PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

Propuesta de solución para garantizar la trazabilidad de requerimientos funcionales usando desarrollo guiado por comportamiento en una entidad del gobierno

Tesis para obtener el grado académico de Magíster en Informática con mención en Ingeniería de Software que presenta:

Roger Armando Contreras Corrales

Asesor:

Dennis Stephen Cohn Muroy

Lima, 2022

DEDICATORIA

A mis padres, por ser el pilar fundamental en mi vida profesional y personal.

AGRADECIMIENTO

A mis padres, por ser mi ejemplo para seguir adelante en el convivir diario y por inculcarme valores que me han servido en la vida, gracias por eso y por muchos más.

A mi asesor de tesis, el Prof. Dennis Cohn, por su apoyo y recomendaciones durante el desarrollo de la tesis.

A mis profesores de maestría, por los conocimientos impartidos, que han sido fundamentales para poder desarrollar el presente trabajo.

A mis compañeros de trabajo que, a pesar de su poco tiempo disponible, se ofrecieron como participantes para la realización del experimento, sin su apoyo, hubiese sido imposible concluir con este trabajo.

A mis amigos y amigas y a todas las personas que me incentivaron y me motivaron para seguir adelante con los objetivos de este propósito.

Propuesta de solución para garantizar la trazabilidad de requerimientos funcionales usando desarrollo guiado por comportamiento en una entidad del gobierno

Resumen:
Se conoce que el levantamiento de requerimientos es uno de los factores críticos de éxito para los proyectos de software. Los requerimientos tienen una naturaleza cambiante, al punto que los identificados en la fase de definición, pueden diferir de los existentes ya teniendo el producto implementado, por lo que resulta necesario poder rastrearlos durante el ciclo de desarrollo.

Para lograr rastrear los requerimientos, es necesario realizar la trazabilidad de estos a través de diferentes artefactos como diagramas de diseño, clases, casos de prueba, etc. Sin embargo, realizar y mantener los elementos de trazabilidad, son prácticas que se van haciendo más difícil de cumplir conforme la cantidad de requerimientos y artefactos aumenta. Esto debido a que el trabajo de trazabilidad suele realizarse de forma manual.

Para enfrentar estos problemas, se han planteado diferentes estrategias que buscan obtener o mantener de manera automatizada la información de trazabilidad, entre las más difundidas esta *"retrieval information"*; sin embargo, su falta de precisión no la convierte en la solución más idónea.

El enfoque de desarrollo guiado por comportamiento (BDD), se presenta como una alternativa que puede ayudar a enfrentar este problema, en particular si se busca contar con la trazabilidad entre los requerimientos y el código fuente.

En la presente investigación se plantea el objetivo de facilitar la trazabilidad entre requerimientos funcionales y código por medio de una propuesta basada en las prácticas del desarrollo guiado por comportamiento.

Para afirmar si realmente se facilita esta trazabilidad, se ha utilizado el Modelo de Evaluación de Métodos (MEM), donde se mide la eficacia actual y la eficacia percibida. La eficacia actual medida por las variables tiempo promedio y número de omisiones al completar la información de trazabilidad. La eficacia percibida medida por las variables de percepción del MEM.

Se llevo a cabo un experimento con 8 participantes donde se evaluaron cada una de estas variables, los resultados obtenidos permiten afirmar, para la muestra estudiada, que, al utilizar la propuesta, se obtuvo un menor tiempo y número de omisiones al registrar la información de trazabilidad, así mismo, se tiene una mejor facilidad de uso y utilidad percibida, y hay evidencia de una posible intención de uso para la propuesta planteada.

Todos estos resultados nos permiten afirmar, al menos para la muestra estudiada, que al utilizar la propuesta con BDD se facilitaría la trazabilidad entre requerimientos funcionales y código fuente.

Índice general

Índice de tablas:

Índice de Figuras:

1. Descripción de la realidad problemática:

Según el reporte CHAOS 1994 del Standish Group *(Standish Group, 1994)* un 31.1% de los proyectos fracasaron y un 52.7% fueron completados superando el presupuesto. Veinte años después, en el reporte CHAOS 2015 *(Standish Group, 2015),* estas cifras mejoraron, donde el 19% de los proyectos fracasaron y un 45% fueron completados superando el presupuesto. Ambos reportes coinciden en que los requerimientos son un factor clave en el éxito de los proyectos.

Los requerimientos tienen una naturaleza cambiante, al punto que los identificados en la fase de definición pueden diferir a los que se tiene llegada la fase de pruebas del producto. Con el fin de mostrar que los requerimientos son cumplidos diligentemente, resulta necesario poder rastrearlos durante el ciclo de vida del desarrollo. *(Chemuturi, 2012, pp. 129–134)*

A través de la trazabilidad de requerimientos cada miembro del equipo debe tener acceso a información que le ayuda a determinar de dónde viene un requerimiento, su importancia, como fue implementado y probado *(Kannenberg & Saiedian, 2009).* Así mismo, la trazabilidad de requerimientos es un elemento crítico en cualquier proceso de desarrollo de software ya que impacta directamente en su calidad. *(Rempel & Mader, 2017)*

