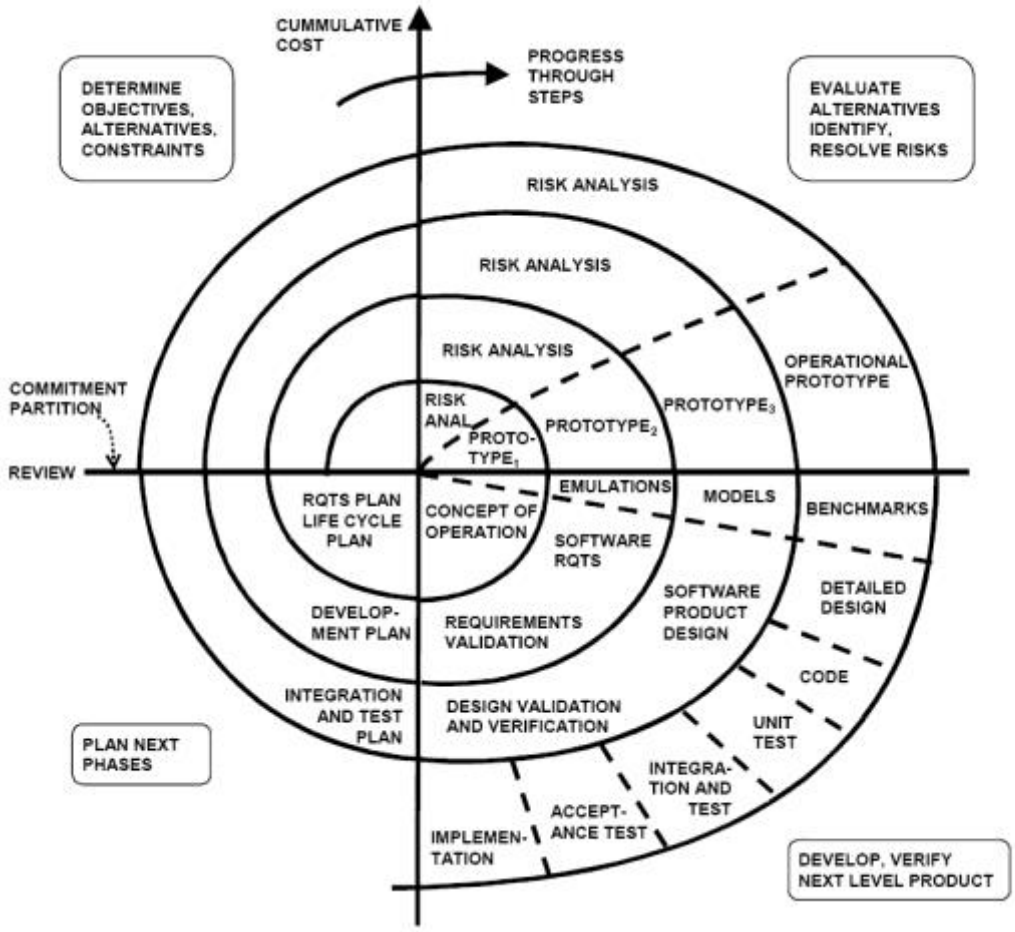

Figura 2. Modelo en Espiral de Boehm (Boehm, 1988).

Tal y como se representa en la figura 2, cada giro de la espiral representa un reto de mitigación de riesgo. Por ejemplo, un riesgo puede ser algún requerimiento que no sea muy claro. Estos giros se repiten hasta que los requerimientos, la solución y la operación de la solución sean completamente entendidos, a fin de para garantizar el posterior desarrollo en cascada, la cual se ilustra en la 'cola' de la espiral (Mooz y Forsberg, 2001).

El modelo también permite el desarrollo incremental al repetir el desarrollo en cascada para cada entrega del *software* que sea requerida.

1.1.4 El proceso unificado

En 1999, Ivar Jacobson *et al*., presentaron lo que se conoce como el proceso unificado de desarrollo de *software* o *Unified Process (UP)*, por sus siglas en inglés. Este consta de cuatro fases principales:

- Incepción.
- Elaboración.
- Construcción.
- Transición.

Estas fases se repiten en cajas de tiempo de una duración específica llamadas *iteraciones*. Al final de cada iteración, se produce un incremento, el cual corresponde a un lanzamiento de una nueva versión del *software* que mejora o añade funcionalidades al producto (Jacobson, Booch y Rumbaugh, 1999).

Similar al modelo en espiral, UP se enfoca en el riesgo, donde los entregables definidos en la fase de elaboración buscan atacar los riesgos más importantes primero.

Por consiguiente, la fase de elaboración representa un pilar fundamental en cada iteración, no solo por la identificación de los principales factores de riesgo, sino porque en ella es donde se espera se capture la mayoría de los requerimientos y se establezca una arquitectura válida.

1.1.5 Scrum: desarrollo iterativo e incremental

Scrum es un enfoque de desarrollo ágil. Inicia con una parte interesada, la cual necesita un producto. Para el desarrollo de tal producto, se cuenta con un equipo. El equipo *Scrum* posee tres roles: dueño de producto, *Scrum* máster y equipo de desarrollo.

El dueño de producto tiene la responsabilidad de decidir qué trabajo se llevará a cabo. El *Scrum* máster se encarga de ser el líder del equipo, remover impedimentos y reducir el ruido que puede distraerlos de llevar a cabo su objetivo.

El equipo de desarrollo toma los requerimientos expuestos por el dueño de producto y lo construye de manera incremental en una serie de periodos cortos de tiempo llamados *sprints*, los cuales normalmente tienen una duración desde una semana hasta un mes.

En *Scrum*, como en otros enfoques ágiles, los requerimientos son llamados historias de usuario. Estas se discuten al principio de cada *sprint* y el equipo decide cuáles se compromete a entregar al final de este.

En el modelo cascada, cada fase podría extenderse por largos lapsos, hasta el punto de que, al llegar a la fase de construcción, algunos de los requerimientos hayan cambiado. *Scrum*, por su parte, acepta el cambio; el enfoque pretende encontrar fallas en el desarrollo, lo antes posible.

Al finalizar el *sprint*, el equipo de desarrollo presenta un grupo de historias convertidas en *software* funcional. Este producto es presentado a los interesados para obtener retroalimentación de manera más rápida y eficiente.

Mediante este enfoque, el equipo es capaz de proveer historias de usuario potencialmente entregables al final de cada *sprint*; es decir, que aporten valor al negocio.

1.1.5.1 Historias de usuario

Una historia de usuario se trata de una narrativa simple cuya función es ilustrar los objetivos de usuario que una funcionalidad de un *software* debe satisfacer (ISO/IEC/IEEE 26515, 2012). Además, son menos formales y estructuradas que los requerimientos listados en el documento de especificación producido al final de la primera fase del modelo en cascada.

Aun así, el estándar *ISO/IEC/IEEE 26515* para la documentación de sistemas de *software* los considera parte de la documentación del proyecto. Dicho documento establece la forma del contenido principal de las historias de usuario, la cual es la siguiente, en idioma inglés: *"As a <role> I want <goal>, so that I can <business value>".* En el español, esta forma se traduce como: *"Yo como un <rol>, quiero <descripción de la funcionalidad>, de manera que pueda <valor de negocio>".* Un ejemplo en inglés de una historia de usuario con el formato antes mencionado es el siguiente: *"**As a power user**, **I can** indicate folders not to backup **so that** my backup drive isn't filled up with things I don't need saved"*. Por su parte, la traducción en español es: "**Yo, como un súper usuario**, **quiero** poder indicar carpetas que no quiero respaldar, **de manera que** mi unidad de respaldo no se llene con cosas que no necesito guardar".

Con base en la estructura *supra* mencionada, se obtiene la información que debe contener una historia de usuario (ISO/IEC/IEEE 26515, 2012):

- El rol de usuario al que aplica.
- El objetivo que la completitud de la historia le permite alcanzar al usuario.
- El valor para el negocio o cliente.
- El criterio de aceptación para identificar cuando la historia de usuario ha sido implementada.

También pueden contener la siguiente información:

- Un identificador único para la historia.
- Detalles adicionales, por ejemplo, notas de conversaciones entre los clientes y el equipo de desarrollo.
- El nombre de la historia.
- El orden de prioridad.

- El tamaño de la historia, la cual corresponde a una estimación de su complejidad realizada por los miembros del equipo de desarrollo.

Una historia con la información anterior se suele considerar como completa; sin embargo, existen algunos atributos adicionales para ser considerada una historia de calidad.

1.1.5.2 Nemónico INVEST para historias de usuario de calidad

En 2003, Bill Wake creó el nemónico *INVEST* para recordar que las historias de usuario de calidad son (Wake, 2003):

- Independientes: no hay dependencias inherentes en otras historias.
- Negociables: hasta el momento en que son incorporadas como parte de una iteración, las historias pueden ser cambiadas o reescritas.
- Valorables: deben proveer un valor al usuario final.
- Estimables: se debe ser capaz de estimar un tamaño para la historia.
- *Small* (Pequeñas): No pueden ser tan grandes, de manera que sea imposible planearlas, subdividirlas en tareas o priorizarlas.
- *Testable* (Verificable): debe proveer la información necesaria para hacer posible el desarrollo de las pruebas.

Las conversaciones que se generan en el equipo a partir de esta técnica, les permite tener un mejor criterio de la historia y que haya un entendimiento compartido de ella.

1.1.5.3 Formato para el criterio de aceptación

Así como las historias de usuario tienen una estructura para su descripción, como se aprecia al inicio de la sección 1.1.5, el criterio de aceptación para esa historia también puede ser formulado con base en una plantilla de escritura.

Uno de los propósitos de un criterio de aceptación es que sea claro cómo probar que lo solicitado en la historia de usuario ha sido completado. Para esto se utiliza la estructura *"Given-When-Then"*, (Dado-Cuando-Entonces), que permite establecer las precondiciones (*Given*), las acciones (*When*) y los resultados (*Then),* propuesta por Dan North en (North, 2006).

Un ejemplo de un criterio de aceptación en este formato es el siguiente: *"**Given** my bank account is in credit, and I made no withdrawals recently, **when** I attempt to withdraw an amount less than my card's limit, **then** the withdrawal should complete without errors or warnings".* La traducción al español corresponde a: "**Dado** que mi cuenta bancaria es en crédito y no he realizado retiros recientemente, **cuando** intente hacer un retiro que es menor al límite de mi tarjeta**, entonces** el retiro debería completarse sin errores o advertencias."

North establece que el criterio de aceptación debe ser suficientemente claro en sus precondiciones, acciones y resultados, al punto de ser "ejecutable"; es decir, pueden ser representados directamente en el código.

1.1.6 Deuda técnica

El concepto de deuda técnica fue introducido por Ward Cunningham en 1992. Se refiere específicamente a la situación en la que la calidad del código a largo plazo se ve reemplazada por una ganancia a corto plazo (Cunningham, 1992).

Cabe señalar que se le llama *deuda*, pues se hace una analogía con los términos financieros de la siguiente forma: si durante el proceso de desarrollo del *software* se toman atajos en alguna de sus fases, su calidad se verá en riesgo, por lo que se dice, entonces, que se acumula una cierta deuda que debe ser pagada en algún momento futuro.

Según McConnell (2008), se puede dividir en deuda intencional y deuda no intencional: la deuda intencional es aquella en la cual el equipo toma una decisión consciente de aceptar cierta

deuda en favor de la necesidad, por ejemplo, de liberar más pronto un producto al mercado. La deuda no intencional es aquella que se acumula por prácticas pobres en el ciclo de vida del proyecto. Comúnmente se tiende a asociar con atajos y malas prácticas de programación, cuando realmente esos atajos se pueden tomar también en el diseño, en la documentación y hasta en las pruebas.

Cada una de las mencionadas anteriormente es, también, un tipo de deuda técnica. A continuación se desarrolla el tipo de deuda técnica que atañe a este estudio: la deuda de documentación.

1.1.7 Deuda de documentación

Dentro de los diferentes tipos de deuda en los que se puede incurrir en un proyecto de desarrollo, se encuentra la *deuda de documentación*, que se refiere a todos aquellos artefactos faltantes, incompletos o incorrectos en la documentación de un sistema (Soares, Alves, Alves, Mendonça y Spínola, 2015).

En Morrison (2011), se listan algunas de las propiedades de la deuda técnica. Una de ellas es el impacto, que se refiere al alcance de los cambios físicos necesarios para remover la deuda. Este impacto incluye posibles cambios en el código, las pruebas, los archivos de configuración y la documentación.

En Soares *et al.* (2015), los autores realizan un estudio para evaluar el impacto que el uso de historias de usuario puede tener en la obtención de deuda técnica en proyectos ágiles.

Conviene aclarar que la investigación tiene dos frentes; una revisión de literatura y un estudio exploratorio. En el primero, se busca identificar las dificultades principales de trabajar con requerimientos ágiles. En el segundo, el estudio exploratorio busca caracterizar las dificultades del uso de historias de usuario en comparación con el método tradicional.

Cada una de las dificultades fue mapeada en indicadores de deuda de documentación donde, algunas de ellas, se consideran las causas de este tipo de deuda (Soares *et al*., 2015). Después de analizar los resultados, los autores llegaron a la conclusión de que la falta de información es la principal dificultad a la hora de trabajar con historias de usuario.

Por su parte, la percepción de los participantes en el estudio fue que la falta de información en las historias de usuario aumenta la complejidad de tareas futuras en el proceso como el mantenimiento y el diseño arquitectónico.

1.1.8 Trabajos relacionados con esta investigación

La búsqueda por el entendimiento de los causantes de la deuda técnica y la forma de administrarla es un tema que se encuentra en investigación actualmente, pues su importancia se hace cada vez más notoria. En su última actualización de la suite de *Visual Studio*[2], la compañía *Microsoft* anunció la integración de su herramienta de manejo de ciclo de vida de aplicaciones *Team Foundation Server (TFS)* con *SonarQube*; una herramienta de código abierto para la administración y reducción de la deuda técnica.

En cuanto a la estimación de la deuda técnica, Yli-Huumo (2014) busca crear un modelo teórico de evaluación y administración de ella en el ciclo de vida de una aplicación. La mayoría de artículos encontrados centran sus análisis de deuda técnica en el código. Por el contrario, el principal aporte del autor mencionado se encuentra en el análisis del ciclo de vida de la aplicación como un todo.

Este estudio es conducido en una empresa de desarrollo de *software* de tamaño mediano, junto con dos líneas de producto independientes. El primer objetivo es entender las causas de la acumulación de deuda técnica en proyectos de *software.*

[2] Actualización realizada el 20 de Julio de 2015.

Posteriormente, los resultados de dicha investigación revelaron que la falta de documentación representa una causa de deuda técnica no intencional, así como la carencia de estándares para codificar o la del conocimiento sobre futuros cambios.

Entre los efectos que la acumulación de deuda técnica puede tener destacan desde horas extra de trabajo, fallas en el sistema y complejidad en el código fuente, hasta la insatisfacción del cliente.

En cuanto a las causas, Soares *et al.* (2015) presentan un estudio para identificar las dificultades más importantes al trabajar con historias de usuario, en vez de casos de uso. La mayor dificultad encontrada en dicha investigación fue la falta de datos. Conviene mencionar que el aporte de Soares *et al.* (2015) contribuye al campo de la deuda técnica a partir de la identificación realizada; sin embargo, no hace un análisis de la sintaxis o la de la composición de las historias de usuario.

En este campo, Plank, Sauer Schaefer (2012) presentan la idea de utilizar técnicas de procesamiento de lenguaje natural (*Natural Language Processing* o *NLP*, por sus siglas en inglés) para apoyar el desarrollo de *software* ágil. Ahí se analizan artefactos como los reportes de error, los comentarios en el código y las bitácoras de integración para determinar conexiones hacia historias de usuario, con el fin de conocer el avance del proyecto.

Los autores proveen un método para rastrear los requerimientos del sistema a través de los artefactos creados en el proceso de desarrollo de *software*, tales como el código, los comentarios, los reportes de fallas o las *'wikis'*. Ciertamente, la investigación demuestra que, aunque las historias son escritas en texto libre, típicamente no son libres de forma. Existen plantillas como la mostrada anteriormente o bien algunas modificaciones como en Zeaaraoui, Bougroun, Bouchentouf Belkasmi (2013).

El NLP permite crear representaciones estructuradas para encontrar interdependencias con los demás artefactos del proceso de desarrollo del proyecto. Para el análisis de historias de usuario con un formato específico, Lucassen *et al.* (2015) presentan un modelo conceptual y una herramienta para el análisis de la calidad de las historias de usuario, mediante el uso de ontologías y el análisis sintáctico, semántico y pragmático de ellas.

En Mayr, Plösch y Körner (2014) se propone un nuevo enfoque para el cálculo automatizado de la deuda técnica, con las ventajas de tener una línea base para la calidad ideal y la habilidad de comparar la deuda técnica con proyectos de calidad conocida. El artículo de Mayr *et al.* presenta tres contribuciones importantes. La primera es un esquema de clasificación sistemática para los enfoques de cálculo de deuda técnica existentes, basado en el paradigma *"Goal/Question/Metric (GQM)"*, el cual especifica categorías y dimensiones para caracterizar Los enfoques de deuda técnica.

La segunda es establecer un modelo de cálculo para costos de remediación de la deuda técnica basado en *benchmarking*, el cual considera diferentes niveles de calidad y permite la comparación con proyectos de calidad conocida.

Finalmente, este modelo es evaluado en dos proyectos de código libre. Los resultados expusieron que el modelo de cálculo es capaz de mostrar costos de remediación esperados para el proyecto, según su calidad y el nivel deseado.

Dautovic, Plösch y Saft (2014) proponen un enfoque basado en herramientas para la inspección de documentos de *software.* La intención es determinar los efectos de las buenas prácticas de documentación en proyectos de software. En su estudio evalúan herramientas existentes utilizadas para el análisis de requerimientos en texto plano, como por ejemplo, *Quality Analyzer of Requirements Specifications (QuARS)* o *Automatic Requirements Measurement*

(ARM). Al respecto, concluyen que no son capaces de reemplazar completamente al experto, pero sí pueden ser utilizadas para hacer la tarea de inspección más sencilla.

Al aplicar la herramienta en 50 documentos, esta fue capaz de calificar dos tercios de alta o muy alta confianza. A su vez, fue posible identificar que el esfuerzo para remediar el defecto fue calificado bajo, debido a que las violaciones encontradas solo afectaban algunas líneas.

Como labor futura, enfatizan en la necesidad agregar reglas para trabajar con términos específicos del dominio, así como glosarios usados en documentos de *software* o en la rastreabilidad de referencias entre documentos de distintas fases del ciclo de vida del proyecto.

1.1.8.1 Enfoque de calidad para historias de usuario

En Lucassen *et al*. (2015) se presenta el modelo para la evaluación de la calidad de usuario más cercano a lo que esta investigación persigue. Los autores definen 14 criterios de aceptación agrupados en tres categorías lingüísticas: sintáctica, semántica y pragmática.

1.1.8.1.1 Categorías y predicados

Debido a que las historias son meramente textuales, en Lucassen *et al*. (2015), las categorías de calidad se agrupan en tres conceptos de la lingüística:

Calidad **sintáctica:** se refieren a la estructura de la historia sin considerar su significado. A este grupo pertenecen las categorías:

- Atómica.
- Mínima.
- Bien formada.

Calidad **semántica:** le conciernen las relaciones y el significado de la historia. Sus categorías son:

- Libre de conflictos.
- Conceptualmente sólida.
- Orientada al problema.
- Sin ambigüedades.

Calidad **pragmática**: se refiere a la forma en la que se comunica un conjunto de requerimientos. Sus categorías son:

- Completa.
- Dependencias explícitas.
- Oración completa.
- Independiente.
- Escalable.
- Uniforme.
- Única.

Si bien algunas categorías pueden ser evaluadas por separado, para saber si una historia es única, completa, uniforme o independiente se deben analizar sus relaciones junto con el conjunto de historias a las que pertenecen.

Para lograr este propósito, los autores utilizan predicados de primer orden para formalizar las relaciones y facilitar su identificación. Por ende, a manera de ejemplo, se procede a mostrar los predicados correspondientes a las relaciones que se tratan de identificar para la categoría **Única.** Sin duda, a la hora de identificar la unicidad de una historia de usuario, se busca que esta no se encuentre duplicada o bien, que exista otra con un significado similar.

Por consiguiente, para detectar estas relaciones, cada parte de la historia de usuario es comparada con la misma parte del resto de historias, mediante una combinación de métricas de

similitud que pueden ser tanto sintácticas (distancia de Levenshtein)[3] como semánticas (utilizando la ontología para encontrar sinónimos). Cuando la similitud alcanza algún límite determinado se necesita el apoyo de un humano para identificar la posible anomalía (Lucassen *et al.*, 2015).

En la figura 3, se observa el predicado que detalla la relación de 'Duplicado completo' o *full duplicate*, la cual se encuentra cuando dos historias son completamente iguales en términos sintácticos.

$$isFullDuplicate(\mu_1, \mu_2) \leftrightarrow \mu_1 =_{syn} \mu_2$$

Figura 3. Predicado de duplicado completo (Lucassen et al., 2015).

El nivel de notación μ corresponde a una historia de usuario. Entonces el predicado anterior se lee de la siguiente forma: "Las historias de usuario 1 y 2 son completamente duplicadas si y solo si son iguales sintácticamente".

En el caso de que existan dos historias que tengan el mismo significado pero con formas diferentes de ser escritas, se identifica la relación 'semánticamente duplicada', la cual se formaliza en el predicado de la figura 4:

$$isSemDuplicate(\mu_1, \mu_2) \leftrightarrow \mu_1 = \mu_2 \wedge \mu_1 \neq_{syn} \mu_2$$

Figura 4. Predicado para la relación 'Es semánticamente duplicada' (Lucassen et al., 2015).

El predicado anterior se interpreta como: "Las historias 1 y 2 son duplicadas semánticamente si y solo si la historia 1 es igual a la historia 2 y la historia 1 no es sintácticamente igual a la historia 2".

[3] Número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra.

Existen más relaciones identificadas para la categoría de unicidad; sin embargo, el propósito de esta sección es demostrar el formato utilizado para caracterizar la lógica detrás de la herramienta de análisis.

1.1.8.2 Modelo conceptual y AQUSA

La utilización de una estructura para las historias de usuario como la que se muestra en la sección 1.1.5 permitió la creación de un modelo conceptual para las historias de usuario descrito en la figura 5. Dicho modelo se desvía del propuesto por Cohn (2004) al abstraer más los conceptos de 'objetivo' y 'valor' en medios y fines, pues algunas plantillas de historias de usuario presentan ciertas variaciones como 'deseo' en vez de 'objetivo' y de 'beneficio' en lugar de 'valor de negocio'.

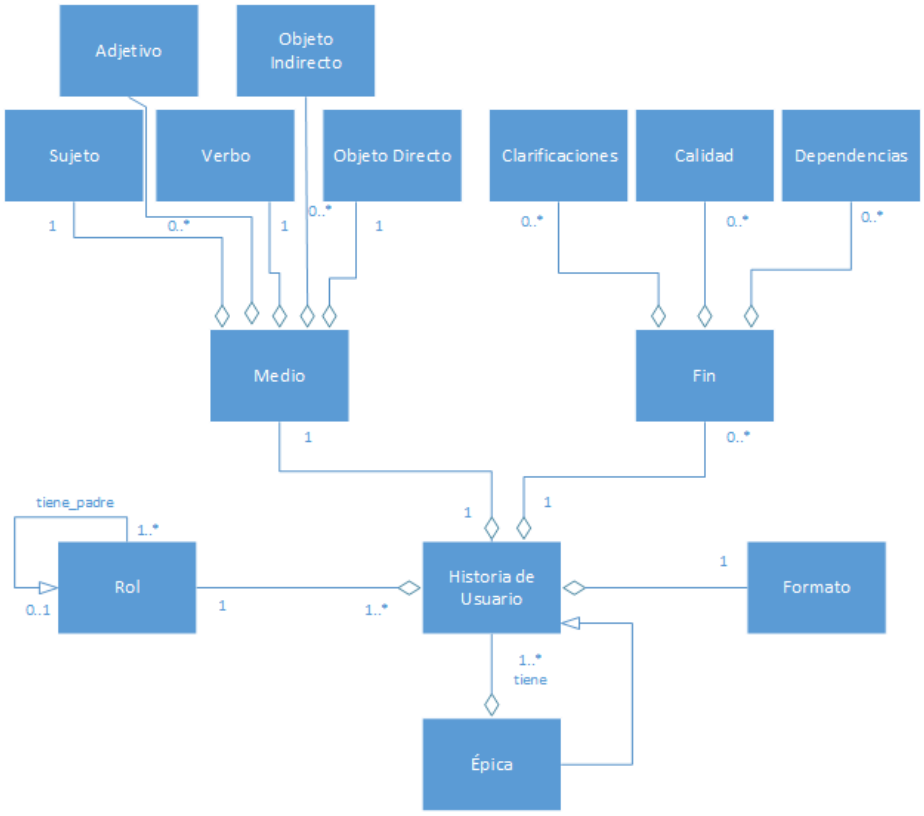

Figura 5. Modelo Conceptual (Lucassen et al., 2015).

Aparte del modelo conceptual y la definición de las categorías y las relaciones que las caracterizan, Lucassen et *al.* (2015) realizan un tercer aporte, mediante la creación de una herramienta llamada *Automatic Quality User Story Artisan (AQUSA)*, la cual se basa en el modelo teórico que los propios autores presentan.

AQUSA es un prototipo que expone defectos y desviaciones de las buenas prácticas a la hora de escribir historias de usuario. Fue creada como una aplicación aparte que analiza un grupo de historias de usuario, independientemente de su origen. Su arquitectura está compuesta por cuatro componentes principales:

- Un analizador lingüístico que busca identificar si la historia está bien formada.
- El modelo AQUSA, donde las historias son procesadas como objetos para alinearlas con el modelo conceptual presentado anteriormente.
- Un potenciador para las historias de usuario. Este se encarga de añadir homónimos, sinónimos e información semántica a partir de ontologías.
- El componente de análisis que corre métodos para verificar los criterios sintácticos y pragmáticos seleccionados, tal y como se observa en la parte inferior de la figura 6.

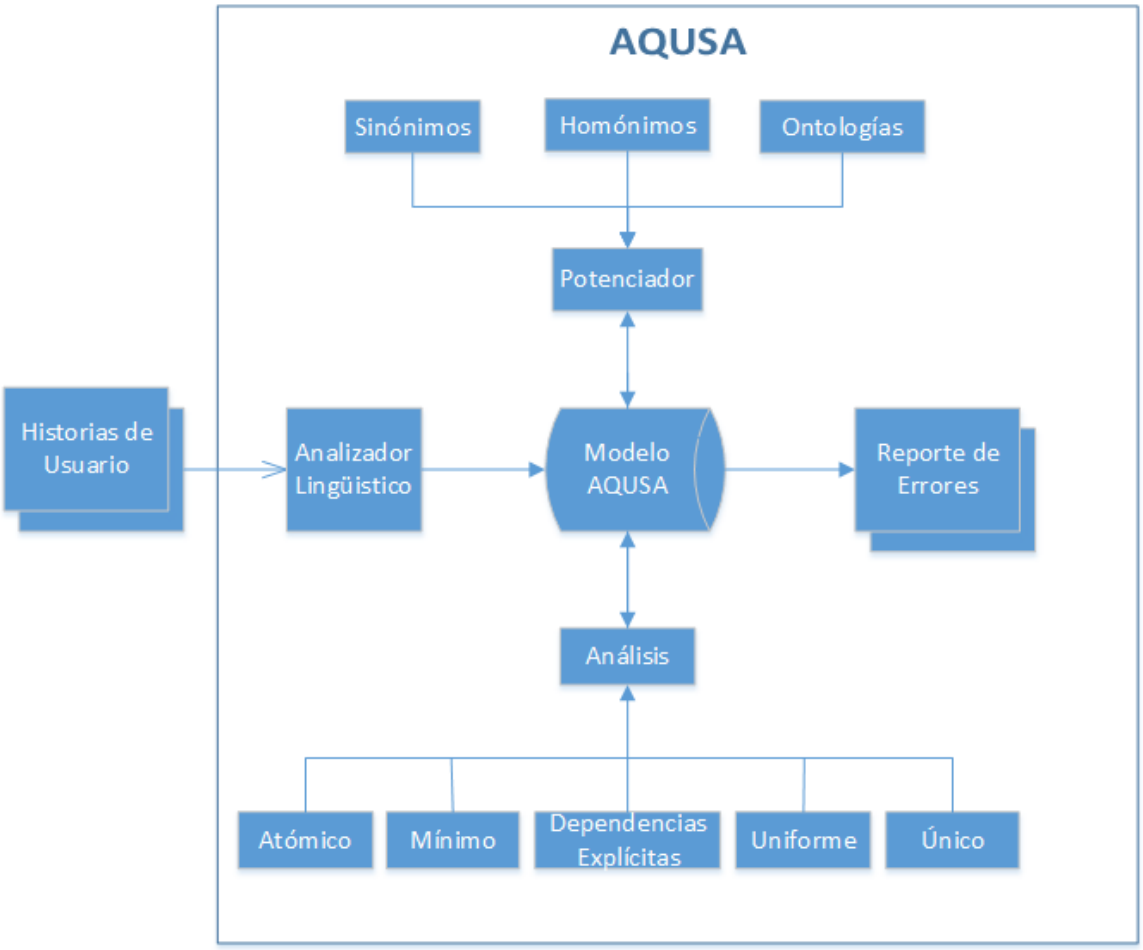

Figura 6. Arquitectura de AQUSA (Lucassen et al., 2015).