Sin embargo, uno de los problemas que impide una amplia adopción de la trazabilidad es la gran cantidad de trabajo manual que se tiene que llevar a cabo para mantener esta información actualizada *(Aizenbud-Reshef et al., 2006)*. Según *(Cleland-Huang, Zemont, & Lukasik, 2004, pp. 1–3)* la creación manual de un vínculo de trazabilidad entre dos artefactos toma un promedio de 15 minutos, y dependiendo del tipo de artefactos el tiempo promedio puede variar entre 10 y 45 minutos *(Heindl & Biffl, 2005)*. En la industria del software llevar a cabo esta tarea de forma manual resulta poco viable, ya que los elementos de la trazabilidad crecen de manera exponencial a medida que aumenta la complejidad y tamaño del software *(Cleland-Huang et al., 2003)*.

Otro problema que dificulta la trazabilidad de requerimientos se presenta cuando se necesita recurrir a diferentes fuentes de información (documentos, repositorios, herramientas de software, etc.) para obtener información de trazabilidad. Al integrar esta información hay riesgo de incurrir en omisiones o errores, dando como resultado información poco fiable *(Gotel & Finkelstein, 2002)*.

La “Figura 1.1” muestra el árbol de problemas que resume la problemática que se abordará en la presente investigación.

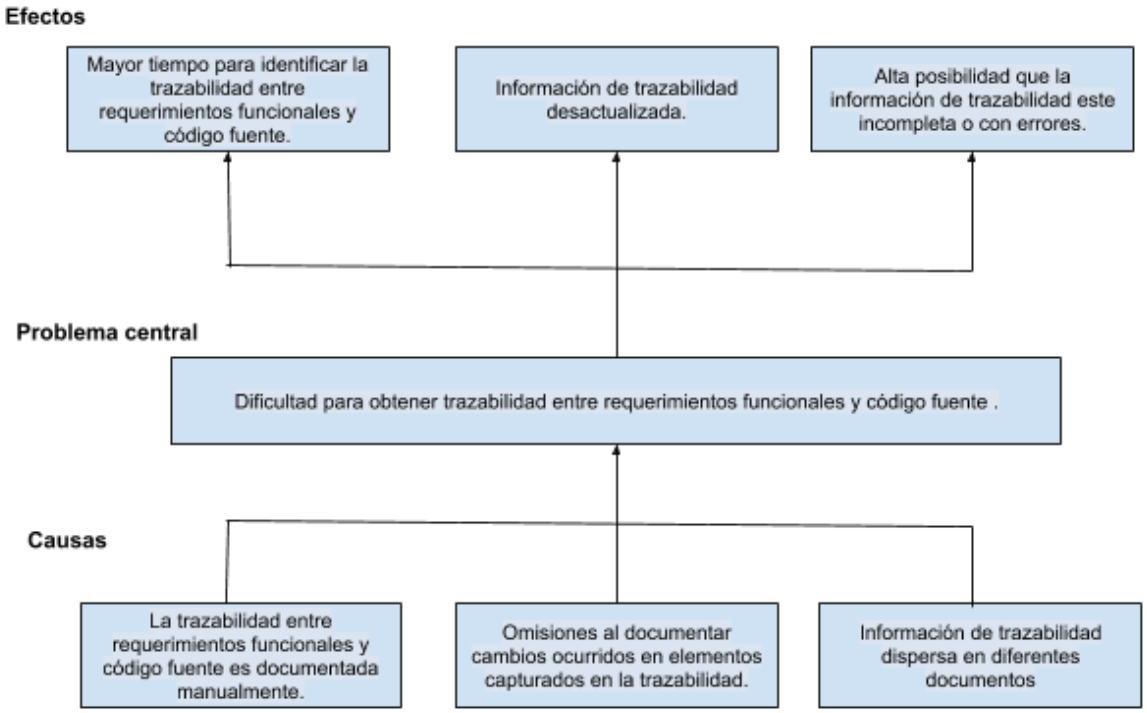


Figura 1.1: Árbol de problemas

La presente investigación se aplicará en una entidad del gobierno, que tiene a su cargo el desarrollo y mantenimiento de un alto número de aplicaciones informáticas (mayor a 100). Estas aplicaciones sirven a un gran volumen de usuarios que, dependiendo de la aplicación, puede variar entre miles y millones de usuarios.

Constantemente se están implementando requerimientos, algunos son nuevos, otros modifican los existentes. Por lo que resultado importante tener información de trazabilidad de estos requerimientos para facilitar su mantenimiento y evaluación del impacto de futuros cambios.

Esta entidad del gobierno actualmente presenta problemas con respecto al registro y mantenimiento de información de trazabilidad; siendo el registro manual una de las causas principales. Por consiguiente, la información de trazabilidad resulta incompleta y poco fiable, lo que dificulta la evaluación del impacto por cambios, así como demoras en la implementación de los requerimientos.

2. Marco conceptual:

A continuación, se detallan los conceptos que se encuentran asociados con la presente investigación:

2.1. Requerimientos funcionales

El SWEBOK *(Bourque et al., 2014, pp. 1–3)* clasifica los requerimientos en dos grandes grupos: Requerimientos funcionales y requerimientos no funcionales.

Los requerimientos funcionales son declaraciones de servicios que el sistema debe ofrecer, como el sistema debería reaccionar a las entradas y cómo debería comportarse en determinadas situaciones. *(Sommerville, 2016, pp. 105–106)*

2.2. Especificación de requerimientos

La especificación de requerimientos según la guía SWEBOK *(Bourque et al., 2014, pp. 1–11)* se refiere a la producción de un documento que puede ser sistemáticamente revisado, evaluado y aprobado. En la especificación para productos software el documento suele ser llamado "Especificación de requerimientos de software" el cual define lo que se espera que el software haga y no haga. Este documento, suele ser casi siempre escrito en lenguaje natural y complementado con diagramas apropiados, notaciones y tablas *(Sommerville, 2016, pp. 120–121).*

2.3. Casos de uso

Los casos de uso son una manera de describir las interacciones entre los usuarios y el sistema usando un **modelo** gráfico y un texto estructurado. En su forma más simple un caso de uso identifica los actores involucrados en la interacción, y luego se agrega información adicional para describir la interacción con el sistema. Esta información adicional puede ser descrita textualmente o con una o más notaciones gráficas tales como los diagramas UML de secuencia o estados. El conjunto de casos de uso representa todas las posibles interacciones descritas en los requerimientos del sistema. *(Sommerville, 2016, pp. 125–126)*

En *(Koelsch, 2016, pp. 277–278)* complementa que, dependiendo de la complejidad o las necesidades, los casos de uso son descritos por una combinación de los siguientes elementos:

- Título: Es el identificador o nombre del caso de uso.
- Descripción: Una breve descripción del propósito del caso de uso.
- Actor: Alguien o algo (otro sistema) que inicia la acción.
- Precondiciones: Lo que debe suceder antes que el caso de uso se ejecute.
- Postcondiciones: Lo que debe suceder después que el caso de uso se ha ejecutado.

- Desencadenadores: Los eventos que causan que el caso de uso sea iniciado.
- Flujo básico: Cuando en la secuencia de pasos del caso de uso nada sale mal.
- Flujo alternativo: Son variaciones al flujo principal, manejan las situaciones en las que algo sale mal en el flujo básico o una condición alterna causa un cambio en el flujo básico.

2.4. Trazabilidad de requerimientos

La trazabilidad es un término de ingeniería de software para referirse a la actividad de documentar los vínculos entre los productos de trabajo de ingeniería de software (requerimientos, casos de prueba, casos de uso, etc.). Para su implementación se pueden usar herramientas como la matriz de trazabilidad que permiten representar las relaciones entre los requerimientos y otros productos de trabajo *(Pressman & Bruce R. Maxim, 2014, pp. 142–143).*

La trazabilidad de requerimientos permite la identificación de los requerimientos en cada uno de los artefactos que se van generando durante el proceso de desarrollo del software. Asimismo, hace posible, en cualquier punto del proceso de desarrollo del producto, se pueda identificar de manera bidireccional la relación entre un artefacto generado y el requerimiento que lo originó. *(Chemuturi, 2012, pp. 129–130)*

A medida que el software es desarrollado o recibe mantenimiento ya sea correctivo o evolutivo, va creciendo la necesidad de mapear y controlar la información de las relaciones de trazabilidad para demostrar la consistencia de los requerimientos del software con el modelo del software y los diferentes productos de trabajo. Entre los diversos beneficios de su uso destaca el facilitar la gestión de los productos de trabajo del software, mejorar la calidad del producto y también facilitar el análisis de impacto por la implementación de cambios en el software. *(Bourque et al., 2014)*

El mecanismo más popular para lograr la trazabilidad es la matriz de trazabilidad, que viene a ser una tabla donde cada fila da información de la trazabilidad de un requerimiento, las primeras columnas de esta tabla serán para identificar el código y descripción del requerimiento, y las siguientes columnas tienen la referencia a los artefactos o productos de trabajo asociados. Esta matriz de trazabilidad puede ser almacenadas en hojas de cálculo; sin embargo, a medida que la complejidad y tamaño del software aumentan, resulta más complicado mantener y actualizar esta hoja de cálculo, por lo que se recomienda usar herramientas especializadas o sistemas que almacenen la información de trazabilidad en una base de datos *(Koelsch, 2016, pp. 130–133)*.

2.5. Desarrollo guiado por comportamiento

El término desarrollo guiado por comportamiento (BDD) fue originalmente inventado por Dan North como una manera más fácil de enseñar y aplicar el desarrollo guiado por pruebas (TDD) *(Smart, 2014, pp. 12)*. Dan North menciona que entre los

programadores siempre surgían complicaciones al momento de aplicar TDD, que los llevaba a las siguientes cuestiones: ¿Cuándo empezar a probar?, ¿Qué se debe y no se debe probar?, ¿A qué llamar prueba y como saber por qué esta falla? Con el fin de enfrentar estas complicaciones, Dan North propone el uso BDD. *(North, 2006)*

En *(Smart, 2014, pp. 12)* se define BDD como un conjunto de prácticas de ingeniería de software diseñadas para ayudar a los equipos a construir y entregar software más rápido, de mayor calidad y valor. BDD está basada en prácticas ágiles como desarrollo dirigido por pruebas (TDD), y diseño dirigido por dominios (DDD). Entre las cosas más importantes, es que BDD provee un lenguaje común que facilita la comunicación entre los miembros del equipo de desarrollo y los stakeholders *(Smart, 2014, pp. 12)*.