Como se puede observar, la arquitectura de AQUSA realiza únicamente el análisis de algunas características pragmáticas y sintácticas, mas no las semánticas de las historias de usuario. Tampoco toma en cuenta el texto correspondiente a los criterios de aceptación para cada una de esas historias, por lo que se considera importante extender esta arquitectura para la inclusión de tales criterios.

Los autores del artículo evaluaron AQUSA en dos conjuntos diferentes de historias y obtuvieron una precisión del 71 % en la detección de errores con un 100 % de cobertura.

1.2 Definición del problema

En 2015, según la encuesta anual de la empresa *Version One*, líder en el mercado de *software* para la administración de proyectos ágiles, el 95 % de las empresas de *software* encuestadas practican enfoques ágiles (Version One, 2014). Gradualmente, las organizaciones se inclinan de forma creciente hacia enfoques de desarrollo ágil como *Scrum*, donde los requerimientos del sistema se definen como 'historias de usuario'. Estas historias (o narrativas) responden al 'qué' del requerimiento y su criterio de aceptación responde al 'cómo'.

En *Scrum*, las iteraciones duran de 1 a 4 semanas, lo que implica que el equipo de desarrollo debe ser capaz de entregar *software* funcional en ese lapso. Es decir, describir las historias de usuario, planearlas, dividirlas en tareas y asignar estas a diferentes miembros del equipo mientras se está en constante comunicación con el dueño de producto.

La urgencia por entregar lo prometido al final de la iteración puede llevar a que se incurra la acumulación de deuda técnica. Un tipo de deuda técnica es la *deuda de documentación*, la cual puede ser determinada al identificar artefactos faltantes, inadecuados o incompletos en la documentación de proyectos de *software* (Soares *et al.*, 2015).

Existen investigaciones empíricas que analizan la deuda de documentación de manera retrospectiva sobre conjuntos de requerimientos ágiles (Soares *et al*., 2015). Sin embargo, no existe una herramienta que sea capaz, a través de análisis de lenguaje natural, de determinar inconsistencias en la escritura de historias de usuario en sus tres componentes más importantes: descripción, criterio de aceptación y título.

El modelo presentado por Lucassen *et al.* (2015) establece criterios de calidad sintácticos y pragmáticos para el análisis de la descripción de una historia de usuario y un prototipo llamado AQUSA. Consiste en una herramienta que consume archivos con descripciones de historias de

usuario y realiza un análisis sintáctico de la composición del texto, al indicar posibles causantes de la deuda de documentación, con base en reglas de calidad previamente definidas.

Al ser una historia de usuario compuesta por tres textos diferentes, esta investigación presenta una extensión a este modelo llamada AQUSA+, a la vez que analiza, además de la descripción, el título y el criterio de aceptación basado en una serie de criterios de calidad definidos en North (2006).

Cabe mencionar que AQUSA+ pretende ser una herramienta de análisis *a priori;* es decir, que permita analizar un conjunto de historias de usuario antes de que sean implementadas y detecte posibles indicadores de futura deuda de documentación. Sin duda, permitirá ejecutar el análisis del conjunto de historias de usuario más sencillo, pues es capaz de detectar errores y reportarlos, así como de indicar las que no presentan anomalías y, de este modo, evitar que un experto deba revisarla. Para su validación, se compara el resultado del análisis contra el de usuarios expertos, con el fin de identificar el nivel de discrepancia entre el análisis humano y el de la herramienta.

1.3 Hipótesis

Se puede crear un modelo para la identificación de la deuda técnica para un proyecto de desarrollo de *software* bajo el enfoque *Scrum*, con base en el análisis de las descripciones y los criterios de aceptación de las historias de usuario, a fin de identificar errores que pueden implicar acumulación de deuda de documentación.

1.4 Justificación

La escritura de requerimientos ágiles o historias de usuario tienen un formato menos estructurado que en el modelo cascada, pues la intención es que genere una conversación entre los desarrolladores y el dueño de producto. Estas conversaciones implican cambios y anotaciones que no necesariamente son documentadas de forma apropiada. Algunos casos incluyen la falta de

componentes básicos o bien, la sobrecarga de la narrativa para sugerir dos o más funcionalidades en una historia.

Esta investigación se fundamenta en la necesidad de proveer una herramienta que apoye a los dueños de producto durante el proceso de desarrollo de las historias de usuario, de manera que este sea capaz de detectar errores que, a la postre, se conviertan en deuda técnica de documentación para el proyecto.

Ciertamente existen trabajos relacionados con este tema, los cuales proponen modelos de análisis sintáctico, pragmático y semántico, mediante el uso de técnicas de NLP. Sin embargo, no existe una herramienta que analice la historia de usuario en sus tres componentes principales.

Al extender el modelo de AQUSA, el dueño de productos y los desarrolladores pueden validar el ¿qué? (descripción), el ¿cómo? (criterio de aceptación) y de responder a la pregunta de cuál es exactamente la funcionalidad que se está dejando de obtener, en caso de la que historia de usuario no se complete (título).

Otra de las razones para esta investigación es la necesidad de reducir el tiempo en que se evalúan las historias de usuario: discriminar entre las que están bien escritas y las que presentan anomalías.

Por ende, poseer una herramienta que "adelante" este trabajo, especialmente en *backlogs* extensos, permitirá al dueño de producto concentrarse de manera más ágil en las historias que necesitan ser divididas o reescritas.

1.5 Objetivos

1.5.1 Objetivo general

Plantear un modelo de estimación de deuda de documentación para proyectos de desarrollo de *software* bajo el enfoque *Scrum* que, mediante el análisis de sus historias de usuario, permita identificar aquellas que requieran una mejora en su composición.

1.5.2 Objetivos específicos

a. Evaluar un conjunto de proyectos de *software*, llevando a cabo la calificación de la deuda de documentación por parte de expertos en el campo para proveer un marco de referencia desde el punto de vista humano.

b. Diseñar y elaborar una herramienta que integre técnicas existentes de análisis pragmático y sintáctico para historias de usuario en inglés de un proyecto de desarrollo de *software* bajo el enfoque *Scrum* y otorgue una calificación de deuda de documentación para el proyecto.

c. Comparar los resultados de la herramienta contra el análisis de los expertos para identificar las similitudes y diferencias entre ambos.

d. Juzgar la herramienta con un *benchmark*, de manera que sirva como referencia para evaluar su precisión.

1.6 Alcance

El alcance de la investigación fue delimitado por los siguientes puntos:

- La construcción de los algoritmos se realizará en el lenguaje de programación *Python*.
- La selección de proyectos, para investigar sus requerimientos, corresponderá a una sola compañía.
- El rango de tiempo en los que estos proyectos fueron iniciados y terminados comprende desde mayo de 2014 hasta febrero de 2016.
- El análisis de las historias comprenderá los componentes pragmáticos y sintácticos. El análisis semántico se encuentra fuera del alcance de nuestra investigación y del desarrollo de la herramienta.
- El criterio de calidad estará dado por la sección 1.1.5.1 del documento (ISO/IEC/IEEE 26515, 2012), de acuerdo con los criterios definidos en Lucassen *et al*. (2015) y en North (2006).

1.7 Limitaciones

La investigación no contempló los siguientes casos:

- Historias de usuario que se encuentren escritas en otro lenguaje que no sea inglés.
- Plantillas de escritura para descripción y criterios de aceptación diferentes de los especificados en las secciones 1.1.5.1 y 1.1.5.3.
- Palabras con errores ortográficos; es decir, si algún verbo estaba mal escrito, no se consideraba como un verbo y, por ende, se registró un error.

- Dado que no existe análisis semántico, el etiquetado de las palabras está únicamente dado por el *Stanford POS Tagger* y, por ende, algunas palabras como *'requests'* pueden ser etiquetadas como un sustantivo en vez de un verbo y generar un error.
- El análisis de similitud semántica se realiza con *Wordnet;* no se utiliza ningún otro tipo de ontología.
- Sugerir soluciones y corregir errores en la herramienta automáticamente.

2 Marco conceptual

2.1 Fundamentos: calidad en las historias de usuario

Lucassen *et al*. (2015), en su *Quality User Story Framework* (QUS), proponen un conjunto de 14 criterios que influencian la calidad de una historia de usuario y los divide en tres grupos:

- Sintácticos: le concierne la estructura textual de una historia de usuario sin tomar en cuenta su significado.
- Semánticos: considera las relaciones y el significado del texto de la historia de usuario.
- Pragmáticos: acerca de escoger las alternativas más efectivas para comunicar un conjunto de requerimientos.

Esta investigación se enfoca en la calidad sintáctica y pragmática de las historias de usuario, pero considera las historias de usuario como un ente compuesto por una descripción, considerado por Lucassen *et al.* (2015) el "texto de la historia", un criterio de aceptación y un título.

Los tres, en conjunto, brindan la información completa que se necesita para comprender el valor detrás de cada requerimiento. Entonces, a los criterios de calidad propuestos se les extiende su definición y validación para tomar en cuenta al criterio de aceptación y el título.

2.1.1 Las historias de usuario como un todo

Cuando un dueño de producto presenta una historia de usuario a un equipo de desarrollo la expresa mediante su descripción:

As a <role> I want <goal>, so that I can <business value>.

En el idioma español esta forma se traduce como:

Yo como un <rol>, quiero <descripción de la funcionalidad>, de manera que pueda <valor de negocio>

Por sí mismo, el párrafo anterior constituye el núcleo de la historia de usuario; sin embargo, una vez entendido esto, el desarrollador debe saber cómo probar que el requerimiento se cumple y cuál es su valor dentro del engranaje del producto que está en desarrollo.

2.1.1.1 Criterio de aceptación

Una forma de especificar el criterio de aceptación es mediante el formato *"Given-When-Then"* (Dado-Cuando-Entonces), que permite establecer las precondiciones (*Given*), las acciones (*When*) y los resultados (*Then).* Fue propuesta por Dan North y es parte de la metodología *"Behaviour-driven development"* (BDD).

El BDD consiste en iniciar "desde afuera"; es decir, comenzar con los resultados de negocio esperados y, de ahí, definir el conjunto de funcionalidades requeridas para alcanzar dichos resultados. Un ejemplo de un criterio de aceptación en este formato es el siguiente:

> ***Given*** *my bank account is in credit, and I made no withdrawals recently,* ***when*** *I attempt to withdraw an amount less than my card's limit,* ***then*** *the withdrawal should complete without errors or warnings.*

La traducción al español corresponde a:

> Dado que mi cuenta bancaria es en crédito y no he realizado retiros recientemente, cuando intente hacer un retiro que es menor al límite de mi tarjeta, entonces el retiro debería completarse sin errores o advertencias.

Al interpretar el criterio de aceptación, el desarrollador logra establecer un escenario de prueba, una corrida a través del requerimiento y su resultado específico. Cabe destacar que un criterio de aceptación puede tener varios escenarios, como se muestra en el siguiente ejemplo:

Given *the account balance is \$100*
And the card is valid
And the machine contains enough money
When *the Account Holder requests \$20*
Then *the ATM should dispense \$20*
And the account balance should be \$80
And the card should be returned

Given *the account balance is \$10*
And the card is valid
And the machine contains enough money
When *the Account Holder requests \$20*
Then *the ATM should not dispense any money*
And the ATM should say there are insufficient funds
And the account balance should be \$20
And the card should be returned

2.1.1.2 Título

El título es otro componente textual de la historia de usuario que la ubica dentro de la lista de requerimientos de un producto. De acuerdo con North, el título siempre debe representar una actividad; por ejemplo: "Dueño de cuenta retira efectivo". De manera que es claro para el dueño de producto, clientes y desarrolladores que hasta que esta historia no se concluya, cada dueño de cuenta no podrá realizar retiros de efectivo.

Para reflejar el comportamiento de un usuario del sistema, el título debe poseer características sintácticas y pragmáticas específicas incorporadas a los criterios de calidad.

2.2 Procesamiento de lenguaje natural

El procesamiento de lenguaje natural o *Natural Language Processing (NLP)* es una rama de las ciencias de la computación y la inteligencia artificial que estudia la forma en que se aplica la computación en los diferentes matices de un lenguaje (Hardeniya, 2015).

En la práctica, *NLP* tiene que ver con crear sistemas que procesen o "entiendan" un lenguaje, de manera que se puedan ejecutar ciertas tareas (Deshapande, 2017), como, por ejemplo:

- Respuesta a preguntas.
- Análisis de sentimientos.
- Mapeo de imágenes a texto.
- Traducción de un texto a otro lenguaje.
- Reconocimiento de discurso.
- Etiquetado de partes de un discurso o *Part of Speech Tagging (POS Tagging).*
- Reconocimiento de nombres de entidades o *Name Entity Recognition.*

En esta investigación se utiliza el *POS Tagging* para identificar y etiquetar los diferentes componentes de la historia de usuario, pero antes es necesario aplicarle al texto algunos procesos característicos de *NLP* para prepararlo.

- Limpieza del texto: la idea detrás de este proceso es eliminar caracteres no deseados en el texto que puedan dificultar el análisis. Para este efecto se creó un método en cada analizador llamado *clean_component* que, mediante expresiones regulares, reemplaza los caracteres innecesarios con espacios en blanco.

- Separador de oraciones: el texto necesita ser separado en oraciones, pues cada componente está determinado por algunas palabras a su inicio. Una vez identificadas, se procede a separarlas en diferentes oraciones.
- *Tokenization:* un *token* es una palabra; la unidad mínima que una máquina puede entender y procesar (Hardeniya, 2015). En NLP, sin importar cuál tipo de aplicación se esté realizando, ningún texto puede ser procesado sin antes haber pasado por la fase de *tokenization*.

El proceso consiste en el texto en *tokens* significativos; un uso común en inglés es utilizar expresiones regulares para escoger palabras y números. El procesamiento natural de lenguaje que se realiza en AQUSA y AQUSA+ sigue la secuencia de la figura 7:

Figura 7. Procesamiento de lenguaje natural en AQUSA+.

2.2.1 Part of Speech Tagging

Una vez que se tiene el texto 'limpio', junto con las partes de cada uno de los componentes de la historia de usuario y con sus *tokens* debidamente identificados, se procede a etiquetar cada una de las palabras con la parte del discurso que representan. Por ejemplo, verbos, sustantivos, adverbios y artículos, entre otros. Este proceso se conoce como *Part of Speech Tagging* o *POS Tagging*.

Un algoritmo que permita hacer *POS Tagging* es llamado *tagger*. Por lo general, estos son algoritmos desarrollados con base en alguna técnica de aprendizaje autónomo (*Machine Learning*).

Como resultado del proceso, cada palabra pertenece a una tupla con su respectiva etiqueta. Por ejemplo, para el verbo *'is'*, una vez finalizado el *POS Tagging*, la tupla sería (*'is',VB)*, lo cual denota que la palabra es un verbo en su forma básica.

Ahora bien, para asignar una etiqueta, el algoritmo necesita una etiqueta correspondiente, una vez que identifica el *token*. Dicha etiqueta es dada por un sistema de notificación *POS* llamado *Penn Treebank*. Se trata de un esquema que realiza el mapeo de etiquetas a una parte del discurso específico. La tabla 1 muestra la totalidad de este esquema con la descripción de cada etiqueta en inglés y su traducción al español:

Etiqueta/*Tag*	**Descripción en inglés**	**Descripción en español**
NN	Noun, singular or mass	Sustantivo singular
NNS	Noun, plural	Sustantivo en plural
NNP	Proper noun, singular	Sustantivo propio, singular
NNPS	Proper noun, plural	Sustantivo propio, plural
DT	Determiner	Determinante
PDT	Pre determiner	Artículo, predeterminante

POS	Possessive ending	Sufijo posesivo
PRP	Personal pronoun	Pronombre personal
PRP$	Possessive pronoun	Pronombre posesivo
RB	Adverb	Adverbio
RBR	Adverb, comparative	Adverbio comparativo
RBS	Adverb, superlative	Adverbio superlativo
RP	Particle	Partícula
SYM	Symbol (mathematical or scientific)	Símbolo
TO	to	Palabra 'to'
IN	Preposition or subordination conjuction	Preposición o conjunción de subordinación
UH	Interjection	Interjección
VB	Verb, base form	Verbo, forma base
VBD	Verb, past tense	Verbo, pretérito
VBG	Verb, gerund/present participle	Verbo, gerundio
VBN	Verb, past	Verbo, pasado simple
WP	Wh-pronoun	Pronombre WH
WP$	Possessive wh-pronoun	Pronombre posesivo WH
WRB	Wh-adverb	Advervio WH
#	Pound sign	Numeral

$	Dollar sign	Signo de dólar
.	Sentence-final punctuation	Punto final
,	Comma	Coma
:	Colon, semi-colon	Dos puntos
(	Left bracket character	Abre paréntesis
)	Right bracket character	Cierra paréntesis
"	Straight double quote	Abre comillas
'	Left open single quote	Abre comilla simple
"	Left open double quote	Abre comilla doble
'	Right close single quote	Cierra comilla simple
"	Right open double quote	Cierra comilla doble

Tabla 1. Penn Treebank.

Entonces, cuando un *tagger* es aplicado sobre un texto específico como el siguiente:

"Well what do you think about the idea of, uh, kids having to do public service work for a year?"

Se genera un resultado similar al párrafo a continuación:

Well/UH what/WP do/VBP you/PRP think/VB about/IN the/DT idea/NN of/IN ,/,

uh/UH ,/, kids/NNS having/VBG to/TO do/VB public/JJ service/NN work/NN

for/IN a/DT year/NN ?/ ?

Como se observa, cada tupla es de la forma *token*/etiqueta. Ahora interesa describir cuál es el modelo para asociar una etiqueta a un *token* específico.

2.2.1.1 Stanford Tagger

El Stanford Tagger fue creado por Toutanova, Klein, Manning y Singer (2003) en el año 2003 y está basado en una técnica de clasificación de texto conocida como red de dependencias (*dependecy networks)*.

Anteriormente se mencionó que los *taggers* son construidos mediante técnicas de aprendizaje automático (*machine learning*). Cabe agregar que una de estas técnicas son los clasificadores bayesianos ingenuos. Estos clasificadores probabilísticos presuponen que cada característica o, en este caso, *token* es independiente de cualquier otro, lo cual suele ser anti-intuitivo en el caso del texto, pero, curiosamente, el algoritmo ha probado su eficacia en casos de uso reales. Una de sus características es que es fácil de implementar y calificar, solo es necesario almacenar las frecuencias y calcular las probabilidades. También es relativamente sencillo de entrenar (Hardeniya, 2015).

Los clasificadores bayesianos ingenuos son un caso especial de redes bayesianas, que son un modelo gráfico probabilístico que representa variables aleatorias y sus dependencias en un grafo dirigido acíclico. Al igual que los clasificadores ingenuos, las redes de dependencias no son un caso estándar de una red de Bayes, pues su grafo tiene ciclos, debido a que cada nodo representa una variable aleatoria con una probabilidad condicional: condicionada por las aristas que llegan a él (Toutanova *et al.*, 2003).

Esta modificación a la forma en que se realiza la clasificación del texto, le permitió a los creadores del Stanford Tagger una precisión del 97.24% en el *Penn Treebank* y es una de las principales razones por las que (Lucassen et al., 2015) lo utilizaron para AQUSA.

El *tagger* es un archivo de Java que contiene dos modelos entrenados para el idioma inglés. Se coloca en la raíz del repositorio de AQUSA y es importado al código fuente del proyecto como una biblioteca.

2.2.2 Análisis de la estructura del texto

El paso final antes de realizar el análisis que realiza AQUSA+ es generar el árbol de sintaxis (análisis sintáctico). Para esto es necesario realizar previamente un análisis de la estructura gramatical del texto. Para realizar esta tarea existen dos enfoques principales: el basado en reglas y el probabilístico (Hardeniya, 2015).

AQUSA+ utiliza el enfoque basado en reglas de la gramática del idioma inglés. Para esto necesita construir una gramática libre de contexto que sea codificada manualmente y un analizador de expresiones regulares. Por su parte, la gramática permite establecer un conjunto de reglas para que las palabras formen oraciones con sentido. Si bien es cierto, el análisis semántico requiere algoritmos más complejos, la validación de una estructura gramatical permite darle mayor pragmatismo a lo que está escrito.

Mediante la definición de una gramática libre de contexto, AQUSA+ es capaz de reconocer una cantidad amplia, pero finita, de oraciones que siguen el patrón de 'buena formación' definido por la gramática, diseñada para ofrecer al redactor suficiente flexibilidad para describir sus requerimientos como historias de usuario. Por último, cabe señalar que el analizador de expresiones regulares trabaja con la gramática sobre el texto previamente etiquetado para generar el árbol de análisis sintáctico.

La siguiente sección detalla cuál biblioteca y cuáles funciones se utilizaron para la implementación de AQUSA+.

2.2.3 Biblioteca NTLK de Python

NTLK es una de las bibliotecas más populares y usadas en la comunidad de procesamiento de lenguaje natural (*NLP*) debido a su simplicidad, que le permite a los desarrolladores implementar tareas complejas con pocas líneas de código.

En AQUSA+ se utilizan diferentes funciones de NTLK para el procesamiento y análisis del texto de cada historia de usuario:

- **Word_tokenize:** es un método genérico y más robusto que la función *split()* de Python, que permite generar *tokens* en cualquier cuerpo textual.
- **Metrics.distance.edit_distance**: es una función del módulo *distance* de NTLK que permite calcular la distancia de Levenshtein entre los dos textos; es decir, el número mínimo de ediciones de caracteres simples requeridos para cambiar una palabra hacia la otra.
- **RegexpParser(<Grammar>):** es el analizador de expresiones regulares de NTLK es utilizado en AQUSA junto con las gramáticas libres de contexto que se detallarán en la sección 3.1.4.3.1.

Por ejemplo, si la gramática que se estuviera utilizando fuera la siguiente:

NP: {<DT>? <JJ>* <NN>*}
P: {<IN>}
V: {<V.*>}
PP: {<P> <NP>}
VP: {<V> <NP|PP>*}

Se estarían definiendo patrones para la formación de distintas frases; o sea, la primera

NP: {<DT>? <JJ> <NN>*}* está expresando que un DT (artículo), seguido por un adjetivo (JJ) y un sustantivo (NN) forman una frase sustantiva (NP) (Hardeniya, 2015) .

Al final, el objetivo de esta función es retornar un tipo de dato *Tree* de Python, con la estructura gramatical del texto en análisis, por ejemplo, para la siguiente frase:

Mr. Obama played a big role in the Health insurance bill

El árbol de análisis sintáctico resultante al utilizar **NTLK.RegexpParser** se ilustra en la figura 8.

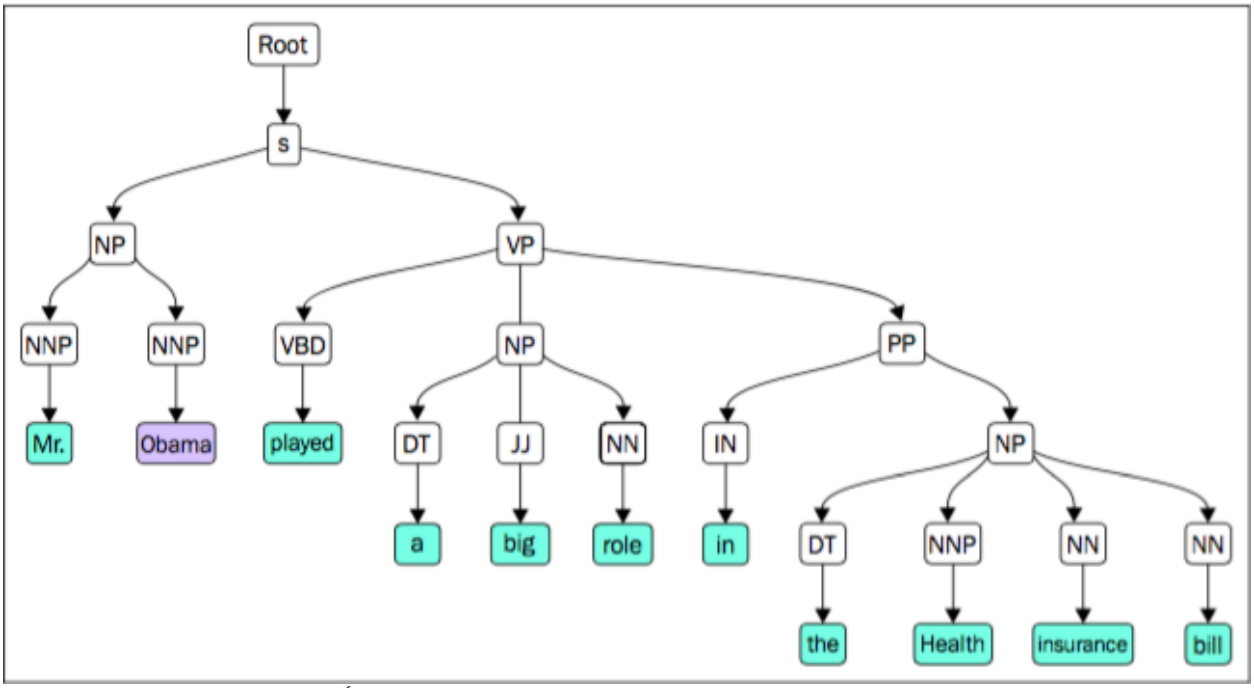

Figura 8. Árbol de Análisis Sintáctico (Hardeniya, 2015)

Por su parte, el contenido de la variable retornada por esta función sobre la cual itera AQUSA+ para realizar el análisis sintáctico y pragmático sería el siguiente:

Tree('S', [('Mr.', 'NNP'), ('Obama', 'NNP'), Tree('VP', [Tree('V',[('played', 'VBD')]), Tree('NP', [('a', 'DT'), ('big', 'JJ'), ('role','NN')])]), Tree('P', [('in', 'IN')]), ('Health', 'NNP'), Tree('NP',[('insurance', 'NN'), ('bill', 'NN')])])

Dado que el análisis de la gramática se realiza sobre el texto etiquetado, se preservan las tuplas dentro del árbol y cada no terminal es un subárbol que puede tener terminales y no terminales como nodos hijo.

2.2.4 Ontologías

Una de las características propuestas en Lucassen *et al*. (2015) para el modelo conceptual de AQUSA fue la presencia de un potenciador que mejorara las historias de usuario, al añadir sinónimos, homónimos y posible información semántica extraída de una ontología a las palabras más relevantes del texto.

Ciertamente, las ontologías son creadas para capturar conocimiento acerca de un dominio de interés. Una ontología describe conceptos en el dominio y, también, las relaciones entre ellos. Cabe agregar que una ontología tiene tres componentes principales:

- **Individuos o instancias**: representan objetos en el dominio de interés. Si se habla de un dominio de personas, Juan, María y Pedro son ejemplos de individuos.
- **Propiedades:** son relaciones binarias entre individuos que los conecta de alguna manera. Por ejemplo, una propiedad entre los individuos Juan y María podría ser "*es hijo de"* para denotar que Juan es hijo de María.
- **Clases:** son conjuntos de individuos y se caracterizan por utilizar descripciones formales en los requerimientos para pertenecer a ellas.

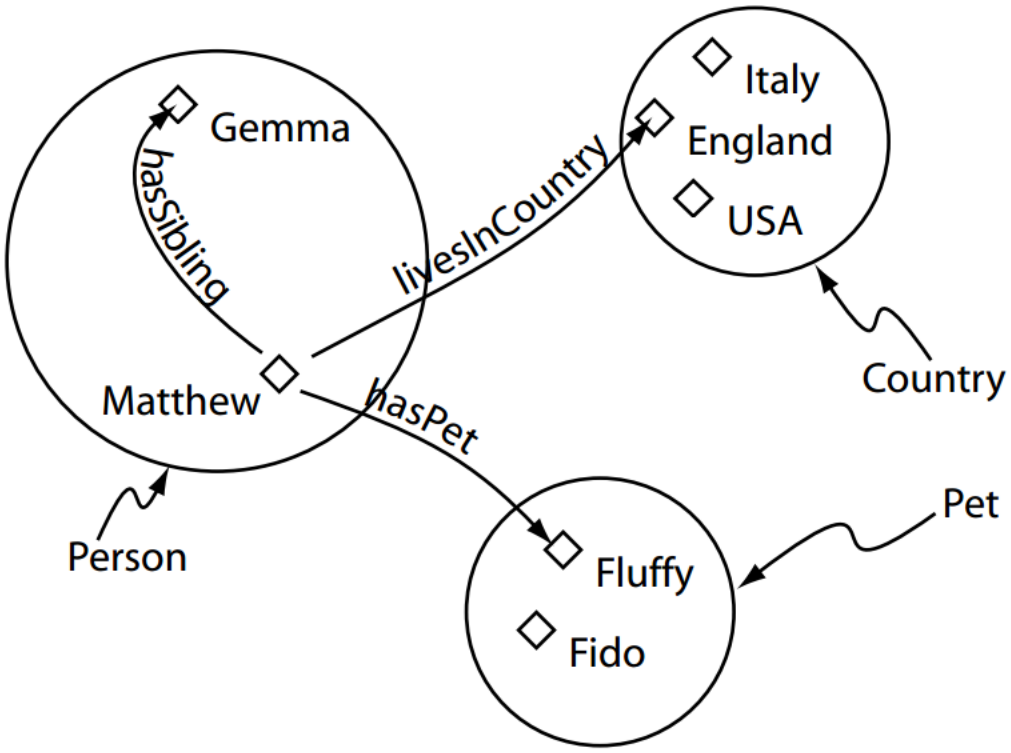

Figura 9. Ontología Simple.