BDD se enfoca en implementar la mínima característica comerciable (MMF, por sus siglas en inglés) que vienen a ser funcionalidades alineadas con en el negocio. Con este enfoque, el equipo del negocio y el de desarrollo pueden cooperar usando un lenguaje común que ayuda a reducir el desperdicio, código y funcionalidades innecesarias. *(Ye, 2013, pp. 6)*

Según lo mencionado en *(Smart, 2014, pp. 15–28)* en BDD se parte de los objetivos de negocio, luego se definen las características o ***features*** que vienen a ser los requerimientos funcionales que el sistema va a ofrecer y que deben estar alineados con los objetivos de negocio.

Las características o *features* son detallados por medio de **ejemplos** que son historias y escenarios que describen lo que el usuario espera que el sistema haga. Los ejemplos son descritos en lenguaje Gherkin, que tiene una sintaxis aproximada al lenguaje natural, puede ser entendido por los interesados y miembros del equipo de desarrollo del software. Una plantilla para definir una característica o *feature* utilizando lenguaje Gherkin, sería:

```
Feature <Nombre de la característica o feature>
In order to <lograr un objetivo de negocio>
As a <stakeholder>
I want <lo que el sistema espera que haga en cumplimiento del feature>
Scenario: <Nombre del escenario>
        Given <Contexto o precondiciones del escenario>
        When <evento o una acción del usuario>
        Then <Resultados esperados>
```

Los ejemplos forman la base para las especificaciones que el equipo de desarrollo utilizará para construir el software. Asimismo, serán traducidos a **especificaciones ejecutables -** pruebas automatizadas para verificar la funcionalidad entregada por el software. Estas especificaciones son ejecutadas cada vez que se da un cambio en la aplicación; de esta manera, siempre se verifica la integridad de las funcionalidades.

3. Objetivos:

Como objetivo general de la presente investigación, se plantea definir una propuesta, basada en prácticas de desarrollo guiado por comportamiento (BDD), que permita facilitar la trazabilidad entre requerimientos funcionales y código fuente.

3.1. Objetivos específicos:

Para el cumplimiento de este objetivo general, es necesario que la propuesta permita cumplir con los siguientes objetivos específicos:

- Automatizar la trazabilidad entre requerimientos funcionales y el código fuente.
- Facilitar el registro de los elementos identificados en la trazabilidad.
- Centralizar la información de trazabilidad.

El planteamiento de estos objetivos puede ser apreciado en la “Figura 3.1”, el cual ha sido derivado del árbol de problemas (ver “Figura 1.1”).

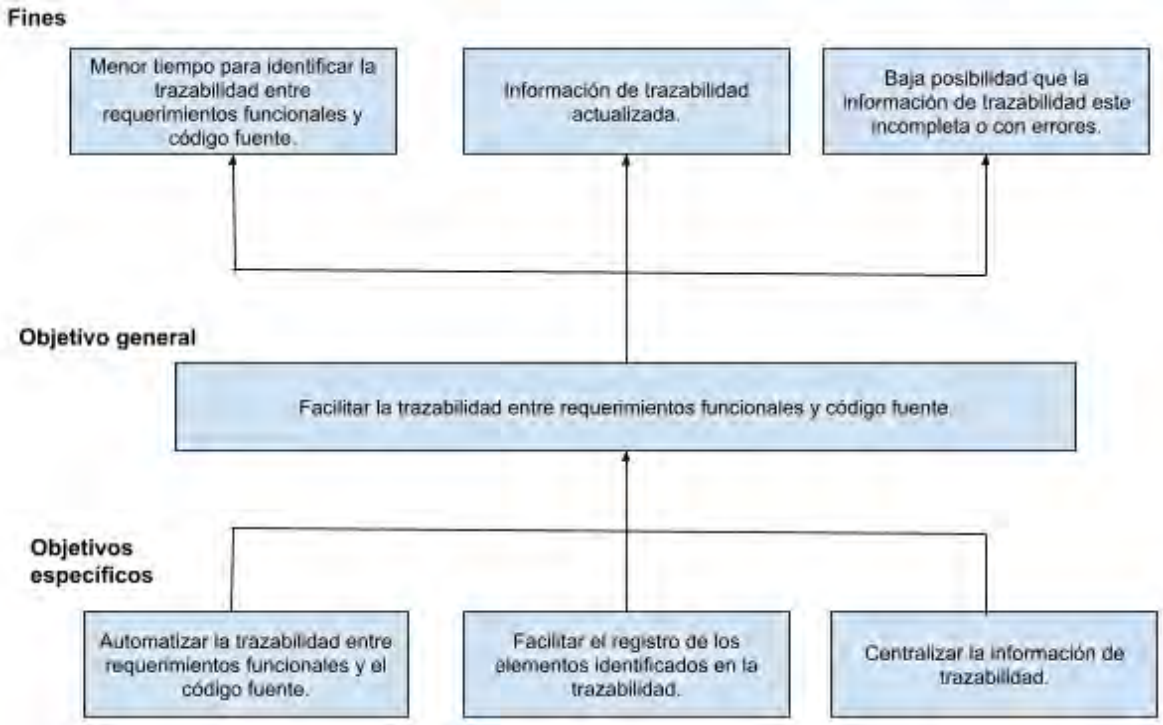


Figura 3.1: Árbol de objetivos

En la “Tabla 3.1” se puede apreciar la matriz de marco lógico, que ha sido definida con el fin de verificar el cumplimiento de los objetivos planteados.

OBJETIVO GENERAL	RESULTADOS FINALES	MEDIOS DE VERIFICACIÓN
Facilitar la trazabilidad entre requerimientos funcionales y código fuente	Obtención de la trazabilidad por medio de BDD.	Evaluación de la eficacia percibida al utilizar BDD para obtener información de trazabilidad.
OBJETIVOS ESPECÍFICOS	**RESULTADOS INTERMEDIOS**	**MEDIOS DE VERIFICACIÓN**
Automatizar la trazabilidad entre requerimientos funcionales y el código fuente	Obtención automatizada de la información de trazabilidad entre requerimientos funcionales y código fuente.	Reporte automático de información de trazabilidad.
Facilitar el registro de los elementos identificados en la trazabilidad.	Reducción de tiempo necesario para registrar la información de trazabilidad.	Medición del tiempo necesario para registrar la información de trazabilidad.
	Menor cantidad de omisiones al momento de registrar la información de trazabilidad.	Medición del número de omisiones al momento de registrar la información de trazabilidad.
	Mejora en la eficacia percibida al registrar la información de trazabilidad.	Evaluación de variables de percepción del modelo de evaluación de métodos.
Centralizar la información de trazabilidad.	Información de trazabilidad se obtiene de una sola fuente de información.	Repositorio o fuente de información de trazabilidad.

Tabla 3.1: Matriz de marco lógico

4. Justificación:

La trazabilidad de requerimientos, llevada de una manera apropiada, ofrece muchos beneficios a la organización. Por ello es un componente importante en muchos estándares y modelos de mejoras de procesos de desarrollo de software, tales como CMMI e ISO 9001-2000 (*Kannenberg & Saiedian, 2009)*

Adicionalmente, trae beneficios como facilitar la evaluación de impacto por cambios en el software al permitir identificar los artefactos asociados a un requerimiento *(Kannenberg & Saiedian, 2009)*. Así mismo, la calidad del software se ve mejorada. Un estudio llevado a cabo por *(Rempel & Mader, 2017)* demuestra que un mayor nivel de completitud de la información de trazabilidad de requerimientos ayuda a disminuir la ratio de defectos esperados.

Dadas las dificultades que involucra tener una trazabilidad de requerimientos, es que surge la necesidad de plantear una propuesta que facilite la trazabilidad entre requerimientos y código fuente.

Es aquí donde el enfoque de desarrollo guiado por comportamiento (BDD) puede ayudar a lidiar con estos problemas, ya que vincula las pruebas automatizadas con una descripción textual (lenguaje Gherkin) de los requerimientos. Esta vinculación va a permitir obtener una trazabilidad entre los requerimientos y las diversas unidades de construcción (código fuente).

La propuesta a desarrollar va a ser probada en una institución del gobierno, cuya área de desarrollo de software presenta las mismas dificultades anteriormente expuestas en la presente investigación.

5. Estado del arte

Para la realización del estado del arte, se desarrollará una revisión sistemática de la literatura (SLR, por sus siglas en inglés). La SLR es un medio para identificar, evaluar e interpretar toda la investigación disponible para responder a unas preguntas de investigación específicas. *(Kitchenham & Charters, 2007)*

Una SLR, a diferencia de una revisión tradicional de la literatura (TLR, por sus siglas en inglés), tiene una planificación formal y se ejecuta de manera metódica y sistemática. Esto hace que una SLR sea repetible de forma independiente, dándole un mayor valor científico que una TLR *(Piattini et al., 2014, p. 199)*.

Para realizar la SLR se seguirá el proceso propuesto por (*Kitchenham & Charters, 2007) y representado en la Figura 5.1*.

Figura 5.1: Proceso para realizar la SLR

5.1. Propósito y preguntas de investigación:

El propósito de esta SLR es tener conocimiento sobre el estado del arte con respecto a la aplicación de trazabilidad de requerimientos de software utilizando BDD.

Para este fin, en la tabla 5.1, se define la siguiente pregunta de investigación:

Pregunta de investigación	Motivación
PI1: ¿Qué propuestas existen para la obtención automática de información de trazabilidad entre requerimientos funcionales y código fuente?	Obtener un estado del arte referente a propuestas para la obtención automática de información de trazabilidad entre requerimientos y código fuente.
PI2: De las propuestas encontradas, ¿Alguna plantea utilizar BDD?	Conocer si alguna de las propuestas planteadas utiliza BDD para obtener la información de trazabilidad.