La figura 9 muestra un ejemplo de una ontología simple, donde los rombos representan individuos que pertenecen a las clases (círculos) y que tienen relaciones o propiedades entre sí, inclusive, desde una clase hacia otra (representadas gráficamente por flechas).

Además, el prototipo no presenta ningún tipo de implementación, función o modulo para este potenciador, por lo que fue necesario realizar una indagación para determinar cuál era la ontología adecuada para el trabajo de investigación reportado en la presente tesis. Primero, se determinó una necesidad detrás del uso de ontologías en AQUSA+: identificar las palabras que denotaran una salida o un resultado observable en el criterio de aceptación. Para esta tarea se analizó el uso de *Protegé* y de *Wordnet*.

2.2.4.1 Protegé

Protegé es un sistema para desarrollar y mantener ontologías. Fue desarrollado por el Centro de Investigaciones Biomédicas de la universidad de Stanford. Para su prueba se descargó el *software* y se procedió a crear una ontología de ejemplo. Para esto, se analizó la posibilidad de construir una ontología con el dominio al que pertenecen las historias de usuario con que se probó AQUSA+. Sin embargo, esto limitaría AQUSA+ a un solo dominio y este no es el propósito de nuestra herramienta. Además, Protegé es un *software* robusto y las ontologías creadas ahí son usualmente utilizadas para el análisis semántico de un texto con respecto de un dominio específico, por lo que se descartó su uso para esta investigación.

2.2.4.2 Wordnet

Wordnet consiste en una ontología de dominio léxico; es decir, una gran base de datos con sustantivos, verbos y adverbios en inglés separados en conjuntos de sinónimos llamados *synsets*. Cada *synset* representa un concepto distinto y está interconectado por medios semántico-conceptuales y léxicos. *Wordnet*, entonces, resulta en una red de palabras relacionadas con sentido.

La característica anterior resulta adecuada para la regla *verifiable_output* de AQUSA+, donde se evalúa que las postcondiciones de un criterio de aceptación sean observables, ya sea en una pantalla, un reporte, el envío de un correo electrónico o el despliegue de un contenido específico, por ejemplo.

Al determinar que *Wordnet* es una ontología ya construida con las relaciones necesarias entre palabras en inglés basado en su sinonimia, surge como la mejor opción para validar el criterio ‘verificable’.

2.2.4.2.1 Similitud semántica en *Wordnet*

Al tener un conjunto de palabras que denotan resultados observables, es necesario establecer una regla para determinar si existen palabras similares en su sentido. Cada *synset* en Wordnet posee una función llamada *wu_similarity,* que recibe como parámetro otro *synset*. Está basado en la medida de similitud de Wu-Palmer (Wu y Palmer, 1994), que calcula el parentesco al considerar las profundidades de ambos *synsets* en la taxonomía de *Wordnet*, junto con la profundidad del ancestro menos común, conocido como *LCS.*

El resultado de esta función es siempre un número mayor a cero y menor o igual a 1, donde uno corresponde a la comparación de dos términos exactamente iguales. Este valor nunca puede ser cero, pues la profundidad del *LCS* nunca lo es.

Al conocer el valor retornado por la función *wu_similarity*, es necesario identificar cuál es un valor válido para decidir la similitud semántica entre dos palabras.

Madhuri, M.M Latesh (2014) realizan una comparación de la calificación del grado de similitud entre dos palabras aleatorias realizado por cinco expertos en el lenguaje inglés, contra el valor obtenido utilizando la medida Wu-Palmer.

La investigación obtuvo que por cada par de palabras comparadas utilizando Wu-Palmer con una similitud mayor al 34 % existía una correlación con la calificación otorgada por los lingüistas. Por ello, se utilizó dicho valor para determinar el umbral de similitud semántica aceptable para AQUSA+.

3 Marco metodológico

3.1 Ejecución de la metodología

3.1.1 Extracción de la información

La información utilizada para alimentar la herramienta corresponde a un conjunto de proyectos procesados por separado, cada uno en un documento, el cual posee un conjunto de historias de usuario para las que ese proyecto es padre.

Los proyectos pertenecen a una compañía de la industria de los conectores electrónicos, la cual desarrolla internamente su propio *software*, mediante un enfoque ágil, específicamente *Scrum*.

Para la administración del ciclo de vida del *software* utilizaron *Team Foundation Server* de Microsoft, el cual llama a las historias *Product Backlog Items (PBI)*. A la hora de extraer la información se creó un reporte que permite elegir el proyecto del cual se desea obtener las historias; los resultados fueron almacenados en un archivo de *Microsoft Excel* que, luego, fue convertido a formato de texto separado por comas (*CSV*).

Los campos pertinentes para la evaluación de cada historia de usuario son los siguientes:

- Título de la historia.
- Descripción.
- Criterio de aceptación.

Esta información permite elaborar la calificación de la deuda de documentación en que cada proyecto puede estar incurriendo.

3.1.2 Arquitectura del modelo

La arquitectura de AQUSA+ tiene como base la del modelo AQUSA, tal y como se mostró en la figura 6; con una extensión para el análisis del criterio de aceptación y el título, además de algunas modificaciones que se detallan en cajas con líneas punteadas.

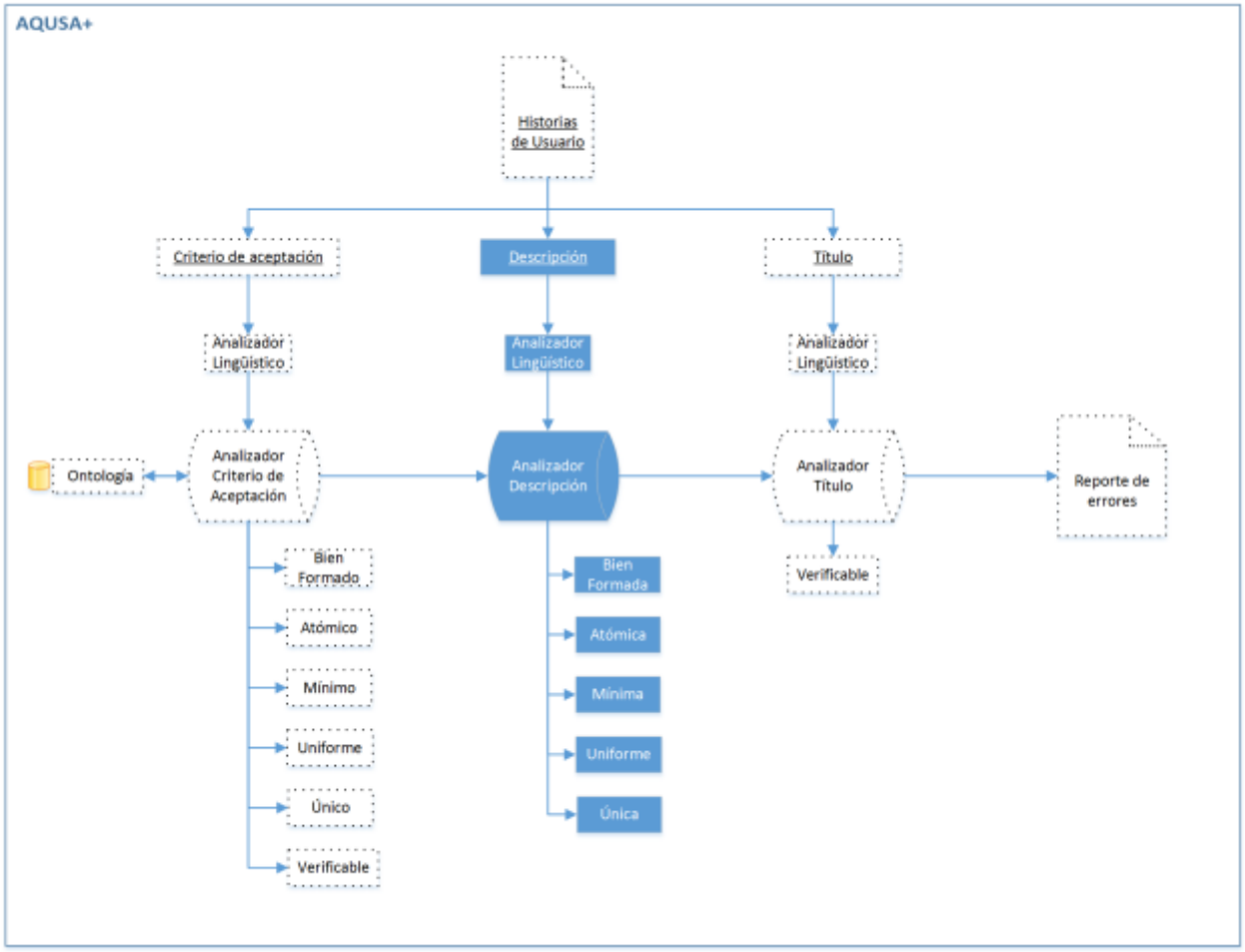

Figura 10. Modelo de arquitectura AQUSA+.

Los cambios en los componentes del modelo se describen brevemente a continuación, mientras que el detalle de su implementación se encuentra en la sección: 3.1.4.

- Archivo de carga de historias de usuario: se realizó la adición de las columnas de criterio de aceptación y título.
- Potenciador: el prototipo de AQUSA no implementa el potenciador, dado que no está conectado a ninguna ontología o diccionario. AQUSA+ realiza consultas a la ontología *Wordnet* a través del analizador de criterios de aceptación, específicamente para el criterio de calidad 'verificable'.
- Se agregaron los criterios de análisis para criterio de aceptación:
    - Bien formado.
    - Atómico.
    - Mínimo.
    - Uniforme.
    - Único.
    - Verificable.
- El criterio verificable está también basado en el análisis del título.
- Se crearon analizadores lingüísticos para el criterio de aceptación y el título, dado que su composición es distinta a la descripción.
- Se modificó el reporte de errores para incluir el análisis del título y el criterio de aceptación.

3.1.3 Extensión de los criterios de calidad del 'Quality User Story Framework'

El QUS presentado en Lucassen *et al*. (2015) establece una serie de criterios de calidad para historias de usuario basados, únicamente, en lo que en esta investigación se ha denominado

como ‘descripción’. En seguida se describen los criterios añadidos al modelo y las consideraciones tomadas para evaluar cada criterio en AQUSA+.

3.1.3.1 Criterios sintácticos

3.1.3.1.1 Atómico

En Lucassen *et al.* (2015) el criterio de atomicidad para la descripción de la historia de usuario se refiere a que una historia exprese el requerimiento para una funcionalidad, únicamente. Para este propósito, AQUSA cuenta con una regla llamada *‘atomic_conjuntions’,* la cual valida la presencia de conjunciones (‘And’, ‘Or’, &, +,’But’) en el texto y presenta el error en el reporte final.

Para el criterio de aceptación, la condición de atomicidad se evalúa de manera distinta; las conjunciones siguen siendo una causante de error, pero únicamente en el componente *When;* es decir, en el evento, el cual debe especificar únicamente una funcionalidad. Cabe señalar que las reglas agregadas en AQUSA+ para la evaluación de la atomicidad fueron denominadas *criteria_conjunctions* y *criteria_atomic_one_feature.*

3.1.3.1.1.1 Regla *criteria_conjuctions*

Evalúa que no existan conjunciones en el componente *When,* pero permite estas en los componentes *Given* y *Then*. El siguiente ejemplo ilustra un criterio de aceptación con conjunciones en los componentes permitidos:

Given *the account balance is \$100*
And *the card is valid*
And *the machine contains enough money*
When *the Account Holder requests \$20*
Then *the ATM should dispense \$20*

And *the account balance should be \$80*
And *the card should be returned*

Como se puede observar en el ejemplo, los componentes ***Given*** y ***Then*** poseen la conjunción '**And**', lo que es completamente válido; tanto las precondiciones como las postcondiciones pueden estar compuestas por varias oraciones, las cuales determinan en conjunto el estado previo del sistema y los cambios en él después del evento de transición.

Por su parte, un criterio de aceptación que incurre en una violación *'criteria_conjuctions'* es el que se muestra en el siguiente párrafo:

Given *the account balance is \$10*
And *the card is valid*
And *the machine contains enough money*
When *the Account Holder requests \$20* ***and*** *wants to check his balance*
Then *the ATM should not dispense any money*
And *the ATM should say there are insufficient funds*
And *the account balance should be \$20*
And *the card should be returned*

En este ejemplo el evento está compuesto por dos partes separadas por la conjunción '**and**', lo que indica que sería conveniente separar la historia de usuario en dos o más.

3.1.3.1.1.2 Regla *criteria_atomic_one_feature*

A diferencia de la descripción, el criterio de aceptación puede contener varios escenarios que establecen cómo la historia cumple con su objetivo. El siguiente conjunto de escenarios corresponde a un único criterio de aceptación:

Given *the account balance is \$100*
And the card is valid
And the machine contains enough money
When *the Account Holder requests \$20*
Then *the ATM should dispense \$20*
And the account balance should be \$80
And the card should be returned

Given *the account balance is \$10*
And the card is valid
And the machine contains enough money
When *the Account Holder requests \$20*
Then *the ATM should not dispense any money*
And the ATM should say there are insufficient funds
And the account balance should be \$20
And the card should be returned

Given *the card is disabled*
When *the Account Holder requests \$20*
Then *the ATM should retain the card*
And the ATM should say the card has been retained

El ejemplo anterior tiene tres oraciones destacadas, los componentes ***When*** o eventos de cada escenario. Como se observa, es el mismo evento a través de todo el criterio de aceptación. Por tanto, para que un criterio de aceptación se considere atómico, todos los escenarios deben girar en torno a un único evento, por lo que el siguiente ejemplo sugiere una violación a esta regla:

Given *the account balance is \$100*
And the card is valid
And the machine contains enough money
When *the Account Holder requests \$20*
Then *the ATM should dispense \$20*
And the account balance should be \$80
And the card should be returned

Given *the account balance is \$10*
And the card is valid
And the machine contains enough money
When *the Account Holder deposits \$100*
Then *the ATM should not dispense any money*
And the ATM should say there are insufficient funds
And the account balance should be \$20
And the card should be returned

Given *the card is disabled*
When *the Account Holder requests \$20*
Then *the ATM should retain the card*
And the ATM should say the card has been retained

3.1.3.1.2 Mínimo

El criterio de aceptación tiene tres componentes principales: precondiciones, evento y resultados. Cualquier información más allá de estos se considera innecesaria y una violación a la calidad de la historia de usuario como tal.

Para la validación de este criterio de calidad se crearon dos reglas heredadas y modificaron del análisis que ya AQUSA realizaba para adaptarlas al criterio de aceptación.

3.1.3.1.2.1 Regla *minimal_criteria_punctuation*

Esta regla valida que una vez identificados los tres componentes del criterio de aceptación, cualquier texto que proceda después de un signo de puntuación es extra y no corresponde a un componente esencial del criterio de aceptación o escenario.

El siguiente ejemplo muestra una violación al criterio 'Mínimo' de un criterio de aceptación:

> *Given I have make corrections to the invoice selected above, when I save my changes, then the invoice should be regenerated with the changes I made.* ***Ask accounting department for details on what to save.***

Como se observa, después del componente *Then* existe una oración que sucede al punto final del criterio de aceptación. Información como esta puede ser confusa para el desarrollador y llevarle a perder el entendimiento de lo que se busca. Por lo tanto, es una violación al criterio de calidad.

3.1.3.1.2.2 Regla *minimal_criteria_brackets*

El criterio de observación no debe incluir detalles en su escritura más allá de los tres componentes ya mencionados. Cuando se encierra información en corchetes o paréntesis en el

criterio de aceptación es un indicador de que esa información pertenece a otros artefactos del proceso de desarrollo como los casos de prueba. El siguiente criterio de aceptación refleja un caso de violación a esta regla:

> *Given an order 123 line 1 with a quantity of 10 and unit price of $0.1045, when the order has shipped then the calculation of 10*0.1045 needs to occur and be rounded to two decimals ($1.05 extended price)* ***(The extended price needs to be calculated by line)***.

Los datos encerrados en paréntesis en este criterio de aceptación no corresponden a ninguno de los componentes principales y deben ser almacenados, en caso de ser necesario, en alguna sección de notas, documento de diseño o casos de prueba.

3.1.3.1.3 Bien formada

Una historia de usuario de calidad debe contener un criterio de aceptación con sus tres componentes principales, en el caso de que alguno no se encuentre, el escenario no puede ser re-creado. Para este criterio, también se basó en la definición realizada en AQUSA y se adaptó al criterio de aceptación mediante cinco reglas.

3.1.3.1.3.1 Reglas *well_formed_no_when_comma y well_formed_no_then_comma*

Es una buena práctica delimitar cada componente del criterio de aceptación con una coma. La regla *well_formed_no when_comma* señala la falta de este delimitador entre las precondiciones (*Given)* y el evento (*When*).

La regla *well_formed_no then_comma*, por su parte, valida la existencia de la coma entre el evento y las postcondiciones *(Then)*.

Given the customer has a preference of which they want to ***receive their invoices when*** *and invoice is not sent to the customer in that* ***manner then AR should*** *be notified via email notification.*

El párrafo anterior ejemplifica un caso en el que la regla aplica y se debe reportar como un error de severidad leve, dado que es posible indicar los componentes a pesar de la falta de las comas en ambos casos.

3.1.3.1.3.2 Reglas *well_formed_no_given*, well_*formed_no_when y well_formed_no_then*

La ausencia de alguno de los tres principales componentes corresponde a un error de severidad alta, pues hace imposible analizar la calidad del criterio de aceptación en el componente faltante.

Los siguientes tres párrafos contienen errores de falta de precondiciones, evento y postcondiciones, respectivamente:

At the inquiry screen when viewing it then there should be an option to export the results grid to excel.

Given the count doesn't match then admin/finance support team should be notified.

Given finance master is open, when viewing finance ***master, I should be able*** *to see an operations tab with the taskbar on the left and the toolbar on the right.*

En los dos primeros párrafos AQUSA+ reporta los errores por la falta de un *Given* y un *When;* sin embargo, para el *Then* se flexibilizó la validación para aceptar el texto que se encuentre después de la primera coma después del *When.*

Lo anterior debido a que en la fase de recolección datos se identificó un patrón de escritura de parte de los dueños de producto en el que no necesariamente utilizan la palabra *'then',* aunque los textos posteriores al *When* tenían la misma función de representar las postcondiciones del criterio. Por lo tanto, si se identifica una coma después del *When,* se toma como las postcondiciones del escenario y se reporta la falta de la palabra. En caso contrario, solo se reporta el error.

3.1.3.2 Criterios pragmáticos

3.1.3.2.1 Uniforme

Busca que el criterio de aceptación siga una plantilla de escritura *"Given/When/Then"*, de manera que concuerde con el resto de historias y se valida su composición de acuerdo con dicha plantilla. Sin embargo, el sistema debe ser capaz de aceptar ciertas variaciones en la escritura, siempre y cuando se identifiquen los tres principales componentes: precondiciones, evento y post-condiciones.

Para que un criterio de aceptación sea uniforme, al igual que la historia de usuario, debe seguir el mismo formato que las demás; esta validación es la misma realizada en AQUSA, pero modificada para los componentes del criterio de aceptación.

"Given the part selected is in my tasks queue, when saving as complete, remove the part from my tasks queue."

En el ejemplo anterior esta regla mostraría que el criterio de aceptación no es uniforme, debido a que falta la palabra *'then'*. Sin embargo, a diferencia de la validación en 'Bien Formado', esta se realiza en comparación con el resto de historias. Es decir; si la mayoría de los criterios de aceptación del proyecto no usan la palabra '*then'* para determinar las post-condiciones el criterio, sería uniforme.

3.1.3.2.2 Único

Cada criterio de aceptación debe ser único, en el entendido de que proveen un escenario diferenciado del resto de las historias de usuario. En Lucassen *et al*. (2015) se plantea la validación de esta regla mediante la función de similitud semántica entre los objetos y verbos presentes utilizando *Wordnet.* Si la combinación de los cálculos otorga un nivel de similitud semántica mayor al 90 % se dice que dos descripciones de la historia de usuario son idénticas.

Sin embargo, en la implementación realizada en AQUSA se hace una comparación simple entre los textos de cada descripción. Por lo tanto, en el criterio de aceptación se realiza el mismo tipo de comparación.

3.1.3.2.3 Verificable

El formato para el criterio de aceptación *("Given/When/Then")* puede ser visto como una máquina de estados finitos que determina un escenario dado (Martin, 2008).

La existencia de un escenario en su criterio de aceptación permite que una historia se considere verificable. También, la forma en que su título es escrito influye en esta característica. Por ende, para validar que una historia puede ser verificable, se analizan los componentes de cada escenario que conforma el criterio de aceptación a través de las siguientes reglas.

3.1.3.2.3.1 Regla *verifiable_user_interaction*

Las precondiciones (*Givens)* tienen como propósito poner al sistema en un estado conocido antes de que el usuario o agente externo comience a interactuar con él. Esta regla valida que no existan pronombres o sustantivos propios en las precondiciones, por lo que un escenario como el siguiente registraría un error en AQUSA+:

> *Given **I** select an invoice from the AR application, when I select it from the grid, then I should be able to open it in "edit mode" for saving corrections to it.*

Como se aprecia en el ejemplo, existe interacción de usuario en cada uno de los componentes del escenario; sin embargo, solo se evalúa en las precondiciones.

3.1.3.2.3.2 Regla *verifiable_stative_verb*

De acuerdo con las precondiciones y su condición de representar un estado, es necesario que el verbo sea de estado o, como se conoce en inglés, *stative.*

El siguiente ejemplo es una clara muestra de un estado representado en las precondiciones:

"*Given the card **is** disabled, when the Account Holder requests $20, then the ATM should retain the card"*

Según el ejemplo anterior, el verbo *"is"* es un verbo de estado que define el estado previo a la acción de transición que representa el *When*. Esta regla no solo valida el tipo de verbo, sino su ausencia. Otro caso que representaría una violación es el siguiente:

***Given a customer**, when calling CS for an order and they want a proforma, then the system should be able to provide it.*

3.1.3.2.3.3 Regla *verifiable_dynamic_verb*

El componente *When* representa la acción o el evento sobre el que gira el escenario o escenarios del criterio de aceptación. Esta regla verifica la existencia de un verbo dinámico en el *When*. Los verbos permitidos corresponden al gerundio; es decir, terminaciones *"ing"* o bien, la forma del verbo en presente.

El siguiente párrafo es un ejemplo de una violación a la regla, pues la palabra ***requested*** es un verbo en pasado.

Given the card is disabled, when the Account Holder requested $20, then the ATM should retain the card.

Por su parte, un correcto uso del verbo dinámico en el *When* sería el siguiente:

*Given the card is disabled, when **requesting** $20, then the ATM should retain the card.*

3.1.3.2.3.4 Regla *verifiable_output*

La observación de los resultados o post-condiciones debe estar relacionada con el valor de negocio esperado. Entonces, este debe ser representado por alguna salida o *'output'*, como un

reporte, mensaje, pantalla o similar. Esta regla se implementó en AQUSA+ mediante la utilización de la ontología *Wordnet* y su método de similitud semántica, al comparar cada sustantivo del *Then* contra una serie de términos que tienen relación con los resultados esperados. Si al realizar la comparación con cada uno de los términos de la lista ninguno alcanza una similitud semántica mayor al 34 %, se dispara una advertencia sobre la falta de un resultado demostrable en el escenario.

3.1.3.2.3.5 Regla *verifiable_title*

Dan North establece que el título es también una parte importante para determinar la verificabilidad de una historia de usuario, pues debe describir una actividad. También, debe especificar claramente cuál es la funcionalidad que se perdería si la historia de usuario no se llega a completar, por ejemplo: “Cliente retira efectivo del cajero automático”.

Si el título se escribiera de otra forma, como: “Administración de cuenta del cliente”, sería necesario indagar en los componentes de la historia de usuario para entender a qué se refiere específicamente.

3.1.4 Implementación de la metodología

Las modificaciones de AQUSA para incorporar las nuevas funcionalidades que lo convierten en AQUSA+ comprendieron varios retos de diseño y programación que serán descritos con detalle en la presente sección. Se inicia con el trabajo para configurar el ambiente de desarrollo y adaptar la herramienta a un archivo con tres columnas en vez de una, desde la captura de los datos, la partición del criterio de aceptación en sus tres componentes y la forma como estos son almacenados en la base de datos.

Posteriormente se explica la forma cómo se añadieron las nuevas reglas de AQUSA+, los valores evaluados para cada una, los mensajes de error y la forma en la que estos fueron presentados en la interfaz de la herramienta.

3.1.4.1 Configuración del ambiente de desarrollo

El repositorio de AQUSA se encuentra en línea, en la dirección: https://github.com/gglucass/AQUSA, donde puede ser descargado y clonado, por lo que se procedió a realizar lo anterior y, posteriormente, a realizar un *'branch'* del repositorio *'master'* llamado *'develop'*.

AQUSA se desarrolló con Python 3.4. Conviene señalar que para AQUSA+ se utilizó Python 3.5 y también se actualizó la versión de *Flask* a 0.10.1, que es una plataforma de desarrollo web para Python.

Para el procesamiento de lenguaje natural se utiliza la biblioteca NTLK, la cual fue actualizada de la versión 3.0.2 a la 3.1.

La siguiente tabla presenta el resto de bibliotecas necesarias para programar y ejecutar AQUSA+:

Biblioteca	**Versión**	**Descripción**
alembic	0.7.5	Migración de datos
altgraph	0.10.2	Construcción de gráficos
Babel	1.3	Aplicación de localización para Python
bdist-mpkg	0.5.0	Manejador de paquetes
bonjour-py	0.3	Interface con Apple bonjour
Flask-Babel	0.9	Formato de fechas
Flask-Bootstrap	3.3.2.1	Construcción de entorno para Flask
Flask-Login	0.2.11	Manejador de inicio de sesión de Flask
Flask-Migrate	1.3.1	Manejador de base de datos
Flask-RESTful	0.3.2	Permite crear API REST

Flask-Script	2.0.5	Creación de scripts externos
Flask-SQLAlchemy	2	Soporte de SQL en Flask
Jinja2	2.7.3	Motor de plantillas
jsonrpc	1.2	Protocolo de llamados de procedimientos remotos
Mako	1.0.1	Biblioteca de plantillas
MarkupSafe	0.23	Escape de caracteres automático
matplotlib	1.3.1	Ploteo 2D
modulegraph	0.10.4	Análisis de dependencia de módulos
NTLK	3.1	Procesamiento de lenguaje natural
nose	1.3.6	Pruebas
numpy	1.10.1	Computación científica
pandas	0.16.0	Análisis de estructuras de datos
pbr	0.10.8	Herramienta para configuración de ambiente
pep8	1.6.2	Guía de estilos
pexpect	3.3	Controlador de aplicaciones hijas
Pillow	2.7.0	Librería de manejo de imágenes
psycopg2	2.6	Adaptador de PostgreSQL
py2app	0.7.3	Comandos de configuración
SQLAlchemy	0.9.9	Herramientas de SQL para Python
sqlalchemy-migrate	0.9.5	Versionamiento del esquema de base de datos
sqlparse	0.1.14	Analizador de sentencias SQL
Tempita	0.5.2	Plantillas de substitución de texto
Unidecode	0.4.17	Traducción Unicode - ASCII
virtualenv	12.0.7	Ambientes aislados de Python
Werkzeug	0.10.4	Utilitario de desarrollo web

Tabla 2. Paquetes de Python utilizados en AQUSA+.

Posteriormente se siguieron las instrucciones para configurar el ambiente: se creó una base de datos vacía utilizando Postgres 9.4.5.0, se modificó la variable de ambiente $DATABASE_URL para apuntar a esta base de datos y se ejecutaron los siguientes comandos que crean las tablas definidas para las historias de usuario:

- . /manage.py db init

- ./manage.py db migrate
- ./manage.py db upgrade

Finalmente es necesario crear un folder en la raíz del proyecto AQUSA con el nombre Stanford, en el cual se deben almacenar los archivos:

- stanford-postagger-withModel.jar: Analizador de las partes del discurso de Stanford.
- english-left3words-distsim.tagger: Modelo de Stanford para el análisis de discurso en inglés.

Las modificaciones a la base de datos para almacenar los datos del criterio de aceptación y el título se detallan en la siguiente sección.

3.1.4.2 Modificación al diseño de base de datos

Para el procesamiento del criterio de aceptación y el título fue necesario agregar un total de cuatro tablas al modelo de base de datos de AQUSA; todas se encuentran relacionadas con la tabla *project* mediante la tabla *story,* que corresponde a la descripción de la historia de usuario.

Cabe destacar que AQUSA+ es una extensión al prototipo de AQUSA, por lo que no se revisó el modelo de datos y se procedió a trabajar con el modelo heredado.

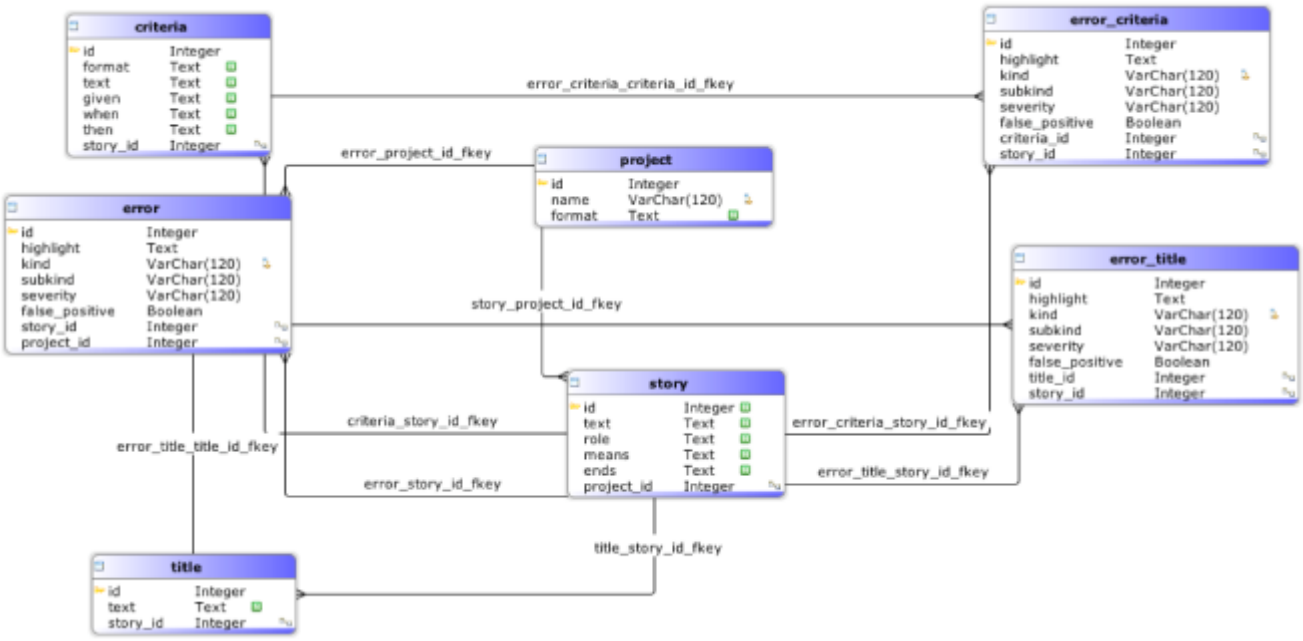

Figura 11. Modelo ER - AQUSA+.

- **Criteria:** se utiliza para almacenar un escenario del criterio de aceptación de cada historia de usuario, por lo que la relación es uno a muchos con la tabla *story.*
    - *id:* identificador de escenario.
    - *format:* atributo que almacena el formato al que debe adherirse cada criterio de aceptación. En este caso almacena (Given, When, Then).
    - *text:* texto del escenario.
    - *given*: texto del *given* una vez realizado el *chunking*.
    - *when*: texto del *when* una vez realizado el *chunking*.
    - *then*: texto del *then* una vez realizado el *chunking*.
    - *story_id:* llave foránea a la tabla *story.*
- **Title:** tabla para almacenar el texto del título y su relación con la descripción y el proyecto al que pertenece.
    - id**:** identificador.

    - *text:* texto del título.
    - *story_id:* llave foránea a la tabla *story.*
- **error_title:** almacena la lista de errores ligados a un título del:
    - *id:* identificador del error.
    - *highlight:* texto a resaltar.
    - *kind: title*
    - *subkind: title*
    - *severity:* severidad del error.
    - *false_positive:* valor binario para determinar si el error es un falso positivo.
    - *criteria_id:* llave foránea a la tabla *criteria.*
    - *story_id:* llave foránea a la tabla *story.*
- **error_criteria:** almacena la lista de errores ligados a un escenario del criterio de aceptación.
    - *id:* identificador del error.
    - *highlight:* texto a resaltar.
    - *kind: given, when o then.*
    - *subkind:* regla del criterio de calidad.
    - *severity:* severidad del error.
    - *false_positive:* valor binario para determinar si el error es un falso positivo.
    - *title_id:* llave foránea a la tabla *title.*
    - *story_id:* llave foránea a la tabla *story.*

La tabla *project* es la principal del proyecto. Por esta razón existe en *story* una llave foránea de *project_id;* es decir, un proyecto puede tener una o más historias de usuario. Por su parte, el resto de atributos de *story* son los mismos que se crearon en AQUSA, *role*, *means* y *ends*. Además, la tabla *error* se sigue utilizando, pero únicamente para los errores de la descripción.

3.1.4.3 Procesamiento de la información

Por un lado, AQUSA procesaba un archivo .csv de una sola columna con la descripción de la historia de usuario. Por otro, AQUSA+ añade la lógica necesaria para recibir archivos del mismo formato, pero de tres columnas en el siguiente orden: descripción, criterio de aceptación, título. Para leer el archivo se utiliza la biblioteca *pandas*, la cual ya era utilizada, pero únicamente en la columna cero. Para obtener el criterio de aceptación se hace un recorrido de la fila y se evalúa la posición 1 y, en caso de que el criterio de aceptación tenga múltiples escenarios, estos se almacenan por separado. Sin embargo, su llave foránea será el identificador de la historia, lo que permite saber cuáles escenarios pertenecen a una determinada historia para el posterior análisis.

AQUSA+ recorre el criterio de aceptación y cada vez que encuentra la palabra *Given* almacena el texto que le procede, hasta que encuentre fin de línea o bien, otro *Given*, correspondiente al inicio de un escenario nuevo.

En el caso del título, se evalúa la posición 2 de la fila que se está recorriendo y se liga su identificador al de la historia de usuario para ser almacenado en la base de datos.

3.1.4.3.1 Gramática utilizada

La gramática diseñada para el criterio de aceptación está basada en la utilizada para la descripción en AQUSA:

```
NP: {<DT|JJ|NN.*>}
NNP: {<NNP.*>}
AP: {<RB.*|JJ.*>}
VP: {<VB.*><NP>*}
MEANS: {<AP>?<VP>}
ENDS: {<AP>?<VP>}
```

En este caso, la composición gramatical de una descripción tiene dos oraciones a las que se les llama: *means* (medios) y *ends* (fin); es decir, los medios para un fin determinado. Estas oraciones están determinadas por la expresión regular {<AP>?<VP>}, que quiere decir que están compuestas por la presencia de una frase adjetiva (AP) de forma opcional y de una frase verbal (VP).

Una frase adjetiva tiene en su composición, al menos, un adverbio (RB) o un adjetivo (JJ). Por su parte, una frase verbal tiene, como mínimo, un verbo (VB) y una frase sustantiva (NP). Las frases sustantivas en AQUSA tienen un determinante (DT), un adverbio (JJ) o bien, un sustantivo en singular o plural (NN). Entonces, dada la amplitud de oraciones que puede recibir esta gramática, se decidió utilizar también para el criterio de aceptación y el título, creando una para cada uno:

NP: {<DT|JJ|NN.*>}
NNP: {<NNP.*>}
AP: {<RB.*|JJ.*>}
VP: {<VB.*><NP>*}
GIVEN: {<AP>?<VP>}
WHEN: {<AP>?<VP>}
THEN: {<AP>?<VP>}

Como se observa, se sustituye *Means* por *Given*, *Ends* por *When* y se crea una nueva entrada para el *Then* con la misma composición.

Para el título se crea también una gramática por separado, con la frase *Title, tal y* como se muestra a continuación:

NP: {<DT|JJ|NN.*>}
NNP: {<NNP.*>}
AP: {<RB.*|JJ.*>}
VP: {<VB.*><NP>*}
TITLE: {<AP>?<VP>}

Cada etiqueta se asigna utilizando el *Stanford POS Tagger,* mediante un proceso llamado *chunking,* el cual se realiza durante el análisis del texto en cada una de las reglas sintácticas y pragmáticas.

3.1.4.3.2 Chunking

Para analizar cada componente por aparte, se crearon clases separadas para cada uno. AQUSA tenía una clase analizadora por lo que se crearon clases analizadoras diferenciadas para el criterio de aceptación y el título.

Para el criterio de aceptación, se modificó el método de fragmentación conocido como *chunking*, a fin de partir el texto en los tres componentes *Given, When y Then*. El proceso de fragmentación es común a todas las reglas en la descripción y el criterio de aceptación. En el caso de este último, cuando es almacenado, es necesario identificar cada uno de sus componentes y etiquetar las palabras que se encuentran ahí, para analizarlas posteriormente dentro del contexto de ese específico fragmento.

Con el objeto de realizar el proceso anterior, se utiliza el *Stanford POS Tagger* y los siguientes son los pasos a seguir para obtener cada componente etiquetado:

- Se analiza la estructura gramatical del componente (*Given, When* o *Then*), según la regla que se vaya a aplicar; por ejemplo, si se evalúa atomicidad, el componente en cuestión sería el *When.* Esto permite identificar qué tipos de frase contiene el texto.
- Se eliminan los indicadores; es decir, las palabras *Given, When o Then,* para que la fragmentación solo se haga sobre el resto del texto.
- Se realiza el etiquetado de las palabras mediante el método de NTLK, llamado *tokenize*, el cual devuelve una lista de tuplas con la palabra y su respectiva etiqueta. Por ejemplo, para el verbo *is* la tupla sería (is, VB).

- Finalmente se analiza la oración utilizando NTLK.RegExpParser con la gramática para el criterio de aceptación, cuyo propósito es identificar las relaciones entre las palabras que construyen las frases establecidas en la gramática.

El último paso hace que se devuelva un árbol (NTLK *Tree*) con la estructura gramatical jerárquica de la oración. Este árbol es recorrido en cada método para las reglas de validación al analizar las tuplas contenidas.

3.1.4.4 Validación de reglas sintácticas

La presente sección detalla la forma en que los criterios de calidad *atómico* y *mínimo* son implementados dentro de AQUSA+. Específicamente, para el criterio de aceptación se realiza una comparación con la implementación original en AQUSA y se proveen algunos ejemplos.

3.1.4.4.1 Atómico

Como se mencionó anteriormente, a diferencia del análisis de la descripción, en el análisis del criterio de aceptación existen dos reglas para la validación de esta condición de calidad, las cuales se implementaron por separado como métodos de la clase *AnalyzerCriteria.*

3.1.4.4.1.1 Regla *criteria_conjunctions*

Para esta regla se utiliza el mismo arreglo de conjunciones no permitidas de AQUSA, pero se agrega la palabra '*but'* (pero), la cual es una conjunción que no se había tomado en cuenta originalmente en AQUSA.

La lista de conjunciones no permitidas es la siguiente:

```
CONJUNCTIONS = [' and ', '&', '+', ' or ', ' but ']
```

El llamado a este método se realiza utilizando la siguiente instrucción:

AnalyzerCriteria.atomic_rule(getattr(criteria,chunk), chunk)

Así, al evaluar un criterio de aceptación en cuanto a su atomicidad, se trabaja únicamente sobre el *When;* es decir *chunk* debe ser 'when'.

La siguiente porción de código recorre la lista de conjunciones inválidas y evalúa si alguna está presente en el *When* y, de ser así, etiqueta esa oración como inválida.

```
def atomic_rule(chunk, kind):
  sentences_invalid = []
  if chunk:
    for x in CONJUNCTIONS:
      if x in chunk.lower():
        if kind == 'when':
          for when in chunk.split(x):
            sentences_invalid.append(False)
  return sentences_invalid.count(False) > 1
```

En vista de que un criterio de aceptación presenta varios escenarios, puede tener varias oraciones *When* también, y cada una debe ser reportada por separado en caso de presentar algún error.

La siguiente imagen presenta la forma que se presenta un error *criteria_conjunctions* en el reporte final de AQUSA+:

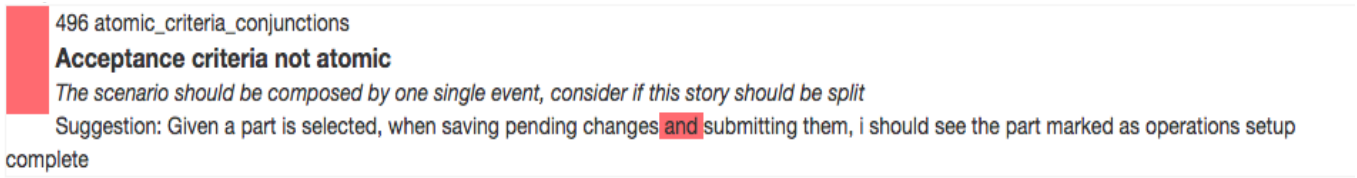

Figura 12. Regla criteria_conjunctions.

Como se observa, el error tiene un identificador único que es almacenado en la tabla *error_criteria* que, al ser reportado, presenta el escenario en el que fue encontrado y señala en rojo la conjunción que debe ser eliminada a manera de sugerencia. El uso del color rojo implica una severidad alta, debido a que la historia debe ser dividida en dos y se corre el riesgo de que la documentación apropiada para dos o más funcionalidades no sea registrada.

3.1.4.4.1.2 Regla *atomic_one_feature*

Con el propósito de determinar si un criterio de aceptación es atómico cuando posee diferentes escenarios, es necesario evaluar que su evento *When* sea el mismo para cada uno de ellos. Criterios de aceptación de un solo escenario nunca presentan este error.

La siguiente porción de código detalla la implementación de esta regla al recorrer todos los eventos de cada uno de los escenarios:

```
def atomic_one_feature_rule(criteria):
  stories = Criteria.query.filter_by(story_id=criteria.story_id).all()
  repeated_when = []
  if len(stories) > 1:
    for x in stories:
      repeated_when.append(x.when == criteria.when)
    return repeated_when.count(True) != len(stories)
  else:
    return False
```

Cada evento recorrido es comparado contra el anterior y almacenado en el arreglo *repeated_when.* Al final del recorrido la cantidad de eventos repetidos y la cantidad de escenarios en el criterio de aceptación *len(stories)* debe ser la misma, de lo contrario, se reporta el error como se muestra en la siguiente figura:

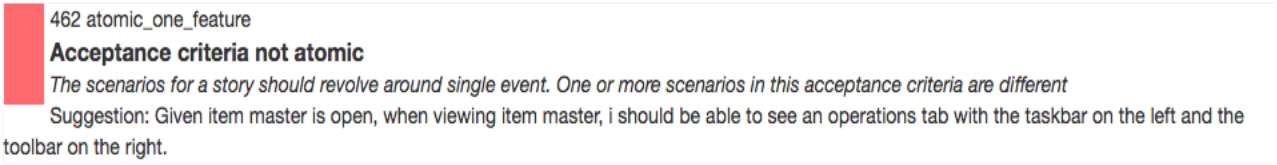

Figura 13. Regla criteria_atomic_one_feature.

El mensaje de error especifica que para que el criterio de aceptación sea atómico, este debe girar alrededor de un solo evento. Utiliza el color rojo para indicar severidad alta y, aunque presenta el campo de sugerencia *Suggestion* para este criterio de calidad, no se implementó un texto de sugerencia debido a que escoger el evento adecuado es una labor del experto.

3.1.4.4.2 Mínimo

Para validar si un criterio de aceptación cumple la característica de ser mínimo, se utiliza la misma lógica de AQUSA, que utiliza una clase llamada *MinimalAnalyzer*.

Cabe destacar que para el criterio de aceptación se creó una nueva clase llamada *MinimalAnalyzerCriteria,* la cual posee dos métodos detallados a continuación.

3.1.4.4.2.1 Regla *criteria_minimal_punctuation*

Esta regla utiliza una lista de signos de puntuación que, una vez identificados los componentes del criterio de aceptación, evalúa la lista está dada por la siguiente variable:

PUNCTUATION = ['.', ';', ':', '–', '–', '—', '—', '-', '-', '?', '*']

Al escanear el texto y encontrar información posterior a alguno de estos signos, el método registra el error, tal y como se muestra en la siguiente figura:

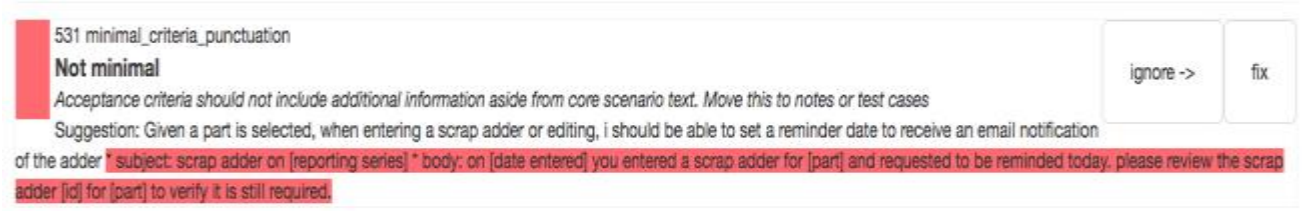

Figura 14. Regla minimal_criteria_punctuation.

Lo anterior indica que no debe incluirse más información que la necesaria para detallar el escenario y señala en rojo el texto que debe eliminarse o moverse a alguna sección de notas o casos de prueba.

3.1.4.4.2.2 Regla *minimal_criteria_brackets*

Al igual que en la regla anterior, existe una lista de caracteres sobre la cual se itera mientras se recorre el texto del criterio de aceptación, dada por la siguiente variable:

BRACKETS = [['(', ')'], ['[', ']'], ['{', '}'], ['⟨', '⟩']]

Entonces, al recorrer el texto del criterio de aceptación a través de sus tres componentes, si encuentra alguno de los caracteres de la lista *Brackets* registra el siguiente error:

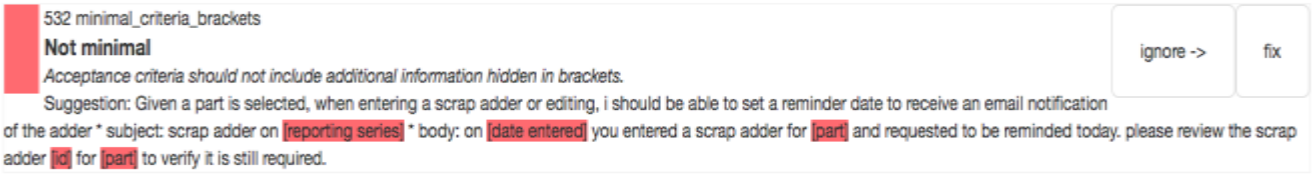

Figura 15. Regla minimal_criteria_brackets.

En el detalle del error se explica que no se debe presentar información entre paréntesis, llaves o corchetes y señala en rojo las partes del texto que cumplen con esta característica sugiriendo su eliminación.

3.1.4.4.3 Bien formado

Para validar las cinco reglas de esta categoría, se utiliza una lógica similar a la de AQUSA, pero adaptada a los componentes del criterio de aceptación. Se acude al mismo método de la descripción, pues este es agnóstico respecto al texto, siempre y cuando pueda encontrar el segmento adecuado; es decir, en vez de utilizar *means* o *ends* se utiliza *when* y *then*.

3.1.4.4.4 Reglas *well_formed_no_when_comma* y *well_formed_no_then_comma*

Para identificar si existe una coma entre el *Given* y el *When* o bien, entre el anterior y el *Then*, se hace un llamado con el segmento del texto que se quiere evaluar:

1. *AnalyzerCriteria.well_formed_content_rule(criteria.when, "when", ["when"])*
2. *AnalyzerCriteria.well_formed_content_rule(criteria.then, "then", ["then"])*

En el punto 1 se observa el llamado para evaluar el segmento de texto que contiene al *when* y determinar si hace falta una coma entre el *Given* y el *When*. Por su parte, en el punto 2 se hace la misma evaluación entre el *When* y el *Then.*

La siguiente figura muestra cómo se reporta esta regla en AQUSA+ para ambos casos:

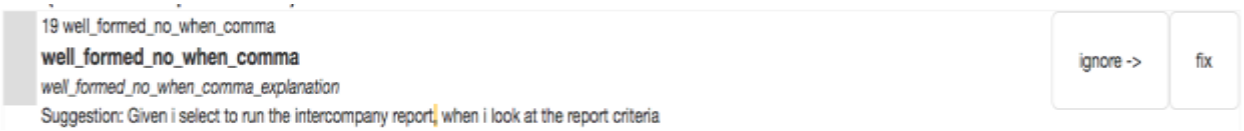

Figura 16. Regla well_formed_no_when_comma

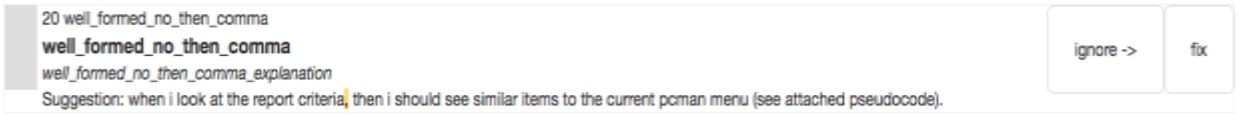

Figura 17. Regla well_formed_no_then_comma.

El color de la barra a la izquierda es gris para reflejar que es una severidad leve y se brinda la sugerencia de agregar la coma en el lugar adecuado.

3.1.4.4.4.1 Regla *well_formed_no_given, well_formed_no_when y well_formed_no_then*

Esta regla se evalúa al revisar si cada uno de los segmentos de los escenarios del criterio de aceptación se encuentra almacenado en la base de datos. Cuando el proceso de segmentación no es capaz de identificar ya sea el *Given*, el *When* o el *Then*, almacena un valor nulo que luego es capturado por este método. Por ejemplo, para el siguiente párrafo se obtienen como resultado los errores en la figura 18:

Since I'm entering a new customer the ERS option should be displayed in customer entry.

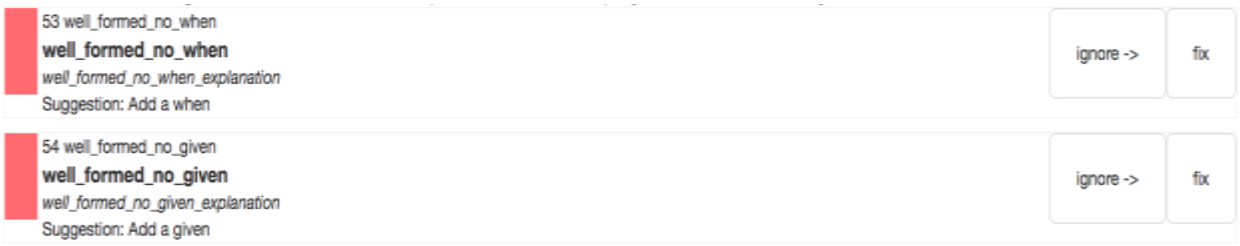

Figura 18. Regla well_formed_no_given y well_formed_no_when.

Acá no se pudo identificar ni el *Given* ni el *When;* en estos casos, al igual que en la total ausencia del *Then*, las reglas asociadas a estos segmentos lanzan errores también. La severidad es alta, por lo que se utiliza el color rojo en la barra del lado izquierdo.

3.1.4.5 Validación de reglas pragmáticas

3.1.4.5.1 Uniforme

Para la implementación de la regla de uniformidad en el criterio de aceptación se utilizó la misma lógica deAQUSA, migrándola hacia la nueva clase analizadora.

En el caso del criterio de aceptación, se crea un atributo en su clase llamado *format*, el cual almacena el formato establecido mediante la siguiente línea:

```
format = db.Column(db.Text, nullable=True, default="Given,When,Then")
```

Después, en el método para validar la regla se utiliza la función *edit_distance* de NTLK para calcular la distancia de Levenshtein entre los dos textos; es decir, el número mínimo de ediciones de caracteres simples requeridos para cambiar de una palabra hacia la otra.

```
if NTLK.metrics.distance.edit_distance(chunks[x].lower(), project_format[x].lower()) > 3:
  result = True
```

La instrucción anterior compara la primera palabra de cada uno de los segmentos contra el formato establecido:

Given I select an invoice from the AR application, when I select it from the grid, I should be able to open it in "edit mode" for saving corrections to it

Entonces, por ejemplo, si el criterio de aceptación es el anterior, se realiza una comparación entre *Given, when, I should* y *Given, When, Then* y, al existir una distancia mayor a 3, se reporta el error.

El ejemplo anterior funciona para mostrar una modificación realizada a AQUSA+ para flexibilizar el análisis del criterio de aceptación.

Al utilizar diferentes archivos con historias de usuario de proyectos reales se notó un patrón en el que muchos dueños de producto utilizan una variación en el *Then.* Entonces, en vez de utilizar la palabra hacen las post-condiciones implícitas, colocándolas después de la coma del *When.* En este caso, utilizan la frase "*I Should"*. Aún y cuando el formato no es adecuado y se registra el error de uniformidad, no se procesa como un *well_formed_no_then*. La siguiente imagen muestra el registro del error en AQUSA+:

Figura 19. Regla uniform_criteria.

La severidad en este caso, al igual que en AQUSA, se considera moderada, pues aunque el criterio de aceptación no se adhiera completamente al formato establecido, queda a criterio del experto, si aun así se pueden identificar precondiciones, evento y postcondiciones en el texto.

3.1.4.5.2 Único

Para la validación de esta regla se implementó un método similar al de AQUSA en la clase analizadora de criterios de aceptación. La evaluación de la duplicidad se realiza al comparar cada escenario del criterio de aceptación contra el resto de escenarios presentes en el archivo *.csv*. De esta manera se analiza que aún y cuando el criterio de aceptación entre historias de usuario sea diferente, siempre pueden existir escenarios duplicados.

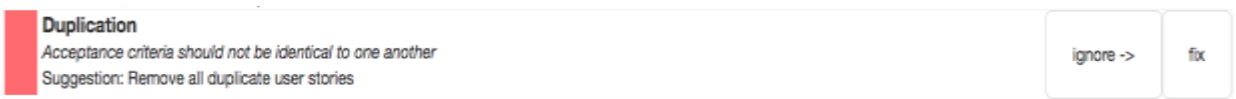

Figura 20. Regla unique_criteria.

La figura anterior presenta un ejemplo del reporte de un error en la regla ‘único’ para el criterio de aceptación. La severidad para este criterio es alta, por eso el color rojo en el indicador a la izquierda.

3.1.4.5.3 Verificable

La implementación de esta regla es la que compendia la mayoría de la nueva lógica que AQUSA+ provee. La presente sección muestra cómo, por primera vez, en el prototipo se introduce el análisis del árbol sintáctico para cada componente del criterio de aceptación.

Como se mencionó *supra*, las reglas pragmáticas tienen en común el proceso de *chunking* y *‘POS Tagging’,* de manera que a la hora de evaluarse solo recorren un árbol de tipo *‘NTLK Tree’* a través de sus tuplas; la etiqueta de la tupla -dependiendo del componente que evaluado- marca la pauta sobre la verificabilidad del escenario.

También, cada componente es procesado para eliminar caracteres especiales del texto, de manera que se pueda realizar un mejor análisis del contenido.

3.1.4.5.3.1 Regla *verifiable_user_interaction*

La implementación de esta regla se realiza mediante la evaluación del *Given*, el cual se recibe como parámetro en el método. Después de descomponerlo y etiquetar cada una de sus palabras, se procede analizar si existe algún tipo de sustantivo propio en singular o plural; es decir, la tupla tiene una etiqueta del tipo *NPP* o *NPPS* o bien, un pronombre propio *PRP*.

```
for x in result:
  if hasattr(x, 'label'):
    if 'PRP' in x.label().upper(): no_interaction = True
    elif 'NNP' in x.label().upper(): no_interaction = True
```

```
  elif 'NNPS' in x.label().upper(): no_interaction = True
else:
  if x[1] == 'PRP': no_interaction = True
  elif x[1] == 'NNP': no_interaction = True
  elif x[1] == 'NNPS': no_interaction = True
```

Este fragmento de código es el que se utiliza para la evaluación de las etiquetas del árbol y se evalúa la posición 1, pues la palabra es la posición 0 de la tupla.

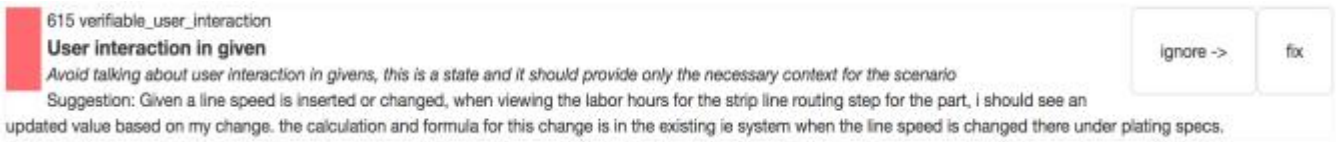

Figura 21. Regla verifiable_user_interaction.