Tabla 5.1: Preguntas de investigación

5.2. Definición de la búsqueda:

Para la redacción de las preguntas de investigación se ha tenido en cuenta los criterios PICOC, que es un acrónimo en inglés para los términos: P - población, I - intervención, C - comparación, O - resultados y C - contexto (Piattini et al., 2014, p. 203)

5.2.1. Cadena de búsqueda

A partir de los criterios PICOC se obtendrán los términos clave, que serán de utilidad para definir las cadenas de búsqueda.

En la “Tabla 5.2” y “Tabla 5.3” se define el PICOC para la pregunta de investigación PI1 y PI2 respectivamente.

Criterio	Alcance
Población	Trazabilidad entre requerimientos y código fuente
Intervención	Obtención automatizada de información de trazabilidad entre requerimientos y código fuente

Tabla 5.2: PICOC para la pregunta PI1

Criterio	Alcance
Población	Trazabilidad entre requerimientos y código fuente
Intervención	Obtención automatizada de información de trazabilidad entre requerimientos y código fuente
Resultados	BDD.
Contexto	Entorno académico o aplicación en la industria

Tabla 5.3: PICOC para la pregunta PI2

En base a las tablas 5.2 y 5.3 se obtienen los términos de búsqueda mostrados en la Tabla 5.4.

Criterio	Conceptos	Términos en inglés
Población	Trazabilidad entre requerimientos y código fuente	("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") AND ("source code” OR “source” OR “code")
Intervención	Automatizado	"automated" OR "automatic" OR "automatized" OR "automatically" OR "automate"
Resultado	Desarrollo guiado por comportamiento	"behavior driven development" OR "behavior-driven development" OR “behavior driven” OR “BDD”

Tabla 5.4: Términos clave

Para la pregunta de investigación PI1, se plantea la siguiente cadena de búsqueda:

- **Cadena 1:** *("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") AND ("source code" OR "source" OR "code") AND ("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate")*

Para la pregunta de investigación PI2, se plantea la siguiente cadena de búsqueda:

- **Cadena 2:** *Cadena 1 AND ("behavior driven development" OR "behavior-driven development" OR "behavior driven" OR "BDD")*

El tipo de búsqueda que se va a realizar es semi automática.

5.2.2. Criterios de selección de estudios:

Los estudios encontrados al utilizar las cadenas de búsqueda tendrán que pasar por los siguientes dos criterios de selección:

Criterios de inclusión:

- **CI.1:** El tema de estudio debe ser referente a obtención automatizada de información de trazabilidad entre requerimientos funcionales de software y código fuente.
- **CI.2:** Se tiene acceso completo al estudio. Es decir se puede tener acceso al contenido de la publicación.
- **CI.3:** Los estudios se encuentran escritos en inglés.
- **CI.4:** Para el caso de estudios duplicados, se selecciona el más detallado.

Criterios de exclusión:

- **CE.1:** No propone una técnica para la obtención automatizada de información de trazabilidad entre requerimientos funcionales de software y código fuente.
- **CE.2:** La técnica propuesta no fue aplicada durante el desarrollo del estudio.

Temporalidad:

No se ha incluido fecha de delimitación, ya que en la presente SLR se pretende abarcar la mayor cantidad de estudios que sea posible.

Fuente de datos:
En la “Tabla 5.5” se indican las bases de datos utilizadas para realizar la SLR.

Id	Base de datos	URL
SC	Scopus	https://www.scopus.com
IEEE	IEEE Xplore	https://ieeexplore.ieee.org
ACM	ACM Digital Library	https://dl.acm.org/
WS	Web of Science	http://isiknowledge.com

Tabla 5.5: Bases de datos consultadas

5.2.3. Procedimiento para selección de estudios:

En esta SLR se ha considerado el siguiente proceso, aplicando los criterios según la “Tabla 5.6”:

- Primera etapa: Se seleccionan, de las fuentes de datos, aquellos artículos en inglés donde el título o resumen estén relacionados al tema: “Obtención automatizada de información de trazabilidad entre requerimientos funcionales y código fuente”. Si tras leer el resumen quedase algunas dudas, se procede a leer el artículo completo.

- Segunda etapa: Se excluyen aquellos artículos donde no se haya propuesto y aplicado alguna técnica para la obtención de información de trazabilidad entre requerimientos funcionales y código fuente.

- Tercera etapa: Si no se tiene acceso completo al estudio, éste no será tomado en cuenta.

- Cuarta etapa: Si hay artículos duplicados, se selecciona aquel que se encuentre más completo.

Procedimiento	Criterios de selección
Primera etapa	CI.1, CI.3
Segunda etapa	CE.1, CE. 2
Tercera etapa	CI. 2
Cuarta etapa	CI. 4

Tabla 5.6: Procedimiento para selección

5.3. Resultados y análisis:

La “Cadena 1” y “Cadena 2” de búsqueda fueron ejecutadas el mes de setiembre del 2021 en cada una de las bases de datos seleccionadas (Ver “Tabla 5.5”).