La severidad del error es alta, pues compromete el estado que debe reflejar el *Given.* Se presenta una leyenda con la razón por la que no debe existir interacción del usuario y en la sugerencia se muestra el texto original, puesto que esta función no estaba contemplada para AQUSA+.

3.1.4.5.3.2 Regla *verifiable_stative_verb*

Esta regla se evalúa en el *Given* y, al igual que la anterior, se recorre el árbol de análisis sintáctico para validar que no existan formas verbales dinámicas expresadas en las etiquetas VB, VBZ, VBN, VBG y VBD. La siguiente porción de código de este método realiza dicha verificación:

```
if 'VB' == x[1].upper(): no_state = False
elif 'VBZ' == x[1].upper(): no_state = False
elif 'VBN' == x[1].upper(): no_state = False
elif 'VBG' == x[1].upper(): no_state = False
elif 'VBD' == x[1].upper(): no_state = False
```

El mensaje de error que se registra es de severidad alta y se presenta en la siguiente figura:

Figura 21. Regla verifiable_stative_verb.

3.1.4.5.3.3 Regla *verifiable_dynamic_verb*

Para esta regla se verifica que no existan verbos estáticos o conjugados en pasado; es decir, las etiquetas que harían que se registre un error serían: VB, VBD, VBN, VBP y VBZ. Esta evaluación se realiza mediante la siguiente porción de código:

```
if 'VB' == x[1].upper(): no_action = False
elif 'VBD' == x[1].upper(): no_action = False
elif 'VBN' == x[1].upper(): no_action = False
elif 'VBP' == x[1].upper(): no_action = False
elif 'VBZ' == x[1].upper(): no_action = False
```

En cuanto al mensaje de error, al igual que los demás de la categoría “Verificable”, se considera de severidad alta y se presenta de la siguiente forma:

Figura 22. Regla verifiable_dynamic_verb.

3.1.4.5.3.4 Regla *verifiable_output*

Para la verificación de esta regla, se utiliza la función de similitud semántica de *Wordnet*. Se creó una lista con términos extraídos de North (2006) que denotan ejemplos de palabras comunes esperadas como post-condiciones:

output_list = ['output','outcome','report','interface','message', 'email', 'screen', 'window', 'show', 'content', 'result']

Al recorrer el árbol de análisis sintáctico, se compara cada sustantivo en plural y singular (NN, NNS) que se encuentre en el *Then* mediante la siguiente porción de código:

```
no_output = True
noun_list=['NN','NNS']
for x in result:
  try:
    if hasattr(x, 'label'):
      if len(x[0][0][1]) > 1:
        if x[0][0][1] in noun_list: no_output = AnalyzerCriteria.semantic_similarity(x[0])
```

Como se puede observar, la comparación se realiza mediante el método *'semantic_similarity'*:

```
for x in output_list:
  w1 = wordnet.synset(x+'.n.01')
  w2 = wordnet.synset(word[0]+'.n.01')
  if w1.wup_similarity(w2) > 0.34: is_similar = False
```

Este método recibe los sustantivos uno por uno y realiza una comparación de similitud semántica contra las palabras de la lista de términos y cada uno de sus sustantivos. Si en alguna de estas comparaciones el valor de similitud es mayor al 34 %, acepta el *Then* como un conjunto de post-condiciones verificable con un resultado observable.

En caso de no encontrar ningún término similar o, en su defecto, que no existan sustantivos en el *Then*, se registra y presenta el siguiente error:

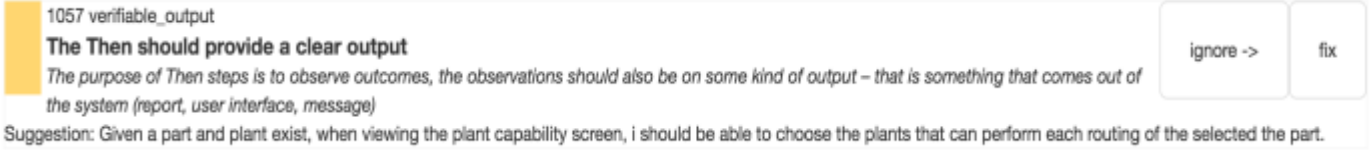

Figura 22. Regla verifiable_output.

En este caso, la severidad es moderada, dado que el análisis de similitud semántica no es tan preciso como una comparación directa con una etiqueta, como se realiza en las reglas anteriores. Por lo tanto, esta categoría se registra más como una advertencia al dueño de producto para que revise si el resultado es realmente observable.

3.1.4.5.3.5 Regla *verifiable_title*

El análisis de la verificabilidad del título tiene un proceso distinto al del criterio de aceptación debido a que no se realiza la fase de *chunking* sino que únicamente se realiza el *POS Tagging* del texto.

En cuanto a la gramática, utiliza la misma que en el criterio de aceptación y únicamente agrega la siguiente línea: TITLE: {<AP>?<VP>}. Aunque no hace *chunking*, utiliza el método de *content_chunk* para hacer el análisis sintáctico y generar el árbol respectivo. Después busca la presencia de alguna forma verbal en las etiquetas de dicho árbol por medio del siguiente código:

```
if 'VB' == x[1].upper(): not_verifiable = False
elif 'VBZ' == x[1].upper(): not_verifiable = False
elif 'VBN' == x[1].upper(): not_verifiable = False
elif 'VBG' == x[1].upper(): not_verifiable = False
elif 'VBD' == x[1].upper(): not_verifiable = False
elif 'VBP' == x[1].upper(): not_verifiable = False
```

Se aceptó la mayoría de formas verbales, dado que se trata de describir una actividad para la cual no se especificó una forma escribirla (North, 2006). El registro y la presentación del error se muestran en la siguiente figura y corresponde a una violación de severidad alta:

Figura 23. Regla verifiable_title.

Para el título tampoco se implementó la sugerencia, por lo tanto, solo se presenta el texto original.

3.1.4.6 Cambios en la interfaz

La interfaz de AQUSA presentaba los errores correspondientes a cada descripción debajo de cada una de ellas con su identificador único, de la siguiente manera:

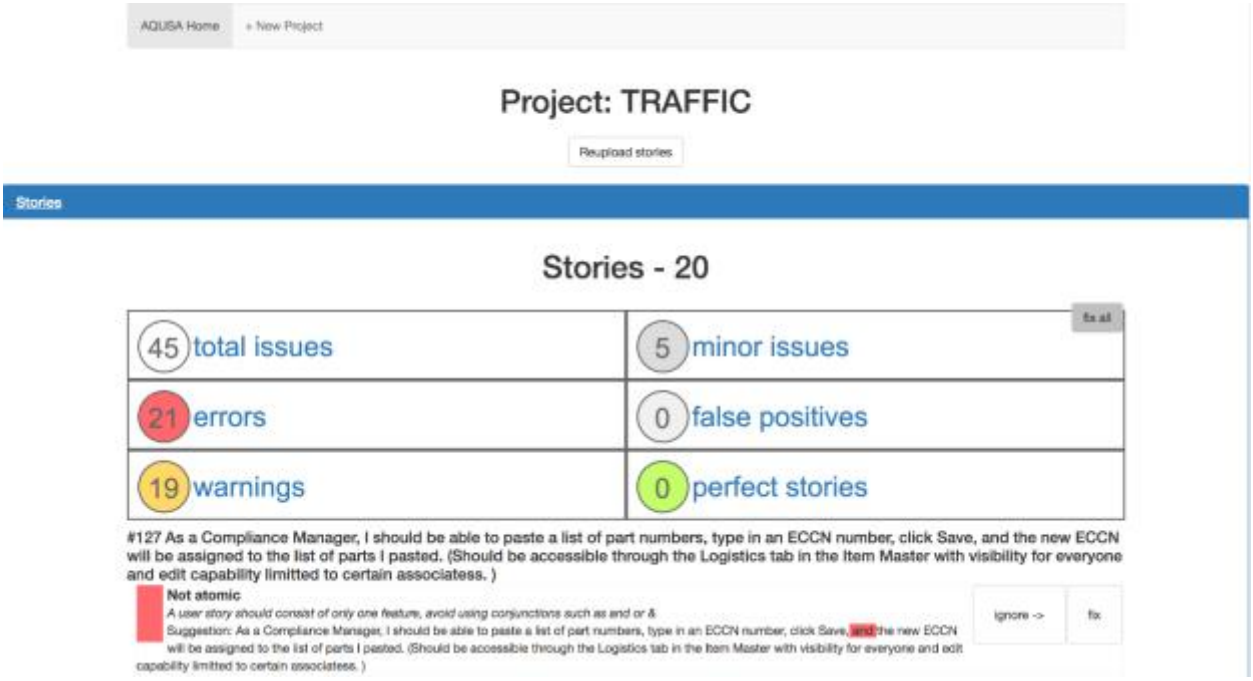

Figura 24. Interfaz de AQUSA.

También presentaba una tabla de dos columnas por tres filas. En la primera columna, se lista el total de violaciones encontradas. Debajo de esa, la cantidad de errores y la de advertencias.

La segunda columna tiene *inconvenientes menores; falsos positivos* e *historias perfectas*. De estos, solo el último enlace muestra la lista de historias sin ningún error o advertencia. Los otros dos enlaces no fueron implementados en AQUSA y tampoco en AQUSA+.

Para AQUSA+ se extendió el "acordeón" para incluir la sección de criterio de aceptación (*Criteria)* y del título (*Title*), como se observa en la siguiente figura:

Figura 25. Interfaz de AQUSA+.

Cada uno de los componentes del acordeón se expande para mostrar el registro de errores para cada categoría, tal y como se mostró en la figura 25 para la descripción de las historias de usuario.

3.1.4.7 Manejo de errores

Cada error o advertencia generada en la clase analizadora es procesado por un método llamado *generate_errors*, el cual fue duplicado desde AQUSA para el criterio de aceptación y el título. En este método se realiza la categorización del tipo de error. El texto que debe ser subrayado en la interfaz y se almacena en la base de datos ligado al componente al que pertenezca. El ejemplo siguiente realiza el registro de un error para el título:

```
ErrorTitle.create_unless_duplicate(eval(error_type['highlight']), kind,
error_type['subkind'], error_type['severity'], title)
```

La tabla *error* tiene un atributo *kind* para determinar a cuál regla pertenece y hacer el enlace con los mensajes de error almacenados en la raíz del repositorio en un archivo conocido como *messages.pot*.

Para AQUSA+ se extendió el archivo, a fin de incorporar los mensajes de error de las reglas relacionadas con el título y el criterio de aceptación.

3.1.5 AQUSA+ en ejecución

Al hacer el análisis de cada una de las historias de usuario, la herramienta emula parcialmente el trabajo de escrutinio que debe realizar un experto humano.

Para validar y comparar los resultados de la evaluación que el modelo provee, se eligió un conjunto de cuatro archivos con 20 historias de usuario cada uno. Estos archivos fueron cargados uno por uno en AQUSA+ y evaluados por la herramienta.

A su vez, un grupo de cuatro expertos evaluaron los archivos basados en una rúbrica de calificación, la cual también fue utilizada para emitir una calificación con base en el resultado que brindó AQUSA+ para cada archivo, de manera que las historias fueran analizadas por ambas partes con los mismos criterios.

Finalmente se evaluó AQUSA+ mediante un archivo *benchmark;* es decir, con historias que no presentaban errores para determinar cuáles son los falsos positivos que AQUSA+ señaló – en caso de que los hubiere.

Las siguientes subsecciones detallan los procedimientos que se llevaron a cabo para determinar la eficiencia de AQUSA+ y el grado de discrepancia que existe entre sus calificaciones y las de los expertos.

3.1.5.1 Evaluación de historias de usuario en AQUSA+

Una vez finalizada la implementación y las pruebas de AQUSA+, se procedió a cargar cuatro archivos de extensión *.csv* por separado para ser analizados por la herramienta. Estos archivos pertenecen a subconjuntos de 20 historias de usuario reales, pertenecientes a una empresa de la industria electrónica que realiza su desarrollo de *software* bajo el marco de trabajo *Scrum*.

Cada archivo representa un subconjunto de requerimientos de un proyecto distinto. La siguiente tabla muestra el nombre de cada archivo, un identificador que será utilizado a lo largo de este documento y la fecha estimada de inicio y finalización del proyecto al que pertenecían.

Identificador	**Nombre**	**Fecha Inicio**	**Fecha Final**
AR	Accounts Receivable	Junio 2015	Febrero 2016
IM	Item Master	Octubre 2014	Abril 2015
MID	Managed Inventory Database	Setiembre 2015	Diciembre 2015
TR	Traffic	Diciembre 2015	Marzo 2016

Table 3 Proyectos seleccionados.

Estos datos provienen la herramienta *Team Foundation Server* de Microsoft, la cual permite la administración de portafolios de proyectos en modalidad ágil. Para crear un nuevo proyecto en AQUSA+, primero se le da un nombre por medio de la interfaz que se muestra en la siguiente figura:

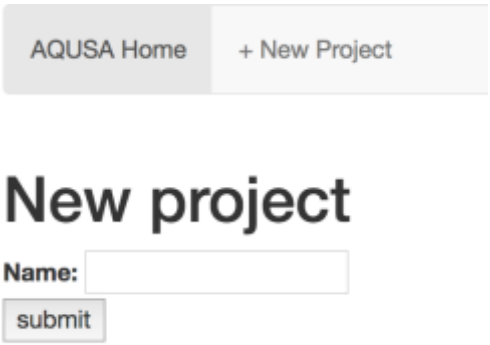

Figura 26. Nuevo proyecto.

Una vez asignado un nombre se presiona el botón *submit* y eso lo dirige a una página que permite hacer la carga del archivo, como se muestra a continuación:

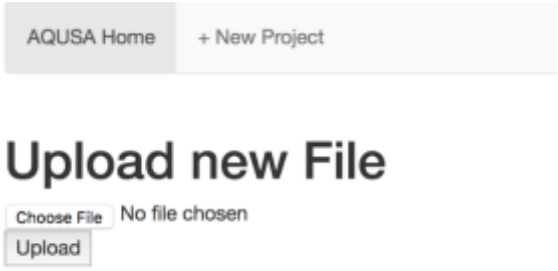

Figura 27. Cargar archivo.

Acá se hace clic en *Choose File*, lo que abre una ventana para seleccionar el archivo *.csv* deseado. Una vez seleccionado, se hace clic en *Upload* para cargar el archivo y, a la vez, hacer el análisis de cada una de las historias de usuario. Cuando este proceso finaliza, se carga el reporte de errores automáticamente. El siguiente es un ejemplo de la interfaz del resultado después del análisis para el archivo AR.

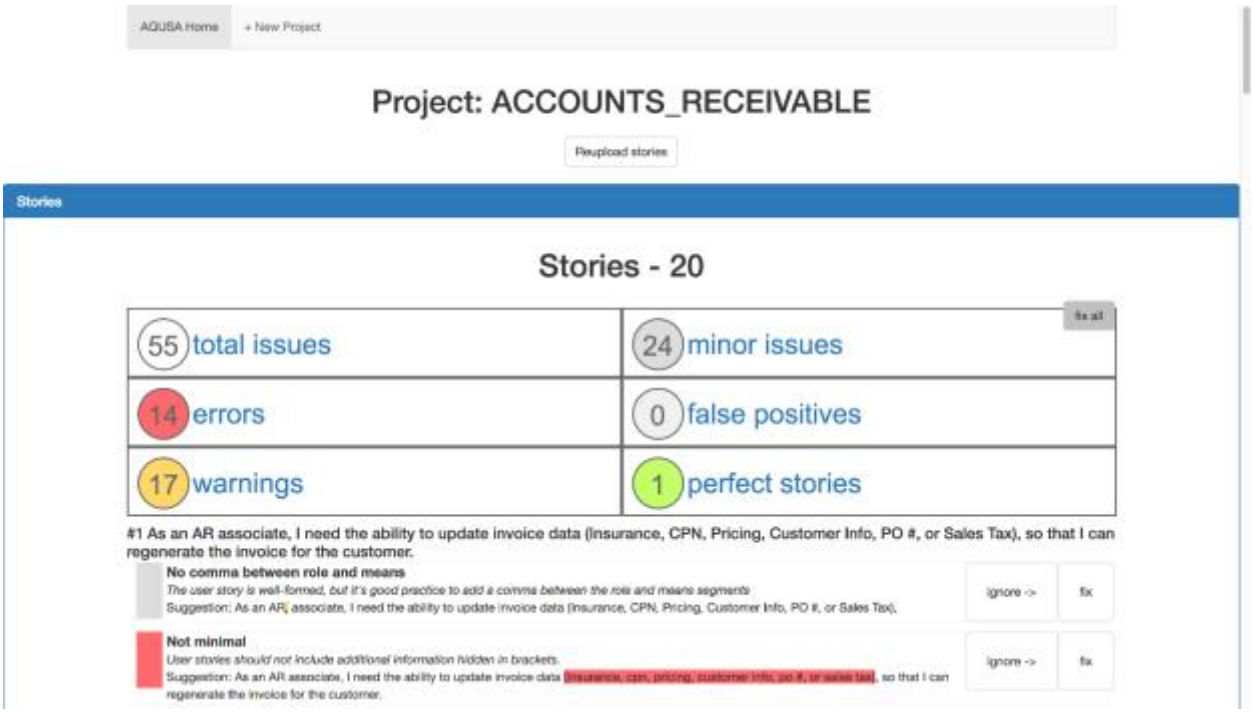

Figura 28. Interfaz de registro de errores AQUSA+.

Una vez generado el informe de errores, se puede consultar en cualquier momento debido a que todos sus datos son almacenados en la base datos de la herramienta.

3.1.6 Rúbrica de evaluación

Dado que AQUSA+ es una herramienta de apoyo al experto, la evaluación final de las historias debe ser responsabilidad de este. Para estandarizar la forma en que se genera la calificación de cada historia de usuario, se creó una rúbrica de evaluación basada en Earls (2005), que permite otorgar un valor de calificación a cada descripción, un criterio de aceptación y un título basado en cada criterio de calidad sintáctico y pragmático.

Componente	Sofisticada	Competente	No es competente
Descripción			
Bien formada	- La descripción de la historia de usuario está gramaticalmente bien escrita con los signos de puntuación adecuados. - Se identifica un rol y un método para alcanzar la funcionalidad requerida.	- La descripción está escrita con algunos errores de puntuación, como comas faltantes. - Se identifica un rol pero no es clara su conformación. - El método carece de un sustantivo o un verbo clave.	- La descripción de la historia de usuario no está bien escrita, presenta errores gramaticales. - No es posible identificar un rol o un método para alcanzar un fin.
Atómica	- La descripción de la historia de usuario representa una única funcionalidad.	- La descripción de la historia de usuario presenta palabras clave como "And" o conectores como: "&". Sin embargo no es claro que la historia deba separarse en dos o más.	- La descripción de la historia de usuario no es atómica, representa más de una funcionalidad y la historia debe ser separada en dos o más.
Mínima	- La descripción de la historia de usuario provee únicamente la información necesaria.	- La descripción de la historia de usuario presenta la información necesaria más algunas notas que son prescindibles.	- La cantidad de información en la descripción de la historia de usuario es excesiva y dificulta que sea entendida.
Uniforme	- La descripción de la historia de usuario se alinea con el formato establecido.	- El formato utilizado se asemeja al recomendado y se identifican los componentes.	- El formato para la descripción de la historia de usuario no cumple con las características necesarias.
Única	- La descripción de la historia no tiene duplicados	- La historia es similar a otra(s) en la lista pero no es claro que sea duplicada.	- La descripción de la historia de usuario está duplicada en la lista.

Componente	Sofisticada	Competente	No es competente
Criterio de Aceptación *Bien Formado*	- El criterio de aceptación está gramaticalmente bien escrito con los signos de puntuación adecuados. - Se identifican las precondiciones y el evento para alcanzar los resultados requeridos.	- El criterio de aceptación está escrito con algunos errores de puntuación, como comas faltantes.	- El criterio de aceptación no está bien escrito, presenta errores gramaticales. - No es posible identificar las precondiciones o el evento que genera el escenario.
Atómico	- El criterio de aceptación representa una única funcionalidad.	- El evento en el criterio de aceptación presenta palabras clave como "And" o conectores como: "&". Sin embargo, no es claro que la historia deba separarse en dos o más.	- Existen eventos diferentes dentro de los escenarios del criterio de aceptación. La historia debe ser separada en dos o más.
Mínimo	- El criterio de aceptación no presenta información adicional además del escenario principal ni tampoco especificada en paréntesis o llaves.	- El criterio de información presenta información adicional o en paréntesis/llaves pero no afecta la comprensión su comprensión.	- El criterio de aceptación presenta información adicional y en llaves/paréntesis haciendo que se dificulte su comprensión.
Verificable Precondiciones Evento	- La historia de usuario provee únicamente la información necesaria y no existe interacción de parte del usuario. - Las precondiciones representan un estado.	- Las precondiciones se distinguen, sin embargo, su tiempo verbal no corresponde a un estado. - Existe interacción del usuario pero se identifica un estado previo a la acción.	- Existe interacción del usuario. - Las precondiciones no representan un estado.

Postcondiciones	- El evento es único entre todos los escenarios del criterio de aceptación. - El evento está escrito como una acción. - Las postcondiciones muestran un resultado, un valor que se provee después de realizar la acción. - El resultado es observable.	- El criterio de aceptación representa una acción pero su tiempo verbal no es el adecuado. - Existe un resultado después de realizar la acción/ evento sin embargo no es claro que éste se pueda verificar.	- Uno o más eventos no son representados por una acción. - No es posible identificar un resultado a la acción realizada. - El resultado no es observable.
Uniforme	- El criterio de aceptación se alinea con el formato establecido (Given/When/Then).	- El formato utilizado se asemeja al recomendado y se identifican los componentes.	- El formato del criterio de aceptación no cumple con las características necesarias y/o no se identifican los componentes.
Único	- No existen duplicados de este criterio de aceptación.	- Existe otro criterio de aceptación que se asemeja pero no es posible identificar que sea duplicado.	- El criterio de aceptación se encuentra duplicado en la lista de historias de usuario.

Componente	**Sofisticada**	**Competente**	**No es competente**
Título			

Representa una actividad	- El título de la historia representa una actividad.	- El título posee un verbo pero la actividad no es clara.	- El título de la historia de usuario no representa una actividad.

Tabla 4. Rúbrica de evaluación.

La tabla anterior corresponde a la totalidad de la rúbrica, como se puede observar y, de acuerdo con los resultados obtenidos en AQUSA+, el experto debe otorgar una calificación a cada componente de la historia de usuario, según el criterio de calidad en evaluación.

La calificación final por parte del experto funciona de la siguiente manera:

- Si el componente en ese criterio de calidad específico es sofisticado, se asignan dos puntos.
- Si el componente es competente, se asigna un punto.
- Si el componente no es competente en su escritura, se asignan cero puntos.

Para asignar estos puntajes de acuerdo con los resultados que arroja AQUSA+, se creó el siguiente heurístico de calificación, basado en la ponderación que realizaría un experto según la cantidad de incidencias encontradas.

Componente	**Criterio**	**Sofisticada**	**Competente**	**No es competente**
Descripción				
	Bien formada	No existen errores *well_formed*	Existen errores de tipo *well_formed_no_means_comma* y/o *well_formed_no_ends_comma*	Errores de tipo *well_formed_no_means, well_formed_no_role*
	Atómica	No existen errrores *atomic.*	Existen errores de tipo *atomic_conjunctions* pero no es claro que existan dos funcionalidades.	Existen errores de tipo *atomic_conjunctions* y es claro que existen dos funcionalidades.

	Mínima	No existen errores *minimal.*	Existen errores de tipo *minimal_bráckets* o *minimal_punctuation*	Existen errores de tipo *minimal_bráckets* y *minimal_punctuation*
	Uniforme	No existen errores *uniform.*	Existe la advertencia *Irregular_format* pero es debido a que los tres componentes fueron especificados con otras palabras.	Existe la advertencia Irregular_format y no es posible descrifrar la plantilla de escritura.
	Única	No existen errores de tipo *identical.*	Existe error de tipo *identical* pero no es posible determinar si ambas historias son iguales.	Existe error *identical* y se comprueba la duplicidad en la lista de historias.
Criterio de aceptación				
	Bien Formado	No existen errores *well_formed*	Existen errores de tipo *well_formed_no_when_comma* y/o *well_formed_no_then_comma*	Errores de tipo *well_formed_no_when, well_formed_no_given,well_formed_no_then*
	Atómico	No existen errrores *atomic*	Existen errores de tipo *atomic_criteria_conjunctions* pero no es claro que existan dos funcionalidades.	Existen errores de tipo *atomic_one_feature*
	Mínimo	No existen errores *minimal.*	Existen errores de tipo *minimal_criteria_bráckets* o *minimal_criteria_punctuation*	Existen errores de tipo *minimal_criteria_bráckets* y *minimal_criteria_punctuation*

	Verificable	No existen errores de tipo *verifiable_user_interaction, verifiable_dynamic_verb,verifiable_stative_verb, verifiable_output*	Existe una incidencia de error de tipo verifiable_user_interaction, o verifiable_dynamic_verb o verifiable_stative_verb, ó verifiable_output	Existe más de una incidencia de error de tipo *verifiable.*
	Uniforme	No existen errores *uniform_criteria*	Existe la advertencia *Irregular_format* pero es debido a que en vez de utilizar la palabra *Then,* se asume el componente como el texto después de la coma en el *When.*	Existe la advertencia *Irregular_format* y no es posible descrifrar la plantilla de escritura.
Título				
	Verificable	No existe error de tipo *verifiable_title*	Existe error de tipo *verifiable_title* y AQUSA no fue capaz de identificar la actividad.	Existe error de tipo *verifiable_title* y es claro que no se describe una actividad.

Tabla 5. Heurístico de calificación con AQUSA+.

Una vez revisado el registro de errores de AQUSA+ y, determinada una calificación para cada historia de usuario, en cada uno de sus componentes para los diferentes criterios de calidad se procedió a llenar la siguiente tabla que reúne la calificación total para cada archivo.

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	2	1	0	1	2	1	2	1	1	2	1
2	0	1	0	1	1	2	1	2	1	1	2	1
3	0	2	2	2	1	2	0	0	2	0	2	0
4	0	2	0	0	1	2	1	2	0	1	2	1
5	0	2	2	1	2	2	0	0	1	0	2	0
6	2	2	0	2	1	2	0	0	2	0	2	0
7	0	2	0	1	1	2	0	0	2	0	2	0
8	0	2	2	2	1	2	2	2	2	1	2	1
9	2	2	1	1	1	2	2	2	2	1	2	1
10	2	2	1	2	1	2	1	2	2	1	2	1
11	0	2	1	2	1	2	1	2	2	2	2	1
12	0	1	0	2	2	1	0	0	2	0	2	1
13	0	2	2	1	1	2	0	0	2	0	2	0
14	2	2	2	1	1	2	0	0	2	0	2	1
15	0	2	2	2	1	2	0	0	2	0	2	1
16	2	1	2	0	1	2	0	0	2	0	2	1
17	0	1	2	2	1	2	0	0	2	0	2	0
18	2	2	2	2	1	2	0	0	0	0	2	0
19	2	1	2	2	1	2	0	0	2	0	2	0
20	2	2	1	1	1	2	0	0	2	0	2	0

Figura 29. Hoja de calificación utilizando AQUSA+.

La figura 29 muestra cómo se ve la tabla final de calificación, una vez que se ha analizado una lista de historias de usuario. Esta tabla no es solo usada para la calificación desde AQUSA+, sino que, también la utilizaron los expertos para calificar cada historia con base únicamente en la rúbrica.

3.1.7 Expertos

Como contraparte humana a AQUSA, se seleccionaron cuatro expertos en la materia; personas que en su día a día se dedican a trabajar con historias de usuario, con dueños de producto y equipos de desarrollo de *software*.

Estos individuos realizaron la evaluación de los mismos archivos de historias de usuario, al otorgar calificaciones a cada historia por componente y criterio de calidad.

Como resultado final, entregaron una hoja de cálculo con dos pestañas; cada una con una tabla de evaluación igual a la especificada en la figura 29. Dichas tablas fueron utilizadas para los diversos experimentos que se detallarán más adelante. La tabla 6 presenta el perfil profesional de cada uno de los expertos y un identificador que se utilizará para referirse a ellos en las futuras secciones.

Identificador	Nombre	Cargo Actual	Experiencia
OA	Oscar Alfaro	*Scrum Master*	• 2.5 años como *Scrum Master*. • 13 años en la industria del software. • Bachiller en Informática Universidad de Costa Rica. • Profesor de Maestría en Administración de Proyectos en Universidad de Costa Rica.
JB	Juan Pablo Blanco	*Scrum Master*	• 7 años como *Scrum Master*. • 10 años en la industria del software. • Bachiller en Informática Universidad de Costa Rica. • Scrum Master certificado por el *Scrum Alliance.*
KL	Kevin Leandro	*Scrum Master*	• 5 años como administrador de proyectos. • 2 años como *Scrum Master* • 14 años en la industria del software. • Bachiller en Informática Universidad Latina de Costa Rica. • Scrum Master certificado por el Scrum Alliance. • Certificado Project Manager (PMP) por el *PMI Institute.*
ER	Esteban Rodríguez	*Scrum Master*	• 5 años como *Scrum Master*.