Para la “Cadena 1” de búsqueda se obtuvo un total de 139 artículos, luego de pasar por los criterios de selección, la cifra se redujo a 46 artículos, de estos 17 eran repetidos, quedando en total 29 artículos. Para la “Cadena 2” de búsqueda sólo se obtuvieron 2 artículos.

5.3.1. Obtener resultados de la búsqueda:

Con las cadenas de búsquedas ya definidas, se procedió a armar la consulta en cada base de datos según los términos búsqueda (ver “Tabla 5.4”), la consulta aplicaba la búsqueda en el título, resumen y palabras clave.

La “Tabla 5.7” y “Tabla 5.8”, indican cuál fue la consulta ejecutada por cada base de datos, así como la cantidad de resultados que se obtuvo.

Base de datos	Consulta	Resultados encontrados
Scopus	(TITLE-ABS ("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") AND TITLE-ABS ("source code" OR "source" OR "code") AND TITLE-ABS ("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate"))	80
IEEE Xplore	((((("Abstract":"requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") OR ("Document Title":"requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements")) AND (("Abstract":"source code" OR "source" OR "code") OR ("Document Title":"source code" OR "source" OR "code"))) AND (("Abstract":"automated" OR "automatic" OR "automatized" OR "automatically" OR "automate") OR ("Document Title":"automated" OR "automatic" OR "automatized" OR "automatically" OR "automate"))))	31
ACM Digital Library	((Abstract:("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") OR Title:("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements")) AND (Abstract:("source code" OR "source" OR "code") OR Title:("source code" OR "source" OR "code"))) AND (Abstract:("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate") OR Title:("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate"))	10
Web Of Siencie	(TS=(("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") AND ("source code" OR "source" OR "code") AND ("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate")))	18

Tabla 5.7: Consultas para la “Cadena 1” de búsqueda

Base de datos	Consulta	Resultados encontrados
Scopus	(TITLE-ABS ("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") AND TITLE-ABS ("source code" OR "source" OR "code") AND TITLE-ABS ("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate") AND TITLE-ABS ("behavior driven development" OR "behavior-driven development" OR "behavior driven" OR "BDD"))	2
IEEE Xplore	(((("Abstract":"requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") OR ("Document Title":"requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements")) AND (("Abstract":"automated" OR "automatic" OR "automatized" OR "automatically" OR "automate") OR ("Document Title":"automated" OR "automatic" OR "automatized" OR "automatically" OR "automate")) AND (("Abstract":"behavior driven development" OR "behavior-driven development") OR ("Document Title":"behavior driven development" OR "behavior-driven development")) AND (("Abstract":"behavior driven development" OR "behavior-driven development" OR "behavior driven" OR "BDD") OR ("Document Title":"behavior driven development" OR "behavior-driven development" OR "behavior driven" OR "BDD"))))	1
ACM Digital Library	((Abstract:("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") OR Title:("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements")) AND (Abstract:("source code" OR "source" OR "code") OR Title:("source code" OR "source" OR "code"))) AND (Abstract:("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate") OR Title:("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate")) AND (Abstract:("behavior driven development" OR "behavior-driven development" OR "behavior driven" OR "BDD") OR Title:("behavior driven development" OR "behavior-driven development" OR "behavior driven" OR "BDD"))	0
Web Of Siencie	(TS=(("requirements traceability" OR "requirements tracing" OR "requirements trace" OR "trace requirements" OR "tracing requirements") AND ("automated" OR "automatic" OR "automatized" OR "automatically" OR "automate") AND ("behavior driven development" OR "behavior-driven development")))	0

Tabla 5.8: Consultas para la “Cadena 2” de búsqueda

5.3.2. Seleccionar los estudios primarios:

Luego de obtener los primeros resultados, se procedió a aplicar los criterios de selección según lo indicado en la sección “Procedimiento para selección de estudios” de este documento.

Al aplicar los criterios de selección, se revisó el título y resumen de cada uno de los artículos obtenidos. En caso hubiese duda acerca de su inclusión, se procedió a leer la introducción y/o conclusión del artículo.

Al finalizar este proceso se obtuvo 46 artículos, luego de eliminar los duplicados, se redujo a 29 artículos que son mostrados en la “Tabla 5.9”.

Artículos	Base de datos	Cadena de búsqueda
S4, S10, S11, S12, S16, S27, S29	SC	1
S17	SC	1, 2
S20	IEEE	1
S21	ACM	1
S22, S23	WS	1
S5	SC, IEEE	1, 2
S1, S3, S6, S7, S13, S14, S15, S18, S24, S25, S26, S28	SC, IEEE	1
S2, S8, S9, S19	SC, WS	1

Tabla 5.9: Selección de estudios primarios

Los artículos en la Tabla 5.9 se encuentran codificados, consultar el Anexo 1, para conocer cuál es su referencia.

5.3.3. Evaluar la calidad de los estudios:

Los estudios previamente seleccionados fueron sometidos a un proceso de evaluación de calidad. Para este fin, se ha definido un cuestionario de evaluación de la calidad de los estudios, mostrado en la Tabla 5.3, que está basado en la propuesta de *(Zarour, Abran, Desharnais, & Alarifi, 2015)*.