			• 15 años en la industria del software. • Bachiller en Ciencias de la computación e Informática Universidad de Costa Rica. • *Scrum Master* certificado por el *Scrum Alliance*. • Practicante certificado de Scrum (CSP) de *Scrum Alliance.*

Tabla 6. Perfil profesional de los expertos.

Cada experto recibió las indicaciones sobre cómo manejar la rúbrica y la tabla de evaluación en una reunión virtual llevada a cabo el 9 de febrero de 2017 a las 18:00 horas, a través de la plataforma Skype. En dicha reunión se evacuaron las dudas que surgieron después de la explicación del procedimiento y se dio por concluida la sesión. Posteriormente, cada experto recibió cuatro archivos; la rúbrica de evaluación, el archivo con las tablas de evaluación y los dos archivos *.csv* que les correspondía evaluar.

Una vez que finalizaron el escrutinio de los archivos, cada uno envió al investigador sus tablas de evaluación vía correo electrónico.

La siguiente sección detalla los experimentos llevados a cabo, una vez que se tuvieron a mano los resultados de los expertos y de la herramienta.

3.1.8 Experimentación

La metodología que siguió esta investigación es de tipo cuantitativa con experimentos controlados. De acuerdo con Wohlin, Host y Henningson (2003), una estrategia cuantitativa es apropiada debido a que se busca evaluar los efectos de uno o más tratamientos que, en este caso, corresponden a las evaluaciones de las historias de usuario por los expertos y por la herramienta. Dicho experimento posee las siguientes características:

- Están limitados a un alcance reducido (historias de usuario de cuatro proyectos).

- Los sujetos son asignados a diferentes tratamientos de manera aleatoria (expertos con sus respectivos grupos de historias de usuario).
- El efecto de la manipulación es medido (calificación de las historias de usuario por parte de la herramienta y los expertos).
- Se puede realizar un análisis estadístico con los resultados.

3.1.8.1 AQUSA+ vs. Expertos

3.1.8.1.1 Objetivo

Determinar si la utilización de AQUSA+ representa una diferencia positiva en el análisis de un conjunto de historias de usuario; es decir, que sus calificaciones sean parecidas a las del grupo de control. A su vez, determinar si su aplicación permite encontrar más errores que los expertos humanos, mediante el análisis de los diferentes promedios de calificación.

Para esto se establecieron tres fases: diseño, operación y análisis e interpretación. En la fase de *diseño* se detallan algunas condiciones comunes acerca de las historias de usuario que fueron utilizadas, en la de *operación* se muestra cómo se llevó a cabo el proceso de calificación por parte de los expertos y la herramienta. Finalmente se sientan las bases para el *análisis* estadístico y la *interpretación* de los resultados

3.1.8.1.2 Diseño

1. Todas las hojas de evaluación siguen la misma conformación y el formato al mostrado en la figura 29.

2. Existen 12 criterios de calidad en total; la mayor calificación por casilla es de dos puntos, por lo tanto, la mejor calificación que una historia puede tener es de 24 puntos. Por ende, la calificación máxima que un conjunto de 20 historias de usuario puede obtener, como grupo, es de 480 puntos.
3. Dado que son 20 historias por archivo, la mejor calificación que un criterio de calidad puede tener es de 40 puntos en total.
4. La muestra corresponde a cuatro archivos con historias de usuario pertenecientes al mismo proyecto escogidas al azar.
5. Los cuatro proyectos también fueron escogidos al azar de un grupo de 13 proyectos de *software* de una compañía de inter-conectores electrónicos que trabaja bajo metodologías ágiles.
6. Estos 13 proyectos tuvieron vigencia entre mayo de 2014 y febrero de 2016.

El tipo de diseño de este experimento es 'Estándar 4', basado en Wohlin *et al*. (2003); es decir, más de una variable independiente. Por su parte, cada historia de usuario tuvo cuatro posibles calificaciones: evaluador 1 (Eval1), evaluador 2 (Eval 2), AQUSA+ y la muestra estándar o grupo control.

La muestra estándar es la calificación de cada historia de usuario realizada por el investigador. Se utiliza como grupo control, pues es la que presenta las calificaciones y observaciones basadas en las definiciones de los diferentes criterios de calidad.

3.1.8.1.3 Operación

1. Se realizó la asignación aleatoria de conjuntos de historias de usuario a cada uno de los participantes, la cual resultó de la siguiente forma:

Id Archivo	Id Experto 1 (Eval1)	Id Experto 2 (Eval2)
AR	OA	JB
IM	KL	ER
MID	OA	JB
TR	KL	ER

Tabla 7. Asignación de expertos a proyectos.

2. Se entregó la siguiente lista de instrumentos el día 8 de febrero de 2017.
    a. Dos archivos con historias de usuario.
    b. Una hoja de evaluación (anexo B).
    c. Una rúbrica de evaluación (tabla 4).

3.1.8.1.4 Ejecución

1. Cada experto realizó la evaluación de sus historias de usuario utilizando los criterios de la rúbrica de evaluación para calificar cada historia de usuario en los proyectos que les fueron asignados.
2. Entregaron hojas de evaluación llenas el día 14 de marzo de 2017.
3. Se procedió a evaluar los diferentes conjuntos de historias de usuario utilizando AQUSA+ y el investigador llenó la hoja de evaluación (propia), con base en los lineamientos de la sección 3.1.7.
4. Se creó una tabla de resultados finales que compendió la calificación de cada una de las partes para cada historia de usuario separada por archivo.

3.1.8.1.5 Análisis e interpretación

El tipo de análisis utilizado para este experimento se conoce como Análisis de Bloques Aleatorios o bien, ANOVA (*Analysis of Variance*) de dos factores. Se utiliza para comparar dos o más poblaciones con observaciones bloqueadas.

En el caso de esta investigación, las observaciones bloqueadas son los proyectos y sus historias de usuario. Las poblaciones corresponden a las calificaciones de cada evaluador. La idea del bloque es separar la variabilidad de historia a historia; es decir, las historias dentro de un proyecto son diferentes entre sí, pero los evaluadores realizaron su calificación sobre un mismo grupo común de historias, o sea, la historia número 1 del archivo AR es la misma para el evaluador 1, el evaluador 2, AQUSA y el grupo control.

El análisis y la interpretación de los resultados se realizó mediante la herramienta de análisis estadístico JMP (Sall, 2017) y su lógica se basó en el siguiente diagrama de flujo [4]:

[4] *Sig Dif: diferencia significativa.*

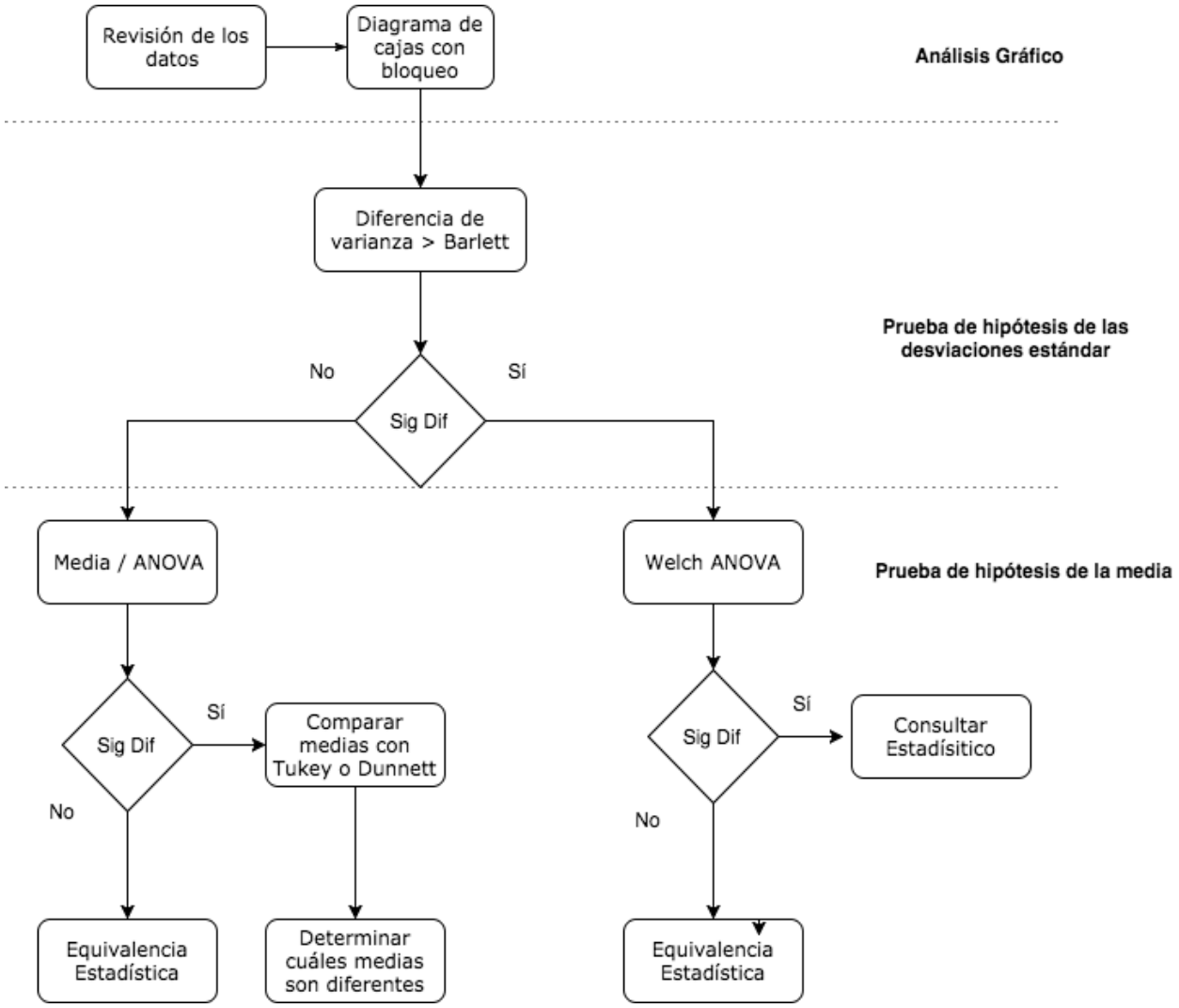

Figura 30. Diagrama de flujo del análisis estadístico (Montgomery, 2008).

Después de tabular los datos en JMP, se deben analizar algunos gráficos, como el de análisis de un factor por historia de usuario para identificar si existe alguna calificación en que se dé una diferencia significativa en alguno de los casos en particular.

Seguido de esto se observa el gráfico de cajas por bloque para determinar cuáles evaluadores tuvieron medias equivalentes o diferentes con respecto a la muestra estándar. Sin embargo, antes de proceder al análisis de varianza se realiza un estudio de la igualdad de varianza de las poblaciones mediante el método de Barlett (Montgomery, 2008), donde el nivel de

significancia del valor p de cada población debe ser mayor a 0.05. Si esto se cumple, se dice que no se puede rechazar la hipótesis nula H_0:

- H_0: Todas las varianzas son iguales.
- H_1: No todas las varianzas son iguales.

Posteriormente se utilizó ANOVA y, con base en el nivel de significancia de cada bloque, se determinó la equivalencia estadística.

3.1.8.2 Falsos Positivos

Para esta investigación se define como '*falso positivo*' a aquella incidencia registrada en el reporte de errores de AQUSA que no representa un error en la escritura de alguno de sus tres componentes principales.

Este apartado tiene los siguientes objetivos:

1. Recorrer el registro de errores de AQUSA+ para cada uno de los cuatro proyectos y determinar cuáles incidencias fueron erróneamente registradas por parte de AQUSA+.
2. Determinar la precisión de AQUSA+; es decir, la diferencia del total de errores para los cuatro proyectos con los falsos positivos.
3. Identificar las reglas con mayor cantidad de falsos positivos por archivo y analizar las razones del comportamiento.

3.1.8.2.1 Benchmark

El objetivo detrás del *benchmarking* es evaluar el rendimiento de la herramienta con una muestra de historias de usuario que siguen las mejores prácticas en su escritura. En teoría, la herramienta debería ser capaz de analizar el archivo y generar una calificación perfecta; es decir, sin reporte de errores.

Las historias de usuario escritas corresponden a un caso ficticio de un sistema de facturación en línea (anexo C). Su creación se realizó por medio de diferentes formas de describir los requerimientos, adhiriéndose previamente a los lineamientos de calidad, de acuerdo con Lucassen *et al*. (2015) y North (2006).

El proceso para evaluar los resultados del *benchmark* fue el siguiente:

1. Analizar el reporte de errores para cada componente, descripción, criterio de aceptación y título.
2. De existir reportes de errores, identificar si corresponden a falsos positivos.
    a. En el caso de la existencia de falsos positivos, registrar la cantidad de incidencias.

4 Análisis de resultados

A continuación se presenta un análisis detallado de los resultados obtenidos a partir de los experimentos, además de una explicación del método elegido para realizar el análisis estadístico con base en el diagrama de flujo de la figura 30.

4.1 AQUSA+ vs. Expertos

Como se mencionó anteriormente, cada experimento está compuesto por la evaluación del mismo conjunto de historias de usuario (bloque) por parte de dos evaluadores; AQUSA y el grupo de control.

Los datos obtenidos en cada evaluación se almacenaron en dos tablas que luego fueron utilizadas para alimentar la herramienta JMP y realizar el análisis estadístico por bloque.

	Managed Inventory Database				**Accounts Receivable**			
Historia de usuario	OA	JB	AQUSA	Control	OA	JB	AQUSA	Control
1	19	20	15	16	24	23	16	16
2	22	20	15	15	23	23	19	19
3	14	10	14	14	20	22	13	15
4	18	21	15	15	23	19	13	13
5	23	22	16	16	20	18	17	17
6	21	23	15	16	19	18	20	20
7	21	16	12	12	21	17	17	19
8	22	24	18	18	22	21	15	15
9	20	24	16	18	19	19	15	17
10	23	19	16	16	22	21	19	19
11	23	21	13	15	21	22	16	17
12	22	21	17	19	20	19	14	14

13	20	23	15	15	20	19	16	16
14	22	24	17	17	22	19	18	19
15	23	24	17	18	22	24	17	18
16	21	21	12	12	19	19	18	18
17	23	23	17	17	22	23	19	19
18	23	17	15	15	22	18	13	13
19	22	22	19	19	23	24	15	17
20	19	18	20	20	22	22	18	18

Tabla 8.Resumen de resultados MID & AR

	Item Master				**Traffic**			
Historia de usuario	KL	ER	AQUSA+	Control	KL	ER	AQUSA+	Control
1	19	18	19	18	17	12	16	16
2	17	19	16	19	19	17	13	13
3	15	17	12	12	22	20	13	15
4	16	19	21	21	13	10	12	12
5	22	17	18	18	21	14	12	12
6	22	23	22	23	24	19	13	13
7	12	19	17	17	19	17	10	10
8	21	20	16	17	23	20	19	21
9	19	20	14	14	24	21	19	20
10	13	17	14	14	19	19	19	19
11	20	15	17	17	22	20	18	18
12	15	20	18	18	17	18	11	11
13	20	20	20	20	23	17	12	12
14	16	12	16	16	17	22	15	15
15	19	19	17	17	16	19	14	14
16	18	19	17	17	18	17	13	13
17	17	18	14	14	23	21	12	12
18	14	14	13	13	16	14	13	13
19	14	16	10	10	24	21	14	14
20	11	15	13	14	23	16	13	13

Tabla 9. Resumen de resultados IR & TR.

Para cada bloque se realizó la prueba de Barlett para la igualdad de las varianzas, con las siguientes hipótesis:

- H_0: Todas las varianzas son iguales.
- H_A: No todas las varianzas son iguales.

Proyecto	**Prob > F**
Accounts Receivable	0.7649
Item Master	0.0539
Managed Inventory Database	0.1941
Traffic	0.3306

Tabla 10. Igualdad de varianzas.

En vista de que el valor p para cada proyecto fue mayor a 0.05, se concluyó que no se podía rechazar las hipótesis nula H_0 y se procedió a aplicar ANOVA sobre cada uno de los bloques.

4.1.1 Análisis por bloque

4.1.1.1 Análisis Gráfico

Para el ANOVA de cada uno de los proyectos se formularon las siguientes hipótesis:

- H_0: $\mu_{Eval1} = \mu_{Eval2} = \mu_{AQUSA+} = \mu_{Guía}$
- H_A: No todos los μ_i *son iguales*

Las siguientes cuatro figuras presentan los diagramas de cajas con la representación gráfica del promedio de las calificaciones de cada uno de los evaluadores para cada uno de los proyectos.

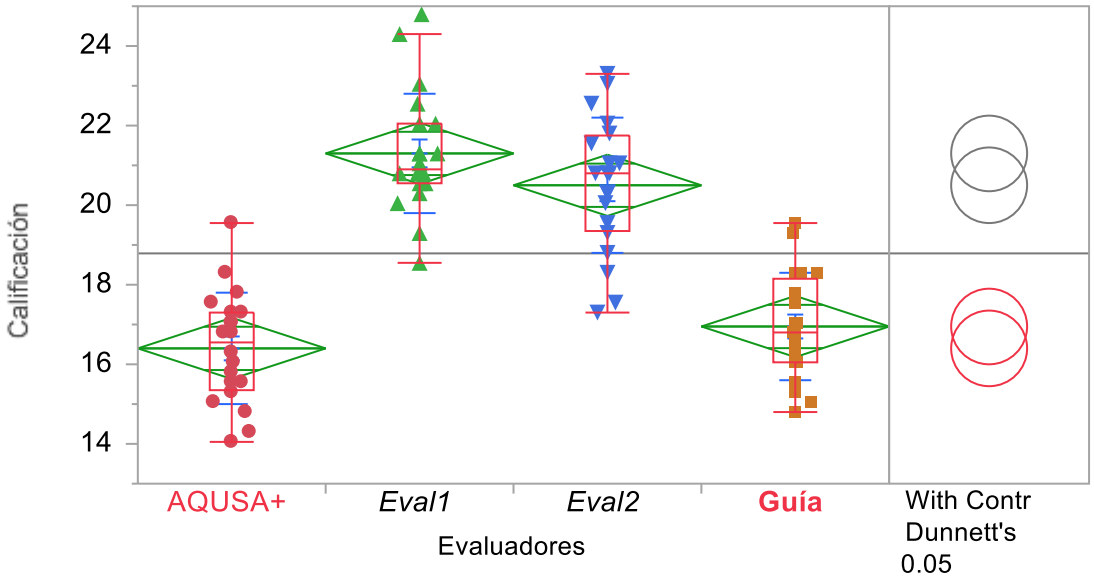

Figura 31. Diagrama de cajas para AR.

En *Accounts Receivable,* los promedios de calificaciones de los evaluadores se encuentran arriba de la media de calificación del proyecto, mientras que AQUSA+ y la muestra guía se ven más alineadas, situación que se confirma a la derecha del gráfico de control de Dunnet, donde sus círculos se traslapan.

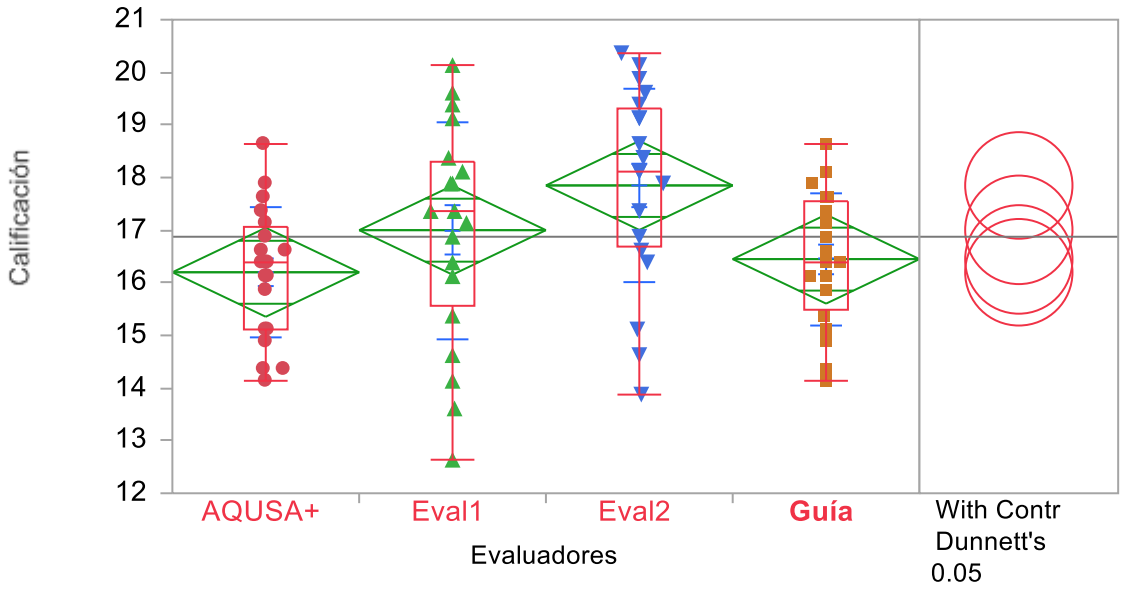

Figura 32. Diagrama de cajas IM.

Item Master fue el que presentó mayores similitudes en su diagrama de cajas y en el de Dunnet, como se observa en la figura 32; los cuatro círculos se traslapan entre ellos y es difícil determinar su diferencia. En cuanto a los promedios de calificación para cada uno de los evaluadores, la media se encuentra entre 16 y 18 por lo que es pertinente observar si hay diferencias significativas.

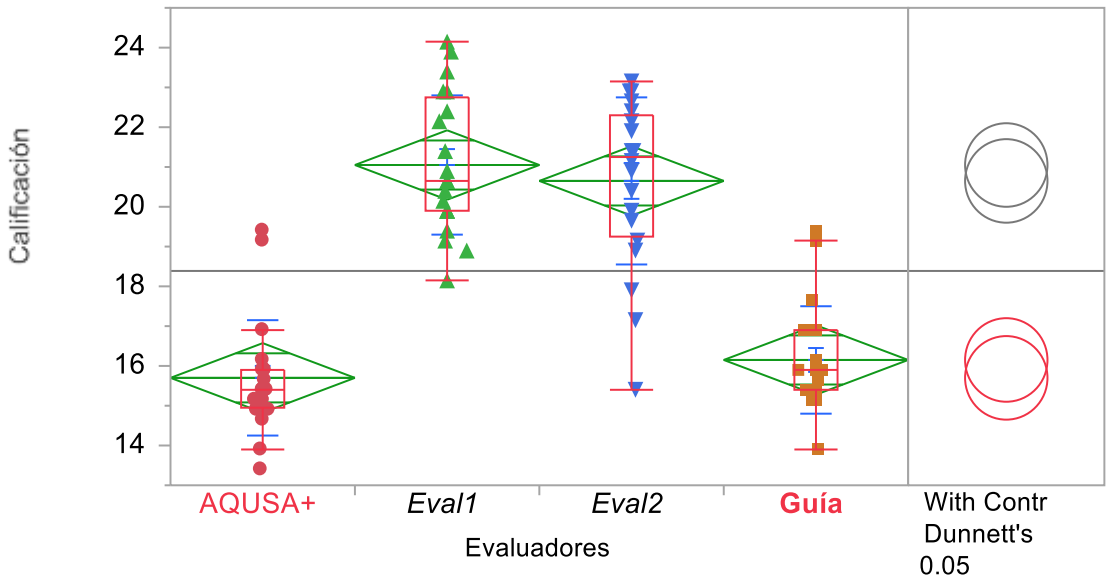

Figura 33. Diagrama de cajas MID

En *Managed inventory Database* las diferencias en el diagrama son notables: se observa cómo AQUSA+ y la guía se encuentran alineadas en sus promedios debajo de la media del proyecto. Por su parte los evaluadores también se encuentran alineados - pero en la parte superior.

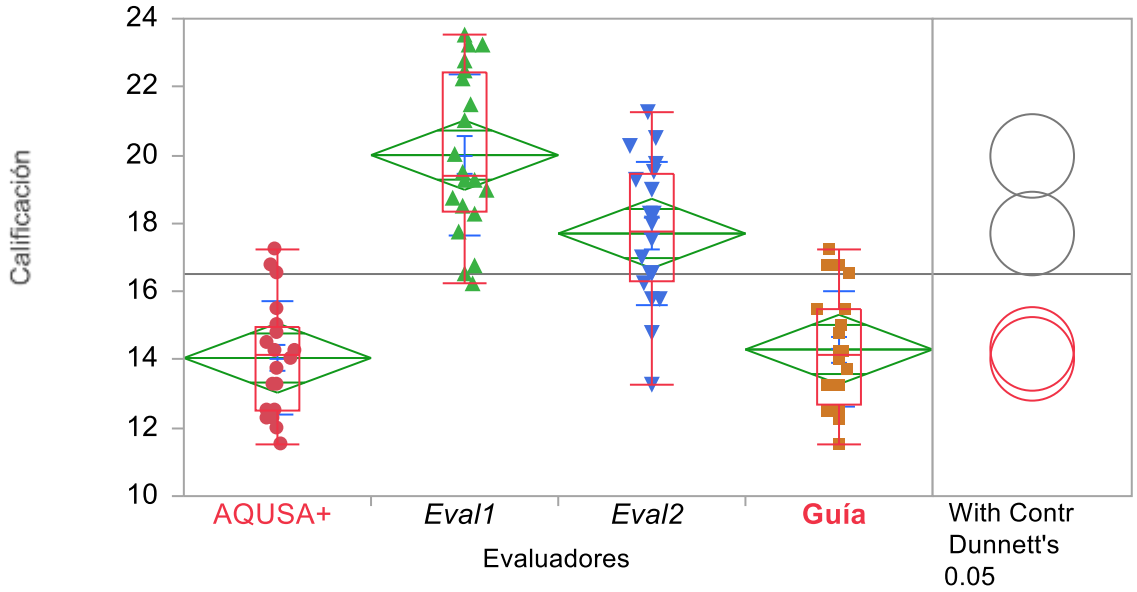

Figura 34. Diagrama de cajas TR.

Similar a los casos anteriores *Traffic* muestra las medias de la guía y AQUSA+ más alineadas que las de los evaluadores, ambas debajo de la media del proyecto.

4.1.1.2 Análisis de varianza

Con los resultados de la prueba de Barlett se determinó que se podía aplicar ANOVA de dos factores. Los resultados para cada uno de los proyectos se encuentran en las tablas 11, 12, 13 y 14.

Análisis de Varianza – *Accounts Receivable*

Fuente	**Grados de Libertad**	**Suma de cuadrados**	**Cuadrado de la media**	**F**	**Valor p**
Evaluadores	3	366.43750	122.146	41.3654	<0.0001*
Historia de usuario	19	140.63750	7.402	2.5067	0.0039*
Error	57	168.31250	2.953		
C. Total	79	675.38750			

Tabla 11. ANOVA para AR.

Análisis de Varianza – *Item Master*

Fuente	**Grados de Libertad**	**Suma de cuadrados**	**Cuadrado de la media**	**F**	**Valor p**
Evaluadores	3	32.05000	10.6833	2.9931	<0.0383*
Historia de usuario	19	489.25000	25.7500	7.2143	0.0001*
Error	57	203.45000	3.5693		
C. Total	79	724.75000			

Tabla 12. ANOVA para IM.

Análisis de Varianza – *Managed Inventory Database*

Fuente	**Grados de Libertad**	**Suma de cuadrados**	**Cuadrado de la media**	**F**	**Valor p**
Evaluadores	3	488.73750	162.913	42.8890	<.0001*

Fuente	Grados de Libertad	Suma de cuadrados	Cuadrado de la media	F	Valor p
Historia de usuario	19	275.7350	14.513	3.8206	<.0001*
Error	57	216.51250	3.798		
C. Total	79	980.98750			

Tabla 13. ANOVA para MID.

Análisis de Varianza – *Traffic*

Fuente	Grados de Libertad	Suma de cuadrados	Cuadrado de la media	F	Valor p
Evaluadores	3	490.6375	163.546	31.6419	<.0001*
Historia de usuario	19	428.7375	22.565	4.3658	<.0001*
Error	57	294.6125	5.169		
C. Total	79	1213.9875			

Tabla 14. ANOVA para TR.

Como se observa en cada tabla, el valor p es menor a 0.05 en todos los casos; solo *Item Master* registró un valor considerablemente mayor que los demás. Sin embargo, dicha condición no impide que se pueda concluir que existe evidencia estadística para rechazar la hipótesis nula; es decir, hay diferencia estadística en los promedios de calificación de los evaluadores. Por lo tanto, también se puede dar por cumplida la hipótesis alternativa.

4.1.1.3 Análisis Dunnet

Dado que existe una diferencia estadística en al menos uno de los tratamientos, se procedió a calcular Dunnet para cada proyecto, utilizando la guía como el grupo de control, con miras a identificar cuáles tratamientos son equivalentes y cuáles son diferentes.

La significancia se evalúa de acuerdo con el valor p; conforme este se acerque a 1 mayor es la equivalencia estadística con el grupo control.

***Accounts Receivable* - Matriz de límite LSD (*Least Significant difference*)**

Level	Abs(Dif)-LSD	p-Value
Eval1	3.039	<0.0001*
Eval2	2.239	<0.0001*
Guía	-1.310	1.00000
AQUSA+	-0.760	0.61890

Tabla 15.. Análisis Dunnet para AR.

***Managed Inventory Database* - Matriz de límite LSD (*Least Significant difference*)**

Evaluador	Abs(Dif)-LSD	p-Value
Eval1	3.413	<0.0001*
Eval2	3.013	<0.0001*
Guía	-1.490	1.00000
AQUSA+	-1.040	0.80700

Tabla 16. Análisis Dunnet para MID.

***Traffic* - Matriz de límite LSD (*Least Significant difference*)**

Evaluador	Abs(Dif)-LSD	p-Value
Eval1	3.413	<0.0001*
Eval2	3.013	<0.0001*
Guía	-1.490	1.00000
AQUSA+	-1.040	0.80700

Tabla 17. Análisis Dunnet para TR.

***Item Master* - Matriz de límite LSD (*Least Significant difference*)**

Evaluador	Abs(Dif)-LSD	p-Value
Eval2	-0.04	0.0589
Eval1	-0.89	0.6822
Guía	-1.44	1.0000
AQUSA+	-1.19	0.9534

Tabla 18. Análisis Dunnet para IM.

En *Accounts Receivable, Managed Inventory Database y Traffic* el promedio de calificación de AQUSA+ es estadísticamente equivalente al de la guía. Por su parte, el promedio de ambos evaluadores es diferente de forma significa.

En *Item Master* se presenta un comportamiento diferente. Acá, el evaluador 2 y AQUSA son estadísticamente equivalentes al grupo control, mientras que el evaluador 1 es diferente de manera significativa.

4.1.2 Análisis de falsos positivos

En la figura 31 se presenta un gráfico con los totales de falsos positivos por proyecto, agrupados por la regla del criterio de calidad en evaluación.

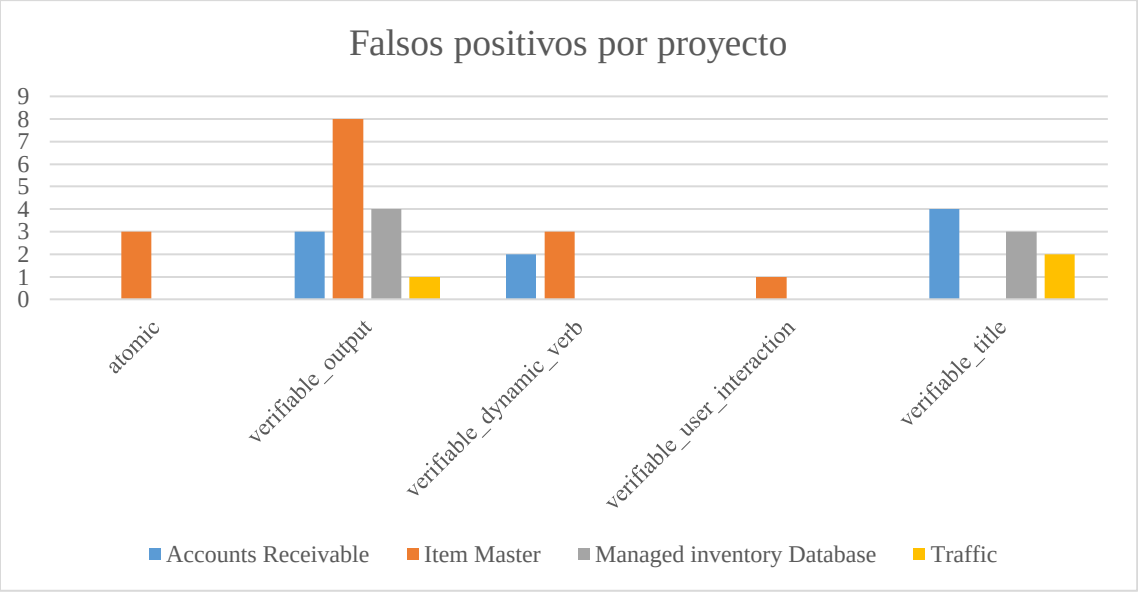

Figura 31. Gráfico de falsos positivos en AQUSA+.

Como se observa, la mayor cantidad de falsos positivos se encontró en *Item Master*, siendo la regla *verifiable_output* la que presentó mayores dificultades para determinar si realmente el resultado era observable. En total, para todos los proyectos se registraron 57 errores de este tipo, donde 16 de ellos fueron falsos positivos, para una precisión del 71.92 %.

Item Master fue el único en reportar falsos positivos en la descripción de la historia de usuario, específicamente en la regla *atomic*. Por su parte, la regla *verifiable_dynamic_verb* presentó cinco falsos positivos. En estos casos el criterio de aceptación representaba una acción mediante palabras que pueden ser etiquetadas como verbos en tercera persona, tiempo presente o bien, sustantivo; por ejemplo, la palabra *'requests'.*

En cuanto a *verifiable_title,* se identificaron falsos positivos con casos similares a la regla anterior, por ejemplo, la palabra *'Bill'* que, en el contexto en el que se encontraba, se refería a la acción 'cobrar' y estaba escrita con su primera letra en mayúscula por ser la primera de la oración, lo cual provocó que la herramienta la identificara como un nombre propio.

La suma total de errores detectados por AQUSA+ para los cuatro proyectos fue de **749,** de los cuales **34** fueron identificados como falsos positivos; es decir, el **4.53%** por lo tanto, AQUSA+ presentó un **95.47 %** de precisión en su evaluación.

4.1.2.1 Análisis Benchmark

Después de someter el *benchmark* al análisis de AQUSA+, se obtuvo el registro de errores siguiente:

Figura 32. Benchmark en descripciones.

No se encontraron errores en las descripciones de las 20 historias de usuario, como se muestra en la imagen de la figura 32, en el apartado *total issues:*

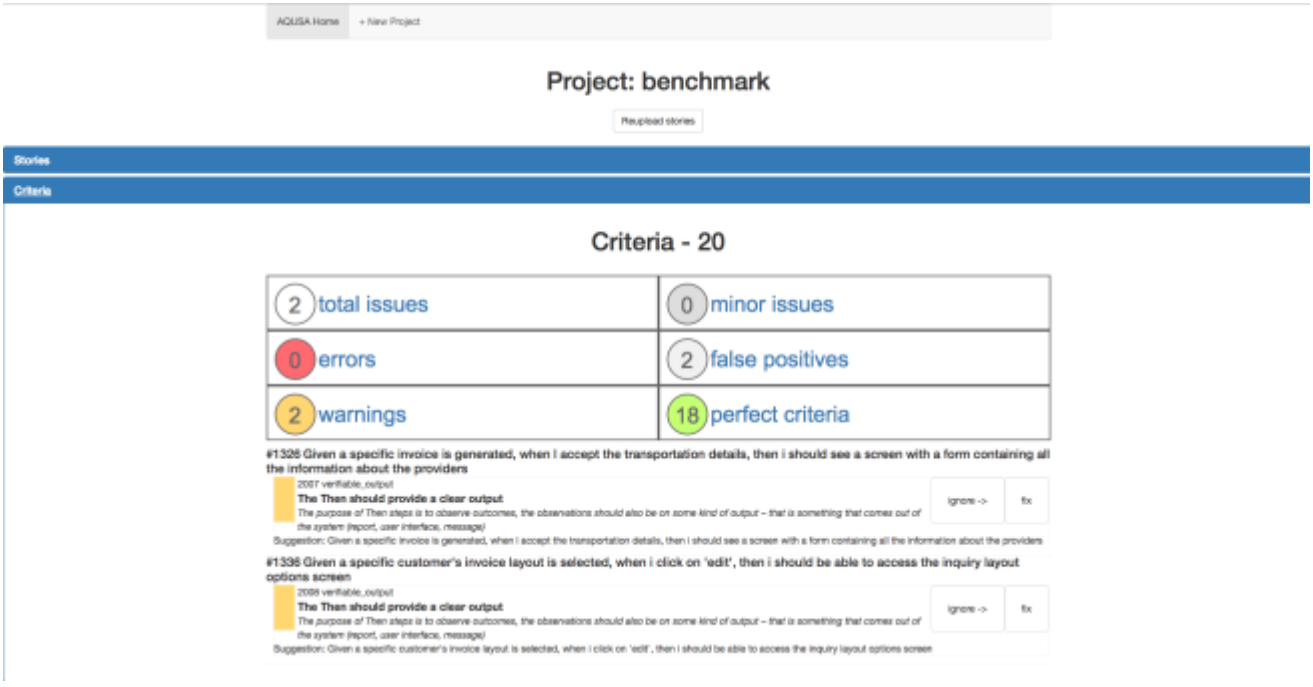

Figura 33. Benchmark en criterios de aceptación.

En el componente de criterio de aceptación se encontraron dos falsos positivos en los siguientes escenarios:

1. *"Given a specific invoice is generated, When I accept the transportation details, Then I should see a screen with a form containing all the information about the providers"*

   Acá el tipo de error es registrado por la regla *verifiable_output;* es decir, erróneamente identifica que no existe un resultado observable, pues se indica en el criterio de aceptación que una pantalla desplegará un formulario que contiene toda la información acerca de los proveedores.
2. *"Given a specific customer's invoice layout is selected, When I click on 'Edit', Then i should be able to access the inquiry layout options screen"*.

En este falso positivo, al igual que en el anterior, existe un resultado observable; se trata de una nueva interfaz que aparece después de hacer click en el botón *'Edit'*.

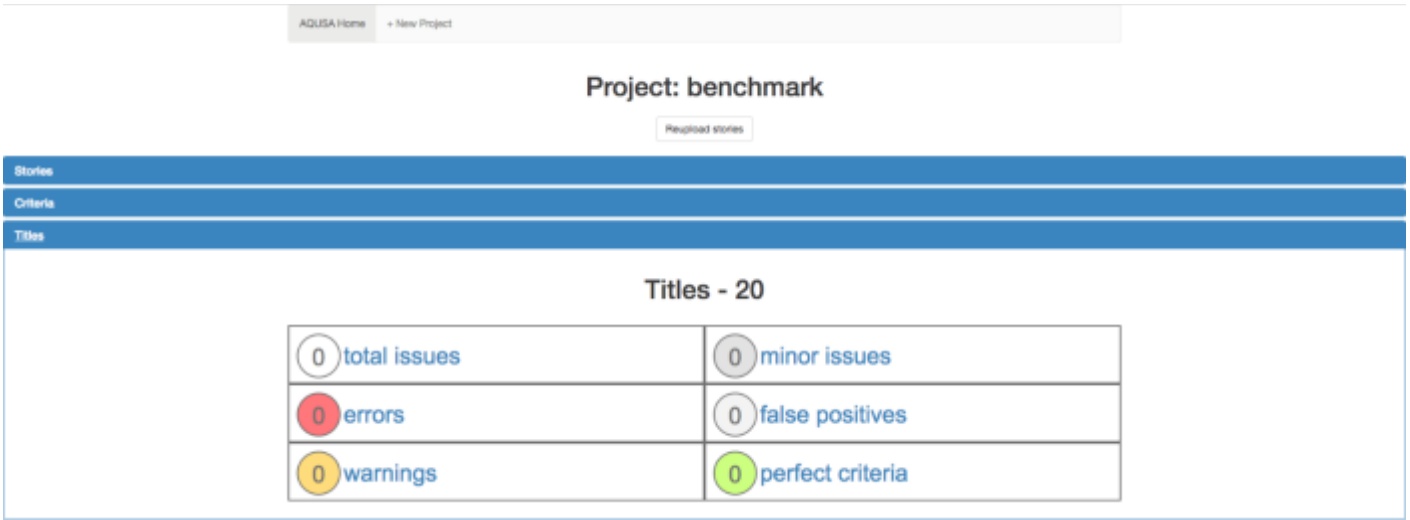

Figura 34. Benchmark en título.

Por su parte, en el título, la aplicación del *benchmark* reflejó los resultados esperados y no reportó errores.

4.1.3 Síntesis de los resultados

AQUSA+ fue capaz de detectar una mayor cantidad de errores a lo largo de los cuatro proyectos a los que fue sometido, en comparación con los cuatro expertos humanos que evaluaron la misma cantidad de historias de usuario. Mediante una evaluación estadística, se confirmó que existe una diferencia entre los evaluadores y AQUSA+ en tres de los cuatro proyectos y que los resultados de la herramienta son estadísticamente equivalentes al grupo control en todos los proyectos; es decir, al análisis completo y correcto de todas las historias de usuario, realizado por el investigador conforme a criterios estándar.

La precisión de la herramienta es de un 95.47 %, debido a que se encontraron falsos positivos en la evaluación de los proyectos muestra y en la aplicación del *benchmark*, siendo la regla *verifiable_output* la que presenta mayor cantidad de incidencias.

5 Conclusiones

Esta investigación fue capaz de generar no solo un modelo de calificación, sino de extender y mejorar el prototipo AQUSA para análisis sintáctico y pragmático. La siguiente tabla presenta una lista de funcionalidades que convirtieron a AQUSA en AQUSA+:

Funcionalidad	**Criterio de calidad**	**AQUSA**	**AQUSA+**
Descripción			
	Atómica	✓	✓
	Mínima	✓	✓
	Uniforme	✓	✓
	Única	✓	✓
	Bien Formada	✓	✓
Criterio de aceptación			
	Atómico		✓
	Mínimo		✓
	Uniforme		✓
	Único		✓
	Bien Formado		✓
	Verificable		✓
Título			
	Verificable		✓
Uso de ontologías			✓
Similitud semántica			✓

Tabla 18. Comparación AQUSA vs. AQUSA+

En el capítulo 1 se describió la problemática de los requerimientos ágiles y cómo la necesidad de entregar *software* en períodos cortos de tiempo puede incidir en la acumulación de deuda técnica. Un primer aporte de Lucassen *et al.,* (2015) en el análisis de historias de usuario

utilizando procesamiento de lenguaje natural (*NLP*) generó el prototipo de AQUSA, que evalúa cinco criterios de calidad en la descripción, únicamente.

AQUSA+, por su parte, completó el análisis de las historias de usuario mediante la evaluación del criterio de aceptación y el título permitiendo un escrutinio integral de los tres textos que conforman este tipo de requerimientos. La creación y utilización de una rúbrica para los evaluadores permitió delimitar la calificación y orientar los objetivos de acuerdo con los criterios de calidad que un conjunto de historias de usuario debe cumplir y establecer una base para la comparación de los resultados.

Los experimentos realizados mostraron que la calificación otorgada por AQUSA+ fue equivalente a la muestra estándar; es decir, la herramienta fue capaz de detectar más errores que los expertos humanos con una precisión del 95.47 % , lo cual representa una mejora en la precisión obtenida por Lucassen *et al.* (2015), que fue del 71 %.

En solo uno de cuatro proyectos, uno de los expertos fue capaz de otorgar una calificación similar a la de la muestra estándar y AQUSA+; es decir, la herramienta provee la capacidad suficiente de apoyar la labor del experto en la evaluación de las historias de usuario, previo a ser entregadas al equipo.

Al igual que en el modelo de referencia, se concluye que cada conjunto de historias es diferente en su calidad y en la forma en la que son escritas. La manera en que cada autor crea estos requerimientos influye la incidencia de un tipo de error, por ejemplo, un dueño de producto puede excederse al agregar datos que cree necesarios en el criterio de aceptación, a la vez que se genera mayor cantidad de errores tipo ‘Mínimo’.

En cuanto a los criterios de calidad, 'Verificable' fue el que obtuvo mayor cantidad de falsos positivos, específicamente en la regla '*verifiable*_output', donde la precisión es de aproximadamente 72 %.

El uso del algoritmo de similitud semántica permitió la identificación de errores en la observabilidad de los resultados. Además estableció el primer acercamiento del prototipo de AQUSA+ con una ontología. Sin embargo, para aumentar su precisión, se requiere una mejora en la lógica del algoritmo, de manera que sea más flexible con respecto de las palabras que se pueden utilizar para denotar una salida.

5.1 Trabajo futuro

Esta investigación culminó con la evaluación de conjuntos de historias en sus tres componentes principales. Sin embargo, se han identificado algunos puntos importantes que pueden hacer de AQUSA+ una herramienta más robusta:

- **Análisis semántico**: implementar el análisis semántico en AQUSA representa el reto más grande, pues requiere tomar en cuenta el contexto, para lo cual una ontología robusta será importante. Herramientas como *Protegé* pueden ser de gran utilidad para este propósito.
- **Historias de usuario en español**: la actual implementación de AQUSA+ solo evalúa historias de usuario en inglés; un aporte que permita traer esta lógica al idioma español implicaría modificaciones interesantes al modelo y las reglas. También sería necesario incluir ontologías en español.
- **Entrada de datos:** actualmente AQUSA+ funciona mediante la carga de un archivo *.csv* con las historias de usuario. Una mejora a la herramienta sería que esta funcione en un *software* para gestionar portafolios de administración de proyectos ágiles como *JIRA, TFS o Rally*. De esta forma, el análisis sería inmediato y ayudaría a prevenir la acumulación de deuda de documentación desde que el dueño de producto escribe la historia.
- **Parametrización:** existen algunas reglas que requieren una refactorización en su código para ser más eficientes. Por ejemplo, en *verifiable_output*, la lista de palabras que se consideran salidas observables podría estar en un archivo por cargar de la misma forma que los archivos con las historias de usuario.

Conforme los dominios cambian, las formas de representar salidas pueden ser otras. Así, una adaptación de cada contexto potenciaría la precisión de AQUSA+ en la detección de errores de este tipo.

Otro aspecto que se podría parametrizar es el formato aceptado en el criterio de aceptación; es decir, que en vez de utilizar "*Given, When, Then"* el usuario pueda determinar un formato diferente y que, con base en ese otro conjunto de palabras, se realice el *chunking* y el análisis sintáctico.

- **Sugerencia:** AQUSA+ tiene un campo en el registro de errores donde brinda sugerencias para corregir el error encontrado en el componente. No todas las reglas tienen esta funcionalidad implementada. Esta característica iría de la mano con el botón *'fix'* que automáticamente corregiría el error y produciría una versión mejorada con base en la sugerencia.

  En esta misma implementación, se podría dar funcionalidad al botón *'ignore'* para ignorar la sugerencia y almacenar la historia revisada, pero sin correcciones - como parte del producto final.
- **Corrector ortográfico incluido:** AQUSA+ procesa como errores las palabras que son escritas incorrectamente, por ejemplo, el gerundio '*openning'* es catalogado como un verbo dinámico, pero en el caso de estar mal escrito la herramienta supone que ahí no existe un verbo y registra el error.

  Una versión mejorada de la herramienta debería detectar este problema y reportar el anterior como error ortográfico, pero debería etiquetar la palabra como un verbo.
- **Modelo de datos:** debe ser revisado para eliminar redundancia de datos entre las tablas de error. También la tabla *story* debería ser llamada *description*, debido a que es únicamente un componente de la historia de usuario.

6 Referencias bibliográficas

Beck, K., Sutherland, J. y Schwaber, K. (2001). *Manifesto for Agile Software Development*. Recuperado el 21 de junio de 2015 de http://www.agilemanifesto.org/

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *Journal Computer*, *21*(5), 61–72.

Cohn, M. (2004). *User Stories Applied For Agile* (13a ed.). California: Addison Wesley Longman Publishing Co. Inc.

Cunningham, W. (1992). *The WyCash Portfolio Management System*. (pp. 29–30). En *Proc. on Object-Oriented Programming systems, languages, and applications*.

Dautovic, A., Plösch, R., y Saft, M. (2014). *Automated Quality Defect Detection in Software Development Documents*. Recuperado de http://ceur-ws.org/Vol-708/sqm2011-dautovic-et-al-11-autoQualityDefectDetect.pdf

Deshapande, A. (2017, marzo 23). *Diving Into Natural Language Processing*. Recuperado el 6 de mayo de 2017 de https://dzone.com/articles/natural-language-processing-adit-deshpande-cs-unde?edition=286910&utm_source=Daily

Earls, C. (2005, octubre). *Civil Engineering Rubric*. Recuperado de https://www.cmu.edu/teaching/resources/Teaching/CourseDesign/Assessment-Grading/Rubrics/CivilEngineeringRubric.doc

Hardeniya, N. (2015). *NLTK Essentials* (1a ed.). Birmingham: PacktPub.

ISO/IEC/IEEE 26515. (2012, marzo 15). *Systems and software engineering - Developing user documentation in an agile environment*. IEEE. Recuperado de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6170923&isnumber=6170922

Jacobson, I., Booch, G. y Rumbaugh, J. (1999). *The Unified Software Development Process*. Massachusetts: Addison-Wesley Longman Publishing Co., Inc.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M. y Brinkkemper, S. (2015). *Forging High-Quality User Stories: Towards a Discipline for Agile Requirements*. Presentado en: *23rd IEEE International Requirements Engineering Conference*, Ottawa, Canada.

Madhuri, T., M.M, R. y Latesh, M. (2014). *Word net based Method for Determining Semantic Sentence Similarity through various Word Senses*. Recuperado de http://ltrc.iiit.ac.in/icon/2014/proceedings/File68-p127.pdf

Martin, R. (2008, noviembre 27). *The Truth about BDD. Uncle Bob Consulting LLC.* Recuperado de https://sites.google.com/site/unclebobconsultingllc/the-truth-about-bdd

Mayr, A., Plösch, R. y Körner, C. (2014). *A Benchmarking-based Model for Technical Debt Calculation* (pp. 305–314). Presentado en *2014 14th International Conference on Quality Software*, Dallas, TX, USA.

McConnell, S. (2008, junio). *Managing Technical Debt.* Construx.com. Recuperado de http://www.construx.com/Resources/White_Papers/Managing_Technical_Debt/

Montgomery, D. (2008). *Design and Analysis of Experiments*. John Wiley & Sons. Recuperado de http://books.google.co.cr/books?id=kMMJAm5bD34C

Mooz, H., y Forsberg, K. (2001). *A visual explanation of development methods and strategies including the Waterfall, Spiral, Vee, Vee+, and Vee++ model*s. Center for Systems Management.

Morrison, P. (2011, Diciembre). *Managing Technical Debt. North Carolina State University*. Recuperado de https://es.scribd.com/doc/76265333/Measuring-Technical-Debt-Final?gad=otd

North, D. (2006, junio 5). Behavior Modification: The evolution of behavior driven development, *2006*(3). Recuperado de http://www.stickyminds.com/better-software-magazine/behavior-modification

Plank, B., Sauer, T., y Schaefer, I. (2012). *Supporting Agile Software Development by Natural Language Processing* (pp. 62–66). En 2012 J*oint Workshop on Intelligent Methods for Software System Engineering*, Montpellier, France. Recuperado a partir de https://sites.google.com/site/jimse2012/

Sall, J. (2017). *JMP Statistical Discovery*. SAS. Recuperado de https://www.jmp.com/en_us/software/download-jmp-free-trial.html

Scacchi, W. (2001, febrero). *Process Models in Software Engineering.* En Encyclopedia of Software Engineering, 2nd Edition. Irvine, California.

Soares, H. , Alves, N., Alves, N., Mendonça, M. y Spínola, R. (2015). *Investigating the Link between User Stories and Documentation Debt on Software Projects* (pp. 385–390). Presentado en *2015 12th International Conference on Information Technology - New Generations*, USA, Las Vegas, Nevada.

Toutanova, K., Klein, D., Manning, C. y Singer, Y. (2003). *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network*, 252–259.

Version One. (2014). *9th Annual State of Agile™ Survey*. *Version One*. Recuperado de http://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf

Wake, B. (2003, agosto 17). *Invest In Good Stories and Smart Tasks.* Recuperado el 27 de julio de 2015, a partir de http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

Wohlin, C., Host, M. y Henningson, K. (2003). Empirical Research Methods in Software Engineering. *Empirical Methods and Studies in Software Engineering: Experiences from*, 7–23.

Wu, Z., y Palmer, M. (1994). Verbs semantics and lexical selection. En *Proceedings of the 32nd Meeting of Association of Computational Linguistics* (pp. 33–138).

Yli-Huumo, J. (2014). *Evaluating and managing technical debt in software development lifecycle* (Vol. 2, pp. 26–30). Presentado en *15th International Conference, PROFES* 2014, Helsinki, Finland.

Zeaaraoui, A., Bougroun, Z., Bouchentouf, T. y Belkasmi, M. (2013). *User Stories Template for Object-Oriented Applications*. Presentado en *Third International Conference on Innovative Computing Technology (INTECH)*. Recuperado de http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6653681

7 Anexos

7.1 Anexo A: Código fuente de las reglas de validación

A continuación se presentan los fragmentos de código que corresponden a las reglas de AQUSA+ que no se basan en funcionalidades ya existentes de AQUSA, sino que implicaron una lógica completamente nueva de implementar.

7.1.1 atomic_criteria_one_feature

```
def atomic_one_feature_rule(criteria):
  stories = Criteria.query.filter_by(story_id=criteria.story_id).all()
  repeated_when = []
  if len(stories) > 1:
    for x in stories:
      repeated_when.append(x.when == criteria.when)
    return repeated_when.count(True) != len(stories)
  else:
    return False
```

7.1.2 verifiable_user_interaction

```
def user_interaction_rule(criteria, kind):
  try:
    given_cleaned = AnalyzerCriteria.clean_component(criteria.given.upper())
    result = AnalyzerCriteria.content_chunk(given_cleaned, kind)
    no_interaction = False
    for x in result:
      if hasattr(x, 'label'):
        if 'PRP' in x.label().upper(): no_interaction = True
        elif 'NNP' in x.label().upper(): no_interaction = True
        elif 'NNPS' in x.label().upper(): no_interaction = True
      else:
        if x[1] == 'PRP': no_interaction = True
        elif x[1] == 'NNP': no_interaction = True
        elif x[1] == 'NNPS': no_interaction = True
    return no_interaction
  except:
    return False
```

7.1.3 verifiable_stative_verb

```
def stative_verb_rule(criteria, kind):
  no_state = True
  if criteria.given is not None:
      try:
        given_cleaned = AnalyzerCriteria.clean_component(criteria.given.upper())
        result = AnalyzerCriteria.content_chunk_given(given_cleaned, kind)
        for x in result:
          try:
            if hasattr(x, 'label'):
              if len(x[0][0][1]) > 1:
                if 'VB' == x[0][0][1]: no_state = False
                elif 'VBZ' == x[0][0][1]: no_state = False
                elif 'VBN' == x[0][0][1]: no_state = False
                elif 'VBG' == x[0][0][1]: no_state = False
                elif 'VBD' == x[0][0][1]: no_state = False
              else:
                if 'VB' == x[0][1]: no_state = False
                elif 'VBZ' == x[0][1]: no_state = False
                elif 'VBN' == x[0][1]: no_state = False
                elif 'VBG' == x[0][1]: no_state = False
                elif 'VBD' == x[0][1]: no_state = False

            else:
              if 'VB' == x[1].upper(): no_state = False
              elif 'VBZ' == x[1].upper(): no_state = False
              elif 'VBN' == x[1].upper(): no_state = False
              elif 'VBG' == x[1].upper(): no_state = False
```

```
            elif 'VBD' == x[1].upper(): no_state = False
          except IndexError as e:
            if 'VB' == x[0][1]: no_state = False
            elif 'VBZ' == x[0][1]: no_state = False
            elif 'VBN' == x[0][1]: no_state = False
            elif 'VBG' == x[0][1]: no_state = False
            elif 'VBD' == x[0][1]: no_state = False
        return no_state
      except:
        return no_state
```

7.1.4 verifiable_dynamic_verb

```
def dynamic_verb_rule(criteria, kind):
  no_action = True
  if criteria.when is not None:
    when_cleaned = AnalyzerCriteria.clean_component(criteria.when.upper())
    result = AnalyzerCriteria.content_chunk(when_cleaned, kind)
    for x in result:
      try:
        if hasattr(x, 'label'):
          if len(x[0][0][1]) > 1:
            if 'VB' == x[0][0][1]: no_action = False
            elif 'VBD' == x[0][0][1]: no_action = False
            elif 'VBN' == x[0][0][1]: no_action = False
            elif 'VBP' == x[0][0][1]: no_action = False
            elif 'VBZ' == x[0][0][1]: no_action = False
          else:
            if 'VB' == x[0][1]: no_action = False
            elif 'VBD' == x[0][1]: no_action = False
            elif 'VBN' == x[0][1]: no_action = False
            elif 'VBP' == x[0][1]: no_action = False
            elif 'VBZ' == x[0][1]: no_action = False
        else:
          if 'VB' == x[1].upper(): no_action = False
          elif 'VBD' == x[1].upper(): no_action = False
          elif 'VBN' == x[1].upper(): no_action = False
          elif 'VBP' == x[1].upper(): no_action = False
          elif 'VBZ' == x[1].upper(): no_action = False
      except IndexError as e:
        if 'VB' == x[0][1]: no_action = False
        elif 'VBD' == x[0][1]: no_action = False
        elif 'VBN' == x[0][1]: no_action = False
        elif 'VBP' == x[0][1]: no_action = False
        elif 'VBZ' == x[0][0]: no_action = False
  return no_action
```

7.1.5 verifiable_output

```
def semantic_similarity(word):
  w1=""
  w2=""
  is_similar = True
  output_list = ['output','outcome','report','interface','message', 'email',
'screen', 'window', 'show', 'content', 'result']
  try:
    for x in output_list:
      w1 = wordnet.synset(x+'.n.01')
      w2 = wordnet.synset(word[0]+'.n.01')
      if w1.wup_similarity(w2) > 0.34: is_similar = False
  except WordNetError as e:
    is_similar = True
  return is_similar


def output_rule(criteria, kind):
  if not criteria.then == None:
    then_cleaned = AnalyzerCriteria.clean_component(criteria.then.upper())
    result = AnalyzerCriteria.content_chunk(then_cleaned, kind)
    no_output = True
    noun_list=['NN','NNS']
    for x in result:
      try:
        if hasattr(x, 'label'):
          if len(x[0][0][1]) > 1:
            if x[0][0][1] in noun_list: no_output =
AnalyzerCriteria.semantic_similarity(x[0])
          else:
            if x[0][1] in noun_list: no_output =
AnalyzerCriteria.semantic_similarity(x[0])
        else:
          if x[1].upper() in noun_list: no_output =
AnalyzerCriteria.semantic_similarity(x[0])
      except IndexError as e:
        if x[0][1] in noun_list:no_output =
AnalyzerCriteria.semantic_similarity(x[0])
    return no_output
```

7.1.6 verifiable_title

```
def verifiable_rule(title):
  result = AnalyzerTitle.content_chunk(title.text)
  not_verifiable = True
  for x in result:
    try:
      if hasattr(x, 'label'):
        if len(x[0][0][1]) > 1:
          if 'VB' == x[0][0][1]: not_verifiable = False
          elif 'VBZ' == x[0][0][1]: not_verifiable = False
          elif 'VBN' == x[0][0][1]: not_verifiable = False
          elif 'VBG' == x[0][0][1]: not_verifiable = False
          elif 'VBD' == x[0][0][1]: not_verifiable = False
          elif 'VBP' == x[0][0][1]: not_verifiable = False
        else:
          if 'VB' == x[0][1]: not_verifiable = False
          elif 'VBZ' == x[0][1]: not_verifiable = False
          elif 'VBN' == x[0][1]: not_verifiable = False
          elif 'VBG' == x[0][1]: not_verifiable = False
          elif 'VBD' == x[0][1]: not_verifiable = False
          elif 'VBP' == x[0][1]: not_verifiable = False
      else:
        if 'VB' == x[1].upper(): not_verifiable = False
        elif 'VBZ' == x[1].upper(): not_verifiable = False
        elif 'VBN' == x[1].upper(): not_verifiable = False
        elif 'VBG' == x[1].upper(): not_verifiable = False
        elif 'VBD' == x[1].upper(): not_verifiable = False
        elif 'VBP' == x[1].upper(): not_verifiable = False
    except IndexError as e:
      if 'VB' == x[0][1]: not_verifiable = False
      elif 'VBZ' == x[0][1]: not_verifiable = False
      elif 'VBN' == x[0][1]: not_verifiable = False
      elif 'VBG' == x[0][1]: not_verifiable = False
      elif 'VBD' == x[0][1]: not_verifiable = False
      elif 'VBP' == x[0][1]: not_verifiable = False
  return not_verifiable
```

7.2 Anexo B: Resultados de calificación por proyecto

7.2.1 Accounts Receivable

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	2	2	2	2	2	2	2	2	2	2	2
2	1	2	2	2	2	2	2	2	2	2	2	2
3	2	2	1	2	2	2	2	1	1	2	2	1
4	1	2	2	2	2	2	2	2	2	2	2	2
5	1	1	2	2	2	2	2	2	1	1	2	2
6	1	1	2	2	2	1	2	2	0	2	2	2
7	1	1	2	2	2	2	2	2	2	2	1	2
8	1	1	2	2	2	2	2	2	2	2	2	2
9	1	1	1	1	2	2	2	2	2	2	2	1
10	2	1	2	2	2	2	2	2	1	2	2	2
11	2	1	2	2	2	2	2	1	1	2	2	2
12	1	1	2	2	2	2	2	2	0	2	2	2
13	1	1	2	2	2	2	2	2	0	2	2	2
14	1	1	2	2	2	2	2	2	2	2	2	2
15	2	1	2	2	2	2	2	2	2	1	2	2
16	1	1	2	2	2	1	2	2	1	2	1	2
17	1	1	2	2	2	2	2	2	2	2	2	2
18	1	1	2	2	2	2	2	2	2	2	2	2
19	2	1	2	2	2	2	2	2	2	2	2	2
20	1	1	2	2	2	2	2	2	2	2	2	2

Figura B1. Evaluación de AR por Oscar Alfaro.

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	2	2	2	2	2	2	2	2	1	2	2
2	1	2	2	2	2	2	2	2	2	2	2	2
3	2	2	2	2	2	2	2	2	1	1	2	2
4	1	2	1	2	2	2	1	2	2	1	2	1
5	1	1	2	1	2	2	1	2	1	1	2	2
6	1	2	2	2	2	2	1	2	0	1	2	1
7	1	1	2	1	1	2	1	2	2	1	2	1
8	1	2	2	2	2	2	1	2	2	1	2	2
9	1	1	1	1	1	2	2	2	2	2	2	2
10	2	2	2	2	2	2	1	2	1	1	2	2
11	2	2	2	2	2	2	2	2	1	1	2	2
12	1	2	2	2	2	2	1	2	0	1	2	2
13	1	2	2	2	2	2	1	2	0	1	2	2
14	1	1	2	1	1	2	2	2	2	1	2	2
15	2	2	2	2	2	2	2	2	2	2	2	2
16	1	2	2	2	2	2	1	2	1	1	2	1
17	1	2	2	2	2	2	2	2	2	2	2	2
18	1	1	2	1	1	2	1	2	2	1	2	2
19	2	2	2	2	2	2	2	2	2	2	2	2
20	1	2	2	2	2	2	2	2	2	1	2	2

Figura B2. Evaluación de AR por Juan Blanco.

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	1	2	1	1	2	2	0	2	0	2	1
2	2	1	2	2	1	2	2	2	2	0	2	1
3	0	1	2	2	1	2	1	0	1	0	2	1
4	2	1	0	2	1	2	0	0	2	0	2	1
5	2	1	2	0	1	2	2	2	1	1	2	1
6	2	2	2	2	2	2	1	2	0	1	2	2
7	0	1	2	2	1	2	1	1	2	1	2	2
8	2	1	2	2	2	2	0	0	2	0	2	0
9	0	1	0	0	1	2	2	2	2	1	2	2
10	2	1	2	2	1	2	2	2	1	1	2	1
11	2	1	2	2	1	2	2	0	1	0	2	1
12	2	1	0	2	1	2	1	2	0	0	2	1
13	2	1	0	2	1	2	2	2	0	1	2	1
14	2	1	2	1	1	2	1	2	2	1	2	1
15	2	1	2	2	1	2	1	0	2	1	2	1
16	2	1	2	1	1	2	1	2	1	2	2	1
17	2	1	2	2	1	2	1	1	2	2	2	1
18	2	2	0	1	1	2	0	0	2	0	2	1
19	0	2	0	1	1	2	1	2	2	0	2	2
20	2	1	2	2	2	2	2	0	2	0	2	1

Figura B3. Evaluación de AR con AQUSA+.

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	1	2	1	1	2	2	0	2	0	2	1
2	2	1	2	2	1	2	2	2	2	0	2	1
3	2	1	2	2	1	2	1	0	1	0	2	1
4	2	1	0	2	1	2	0	0	2	0	2	1
5	2	1	2	0	1	2	2	2	1	1	2	1
6	2	2	2	2	2	2	1	2	0	1	2	2
7	2	1	2	2	1	2	1	1	2	1	2	2
8	2	1	2	2	2	2	0	0	2	0	2	0
9	2	1	0	0	1	2	2	2	2	1	2	2
10	2	1	2	2	1	2	2	2	1	1	2	1
11	2	1	2	2	1	2	2	0	1	1	2	1
12	2	1	0	2	1	2	1	2	0	0	2	1
13	2	1	0	2	1	2	2	2	0	1	2	1
14	2	1	2	1	1	2	1	2	2	2	2	1
15	2	1	2	2	1	2	1	0	2	2	2	1
16	2	1	2	1	1	2	1	2	1	2	2	1
17	2	1	2	2	1	2	1	1	2	2	2	1
18	2	2	0	1	1	2	0	0	2	0	2	1
19	2	2	0	1	1	2	1	2	2	0	2	2
20	2	1	2	2	2	2	2	0	2	0	2	1

Figura B4. Guía o Grupo Control AR.

7.2.2 Item Master

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	El Título describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	2	1	2	1	2	2	1	2	1	2	1
2	2	1	1	2	1	2	2	0	2	1	2	1
3	2	2	0	0	1	2	2	1	2	1	2	0
4	2	1	2	2	1	2	0	0	2	1	2	1
5	2	2	2	2	1	2	2	2	2	1	2	2
6	2	2	2	2	1	2	2	2	2	1	2	2
7	0	1	0	1	1	2	1	1	1	1	2	1
8	2	2	2	2	1	2	1	2	2	1	2	2
9	2	2	2	2	1	2	2	1	1	1	2	1
10	1	2	0	0	1	2	1	0	2	1	2	1
11	2	1	2	2	1	2	2	2	0	2	2	2
12	1	1	1	1	1	2	1	1	2	1	2	1
13	0	2	2	2	1	2	2	2	2	1	2	2
14	0	2	2	0	1	2	1	1	2	1	2	2
15	1	2	1	1	1	2	2	2	2	1	2	2
16	2	2	1	2	1	2	1	1	1	1	2	2
17	2	0	2	2	1	2	1	1	2	1	2	1
18	1	2	1	1	1	2	1	0	1	1	2	1
19	2	1	0	1	1	2	1	0	1	1	2	2
20	0	1	1	1	0	2	1	1	1	1	2	0

Figura B5. Evaluación de IM por Kevin Leandro

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	El Título describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	0	1	2	2	1	2	2	0	2	2	2	2
2	0	2	2	2	1	2	2	0	2	2	2	2
3	0	2	1	1	1	2	2	2	2	2	2	0
4	1	2	2	2	1	2	1	0	2	2	2	2
5	1	2	1	1	1	2	1	0	2	2	2	2
6	2	2	2	2	1	2	2	2	2	2	2	2
7	1	2	2	2	2	2	2	0	1	1	2	2
8	1	2	2	2	1	2	2	1	2	2	2	1
9	1	2	2	2	1	2	2	1	1	2	2	2
10	1	1	1	1	1	2	2	0	2	2	2	2
11	1	2	1	1	1	2	2	0	0	2	2	1
12	0	2	2	2	1	2	2	1	2	2	2	2
13	1	2	1	2	1	2	2	2	2	1	2	2
14	0	1	0	0	1	2	1	1	2	1	2	1
15	1	2	1	1	1	2	2	2	2	1	2	2
16	1	2	2	2	1	2	2	0	1	2	2	2
17	1	2	2	1	1	2	1	1	2	2	2	1
18	0	1	1	2	1	2	1	0	1	2	2	1
19	2	2	1	1	1	2	2	0	1	1	2	1
20	0	2	1	1	1	2	2	0	1	2	2	1

Figura B6. Evaluación de IM por Esteban Rodríguez

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	El Título describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	0	2	2	2	2	2	1	0	2	2	2	2
2	0	2	0	2	2	2	1	0	2	1	2	2
3	0	2	1	1	2	2	0	0	2	0	2	0
4	0	2	2	2	2	2	1	2	2	2	2	2
5	0	2	1	1	2	2	1	2	2	1	2	2
6	2	2	2	2	2	2	1	2	2	1	2	2
7	0	2	0	2	2	2	1	2	1	1	2	2
8	0	2	2	2	2	2	1	0	2	0	2	1
9	0	2	2	2	2	2	0	0	1	0	2	1
10	0	1	2	0	2	2	1	0	2	0	2	2
11	0	2	2	2	2	2	1	0	0	2	2	2
12	0	2	2	2	2	2	1	0	2	1	2	2
13	0	2	2	1	2	2	1	2	2	2	2	2
14	0	2	2	0	2	2	1	0	2	1	2	2
15	0	2	1	0	2	2	1	2	2	2	2	1
16	0	2	2	2	2	2	1	0	1	1	2	2
17	0	2	0	1	1	2	1	0	2	2	2	1
18	0	2	0	1	2	2	1	0	1	1	2	1
19	0	2	2	0	0	2	0	0	1	0	2	1
20	0	2	1	1	2	2	1	0	1	1	2	0

Figura B7. Evaluación de IM con AQUSA+

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	El Título describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	0	2	1	2	2	2	1	0	2	2	2	2
2	0	2	2	2	2	2	1	0	2	2	2	2
3	0	2	1	1	2	2	0	0	2	0	2	0
4	0	2	2	2	2	2	1	2	2	2	2	2
5	0	2	1	1	2	2	1	2	2	1	2	2
6	2	2	2	2	2	2	1	2	2	2	2	2
7	0	2	0	2	2	2	1	2	1	1	2	2
8	0	2	2	2	2	2	1	0	2	1	2	1
9	0	2	2	2	2	2	0	0	1	0	2	1
10	0	1	2	0	2	2	1	0	2	0	2	2
11	0	2	2	2	2	2	1	0	0	2	2	2
12	0	2	2	2	2	2	1	0	2	1	2	2
13	0	2	2	1	2	2	1	2	2	2	2	2
14	0	2	2	0	2	2	1	0	2	1	2	2
15	0	2	1	0	2	2	1	2	2	2	2	1
16	0	2	2	2	2	2	1	0	1	1	2	2
17	0	2	0	1	1	2	1	0	2	2	2	1
18	0	2	0	1	2	2	1	0	1	1	2	1
19	0	2	2	0	0	2	0	0	1	0	2	1
20	0	2	1	1	2	2	1	0	1	2	2	0

Figura B8. Guía o grupo control IM

7.2.3 Managed Inventory Database

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de hi	El Título describe una act	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	1	1	2	1	1	1	2	2	2	2	2
2	2	1	2	1	2	2	2	2	2	2	2	2
3	2	1	0	1	1	1	2	1	0	1	2	2
4	2	1	2	2	1	0	1	2	2	1	2	2
5	2	1	2	2	2	2	2	2	2	2	2	2
6	2	1	1	2	1	2	2	2	2	2	2	2
7	2	2	1	1	2	2	2	2	1	2	2	2
8	2	2	2	2	1	2	1	2	2	2	2	2
9	2	2	2	2	1	1	1	1	2	2	2	2
10	2	2	2	2	2	2	2	2	1	2	2	2
11	1	2	2	2	2	2	2	2	2	2	2	2
12	1	2	2	1	2	2	2	2	2	2	2	2
13	2	1	2	1	1	1	2	2	2	2	2	2
14	2	2	2	2	1	1	2	2	2	2	2	2
15	2	1	2	2	2	2	2	2	2	2	2	2
16	2	1	1	2	1	2	2	2	2	2	2	2
17	2	1	2	2	2	2	2	2	2	2	2	2
18	2	1	2	2	2	2	2	2	2	2	2	2
19	2	1	2	2	1	2	2	2	2	2	2	2
20	1	0	0	2	2	2	2	2	2	2	2	2

Figura B9. Evaluación de MID por Oscar Alfaro

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	El Título describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	1	1	1	1	2	2	2	2	2	2	2
2	2	2	2	1	1	2	2	1	2	1	2	2
3	2	0	0	0	1	2	1	1	0	1	1	1
4	2	1	2	1	1	2	2	2	2	2	2	2
5	2	1	2	1	2	2	2	2	2	2	2	2
6	2	2	2	1	2	2	2	2	2	2	2	2
7	2	0	1	1	1	2	1	2	1	1	2	2
8	2	2	2	2	2	2	2	2	2	2	2	2
9	2	2	2	2	2	2	2	2	2	2	2	2
10	2	2	2	2	2	2	1	1	1	1	2	1
11	1	2	1	1	2	2	2	2	2	2	2	2
12	1	2	2	2	2	2	2	1	2	2	1	2
13	2	2	2	1	2	2	2	2	2	2	2	2
14	2	2	2	2	2	2	2	2	2	2	2	2
15	2	2	2	2	2	2	2	2	2	2	2	2
16	2	2	1	1	1	2	2	2	2	2	2	2
17	2	2	2	1	2	2	2	2	2	2	2	2
18	2	1	1	1	1	2	1	1	2	1	2	2
19	2	2	2	1	1	2	2	2	2	2	2	2
20	1	1	1	0	1	2	2	2	2	2	2	2

Figura B10. Evaluación de MID por Juan Blanco

Rúbrica de evaluación AQUSA

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	El Título describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	0	1	2	0	1	2	1	2	2	1	2	1
2	2	1	2	1	1	2	1	0	2	0	2	1
3	2	1	2	0	2	2	1	0	0	1	2	1
4	0	1	2	1	1	2	1	2	2	0	2	1
5	0	1	2	1	1	2	1	2	2	1	2	1
6	0	1	2	1	1	2	1	2	2	0	2	1
7	0	1	2	0	2	2	1	0	1	0	2	1
8	2	1	2	1	2	2	1	2	2	0	2	1
9	0	1	2	1	2	2	1	2	2	0	2	1
10	2	1	2	2	1	2	1	0	1	1	2	1
11	0	1	0	1	1	2	1	2	2	0	2	1
12	0	1	2	2	2	2	1	2	2	0	2	1
13	2	1	2	1	1	2	1	0	2	0	2	1
14	0	1	2	2	1	2	1	2	2	1	2	1
15	2	1	2	1	1	2	1	2	2	0	2	1
16	0	1	0	1	2	2	1	0	2	0	2	1
17	2	1	2	0	2	2	1	2	2	0	2	1
18	2	1	2	0	2	2	1	0	2	0	2	1
19	2	1	2	1	2	2	1	2	2	1	2	1
20	2	2	2	0	2	2	1	2	2	1	2	2

Figura B11. Evaluación de MID con AQUSA+

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	El Título describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	0	1	2	0	1	2	1	2	2	2	2	1
2	2	1	2	1	1	2	1	0	2	0	2	1
3	2	1	2	0	2	2	1	0	0	1	2	1
4	0	1	2	1	1	2	1	2	2	0	2	1
5	0	1	2	1	1	2	1	2	2	1	2	1
6	0	1	2	1	1	2	1	2	2	1	2	1
7	0	1	2	0	2	2	1	0	1	0	2	1
8	2	1	2	1	2	2	1	2	2	0	2	1
9	2	1	2	1	2	2	1	2	2	0	2	1
10	2	1	2	2	1	2	1	0	1	1	2	1
11	2	1	0	1	1	2	1	2	2	0	2	1
12	2	1	2	2	2	2	1	2	2	0	2	1
13	2	1	2	1	1	2	1	0	2	0	2	1
14	0	1	2	2	1	2	1	2	2	1	2	1
15	2	1	2	1	1	2	1	2	2	1	2	1
16	0	1	0	1	2	2	1	0	2	0	2	1
17	2	1	2	0	2	2	1	2	2	0	2	1
18	2	1	2	0	2	2	1	0	2	0	2	1
19	2	1	2	1	2	2	1	2	2	1	2	1
20	2	2	2	0	2	2	1	2	2	1	2	2

Figura B12. Guía o grupo control MID

7.2.4 Traffic

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	1	1	1	1	1	2	2	1	1	2	2	2
2	0	2	1	1	2	2	2	2	1	2	2	2
3	1	2	2	2	2	2	2	2	2	2	2	1
4	1	1	0	0	1	2	2	1	0	2	2	1
5	0	2	2	2	2	2	2	2	1	2	2	2
6	2	2	2	2	2	2	2	2	2	2	2	2
7	0	2	2	2	2	2	1	1	2	1	2	2
8	1	2	2	2	2	2	2	2	2	2	2	2
9	2	2	2	2	2	2	2	2	2	2	2	2
10	2	1	1	1	1	2	1	2	2	2	2	2
11	1	1	2	2	2	2	2	2	2	2	2	2
12	0	1	1	1	1	2	2	1	2	2	2	2
13	1	2	2	2	2	2	2	2	2	2	2	2
14	2	1	1	1	1	2	1	1	2	1	2	2
15	0	1	2	2	1	2	1	1	2	0	2	2
16	2	1	1	1	1	2	1	2	2	2	2	1
17	1	2	2	2	2	2	2	2	2	2	2	2
18	1	2	1	1	2	2	2	1	0	1	2	1
19	2	2	2	2	2	2	2	2	2	2	2	2
20	2	2	2	2	2	1	2	2	2	2	2	2

Figura B13. Evaluación de TR por Kevin Leandro

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	1	1	0	1	1	1	1	1	1	1	1
2	0	2	1	2	2	2	2	1	1	1	2	1
3	1	2	2	2	1	2	1	2	2	2	2	1
4	1	1	0	1	0	2	1	1	0	1	1	1
5	0	1	0	1	1	2	1	2	1	2	2	1
6	1	2	2	2	2	2	1	1	2	1	2	1
7	0	2	2	2	2	2	1	1	2	1	1	1
8	1	1	2	2	1	2	2	2	2	2	2	1
9	2	1	2	2	1	2	2	2	2	2	2	1
10	2	1	1	1	1	2	2	2	2	2	2	1
11	1	1	2	2	1	2	2	2	2	2	2	1
12	1	2	2	1	1	2	2	1	2	1	2	1
13	1	2	2	2	1	2	1	1	2	1	1	1
14	1	2	2	2	2	2	2	2	2	2	2	1
15	0	2	2	2	1	2	1	2	2	2	2	1
16	1	1	0	1	1	2	2	2	2	2	2	1
17	1	2	2	2	1	2	2	2	2	2	2	1
18	1	2	0	1	1	2	2	0	0	2	2	1
19	1	2	2	2	1	2	2	2	2	2	2	1
20	1	2	1	1	1	1	1	1	2	2	2	1

Figura B14. Evaluación de TR por Esteban Rodríguez

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	2	1	0	1	2	1	2	1	1	2	1
2	0	1	0	1	1	2	1	2	1	1	2	1
3	0	2	2	2	1	2	0	0	2	0	2	0
4	0	2	0	0	1	2	1	2	0	1	2	1
5	0	2	2	1	2	2	0	0	1	0	2	0
6	2	2	0	2	1	2	0	0	2	0	2	0
7	0	2	0	1	1	2	0	0	2	0	2	0
8	0	2	2	2	1	2	2	2	2	1	2	1
9	2	2	1	1	1	2	2	2	2	1	2	1
10	2	2	1	2	1	2	1	2	2	1	2	1
11	0	2	1	2	1	2	1	2	2	2	2	1
12	0	1	0	2	2	1	0	0	2	0	2	1
13	0	2	2	1	1	2	0	0	2	0	2	0
14	2	2	2	1	1	2	0	0	2	0	2	1
15	0	2	2	2	1	2	0	0	2	0	2	1
16	2	1	2	0	1	2	0	0	2	0	2	1
17	0	1	2	2	1	2	0	0	2	0	2	0
18	2	2	2	2	1	2	0	0	0	0	2	0
19	2	1	2	2	1	2	0	0	2	0	2	0
20	2	2	1	1	1	2	0	0	2	0	2	0

Figura B15. Evaluación de TR con AQUSA+

Rúbrica de evaluación

#	Título	Descripción					Criterio de Aceptación					
Número de historia de usuario	Describe una actividad	Bien Formada	Atómica	Mínima	Uniforme	Única (No duplicada)	Bien Formado	Atómico	Mínima	Verificable	Único (no duplicado)	Uniforme
1	2	2	1	0	1	2	1	2	1	1	2	1
2	0	1	0	1	1	2	1	2	1	1	2	1
3	2	2	2	2	1	2	0	0	2	0	2	0
4	0	2	0	0	1	2	1	2	0	1	2	1
5	0	2	2	1	2	2	0	0	1	0	2	0
6	2	2	0	2	1	2	0	0	2	0	2	0
7	0	2	0	1	1	2	0	0	2	0	2	0
8	2	2	2	2	1	2	2	2	2	1	2	1
9	2	2	1	1	1	2	2	2	2	2	2	1
10	2	2	1	2	1	2	1	2	2	1	2	1
11	0	2	1	2	1	2	1	2	2	2	2	1
12	0	1	0	2	2	1	0	0	2	0	2	1
13	0	2	2	1	1	2	0	0	2	0	2	0
14	2	2	2	1	1	2	0	0	2	0	2	1
15	0	2	2	2	1	2	0	0	2	0	2	1
16	2	1	2	0	1	2	0	0	2	0	2	1
17	0	1	2	2	1	2	0	0	2	0	2	0
18	2	2	2	2	1	2	0	0	0	0	2	0
19	2	1	2	2	1	2	0	0	2	0	2	0
20	2	2	1	1	1	2	0	0	2	0	2	0

Figura B16. Guía o grupo control TR

7.3 Anexo C: Historias de usuario del benchmark

Descripción	Criterio de aceptación	Título
As an accountant, I want to update invoice data, so that I can regenerate the invoice for the customer.	*Given an invoice is selected, when viewing it on the grid, then it should display the 'Edit Mode' window.*	*Regenerate Invoice*
As an associate, I want to be able to reprint invoices, so that I can provide it to a customer in need.	*Given an invoice is displayed, when I search the applicaiton, then the invoice report should be regenerated.*	*Printing Invoice*
As a dispatcher, I want to know all shipments, so that I can create the proper documentation.	*Given the reports menu is open in the application when looking at the list of invoices then a window should appear with a list of reports for invoices.*	*Display Invoice Report Options*
As a cashier, I want to have the ability to calculate the import tax depending on the country, so that the invoice reflects the appropriate total	*Given the invoice window is open, when calculating import tax, then it displays the appropriate amount for the specific country*	*Calculate Import Tax*
As a customer, I want to be able to see the discount in my invoice, so that I can decide wheter or not to buy	*Given a proforma is displayed, when clicking on discount option, then a window should appear with the total savings due to discounts*	*Display Discount*
As a dispatcher, I want to be able to hide the transportation details, so that I can negotiate with different providers	*Given a specific invoice is generated, when looking at the transportation details, then I should see a report of providers*	*Hide Transportation Details*
As an customer, I want to manage the items in my shopping cart, so that I can decide what to buy	*Given a shopping cart has items, when viewing the list of items, then I should be able to remove one item*	*Manage Shopping cart*
As a customer, I want to have the ability to finish my order, so that it can be dispatched	*Given an order is open, when I click on proceed to check out, then a confirmation screen should be prompted*	*Dispatching Orders*

As an administrator, I want to have the ability to add new customers, so that they can access our system	*Given the list of customers is displayed, when clicking on the plus sign, then the 'Add customer' window should promt*	*Add New Customer*
As an administrator, I want to have the ability to delete customers, so that they can't access our system	*Given the list of customers is displayed, when clicking on the minus sign, then the remove customer window should be prompted*	*Remove Customer*
As a manager, I want to see the list of customers, so that I can edit it	*Given the main menu is displayed, when clicking on customer list icon, then I should see the lisf of customers*	*Display Customer List*
As an accountant, I want to be able to define an order as 'Non paid', so that I can later track it down	*Given an invoice has been generated, when clicking in 'Non-paid' orders icon, then a list of those orders should be displayed*	*Update Non-Paid Orders*
As a cashier, I want to be able to see the due invoices, so that I can contact the customer	*Given an invoice is over due, when clicking on 'Further actions', then a window to communicate with the customer should appear*	*Display due Invoices*
As a manager, I want to be a able to fix any errors in the invoice, so that I can help the customer	*Given an invoice is displayed, when clicking on edit, then a window should prompt with the option to edit the invoice*	*Edit Invoice*
As a customer, I want to have the choice to select different payment methods, so that I can decide which one is better	*Given a bill is generated, when clicking on 'Payment Method', then a list of payment options should be displayed*	*Select Payment method*
As a business owner, I want to have the ability to change an invoice layout in the system, so that I can change the corporate image	*Given an invoice layout is selected, when clicking on 'Edit', then the different layout options should be displayed*	*Edit Invoice Layout*
As a cashier, I want to have the ability to charge remotely for an invoice, so that the customer doesn't have to come to the store	*Given an invoice is generated, when clicking at the remote payment, then the system should prompt a link to a bak transfer screen.*	*Enable Online bill*

As a salesman, I want the ability to change the freight option depending on the country, so that my customers get charged appropriately	*Given a freight is calculated, when selecting the country, then it should change the value accordingly*	*Modify Freight charge*
As a manager, I want the ability to update the customer info, so that I can have the latest info	*Given the customer info is listed, when clicking on the 'Edit' icon, then I should be able to edit and view the changes*	*Edit customer info*
As a salesman, I want the ability to remove the taxes in an invoice, so that I can give the customer a better price	*Given the invoice is stored, when openning the ivoice, then I should be able to cremove it's taxes from the report*	*Remove taxes from invoice*

Tabla C1. Historias de usuario benchmark