ID	Pregunta
QA1	¿Se ha definido de forma clara el objetivo del estudio?
QA2	¿El método para obtener la trazabilidad entre requerimientos y código fuente, ha sido claramente definido?
QA3	¿Se discute alguna amenaza a la validez?
QA4	¿Se ha descrito el contexto en el cual se ha llevado el estudio?
QA5	¿Hay algún resultado presentado como salida de la investigación?

Tabla 5.10: Cuestionario para evaluación de calidad

Este cuestionario utiliza una escala de tres niveles (Si = 1 punto, No = 0 Punto, Parcialmente = 0.5 punto). Todos aquellos estudios que obtengan una calificación igual o menor a 5.0 y mayor o igual a 2.5 serán consideradas como aceptados. El resultado de la evaluación de calidad se resume en la “Tabla 5.11”:

ID	QA1	QA2	QA3	QA4	QA5	Calificación
S1	1	1	0	1	1	4
S2	1	1	1	1	1	5
S3	1	1	1	1	1	5
S4	1	1	1	1	1	5
S5	1	1	0	1	1	4
S6	1	1	1	1	1	5
S7	1	1	0	1	1	4
S8	1	1	1	1	1	5
S9	1	1	1	1	1	5
S10	1	1	1	1	1	5
S11	1	1	0	1	1	4
S12	1	1	0	0.5	1	3.5
S13	1	1	1	1	1	5
S14	1	1	1	1	1	5
S15	1	1	1	1	1	5
S16	1	1	1	1	1	5
S17	1	1	0	1	0	3
S18	1	1	1	1	1	5
S19	1	1	0	1	1	4
S20	1	0.5	0	0.5	0	2
S21	1	1	0	0	0	2
S22	1	1	1	1	1	5
S23	1	1	1	1	1	5
S24	1	0.5	0	1	1	3.5
S25	1	1	1	1	1	5
S26	1	1	1	1	1	5
S27	1	1	1	1	1	5
S28	1	1	1	1	1	5
S29	1	1	1	1	1	5

Tabla 5.11: Resultado de evaluación de calidad

Según los resultados de la “Tabla 5.11”, hay dos artículos que no pasaron la evaluación de la calidad, por tanto, se tendrá en cuenta los otros 27 artículos.

5.3.4. Extracción de los datos y síntesis:

Luego de revisar los artículos, se procederá a contestar las preguntas planteados en esta revisión sistemática.

- ***PI1: ¿Qué estudios existen referentes a obtención automática de trazabilidad entre requerimientos funcionales y código fuente?***

  Según lo recopilado, se observa que existen diferentes estudios orientados a obtener de manera automatizada trazabilidad de requerimientos. Estos estudios se resumen en la Tabla 5.12.

Propuesta	Estudios	Cantidad
Algoritmos para recuperación de información (*“Information Retrieval”)*	S1, S2, S3, S4, S6, S8, S9, S12, S13, S14, S16, S18, S25, S27, S28, S29	16
Desarrollo guiado por comportamiento (BDD)	S5, S17	2
R2C (Herramienta que usa varias técnicas para obtener la trazabilidad)	S7	1
Trazabilidad basada en la información obtenida de los logs registrados.	S10	1
Extracción de un prototipo de modelo de dominio a partir de la descripción textual de los requerimientos.	S11	1
Enfoque sistemático sobre cómo usar los productos de trabajo para capturar los vínculos entre requerimientos y código.	S15	1
Herramienta a la que se debe proveer cierta información de trazabilidad, y a partir de esta infiere la información faltante.	S19	1
Lenguaje para capturar información de trazabilidad.	S22	1
Técnicas bayesianas para valorar la relevancia de información de trazabilidad obtenida automáticamente.	S23	1
Enfoque basado en heurística, utilizando análisis de código estático y análisis de texto.	S24	1
Se enfoca como un problema de optimización, en el cual se utiliza el algoritmo NSGA-II	S26	1

Tabla 5.12: Propuestas planteadas en los estudios

De esta tabla se aprecia que la mayoría de las técnicas utilizadas para obtener la información de trazabilidad, son realizadas por medio de algoritmos para recuperación de información. Estos algoritmos lo que buscan es la similitud textual entre los requerimientos funcionales y el código fuente, mientras haya una mayor similitud, entonces es muy probable que se haya encontrado un elemento de trazabilidad. De todos los algoritmos para recuperación de información, se encontró que el que daba mejores resultados es el que aplicaba el modelo probabilístico *(Rodriguez, 2009)*.

Se plantean también otras propuestas, que buscan mejorar los resultados al aplicar algoritmos de recuperación de información. Una de estas propuestas, es utilizar los logs generados al subir cambios en repositorios de código fuente como SVN o GIT, pues en ellos hay información descriptiva referente al cambio realizado y los elementos de código fuente involucrados *(Tsuchiya et al., 2006)*. Otra propuesta busca calificar la relevancia de la información de trazabilidad según la tarea que se este realizado, como por ejemplo el mantenimiento de un requerimiento funcional, en donde sólo se desea obtener los elementos de código fuente vinculados *(Omoronyia, 2011).*

Se puede apreciar a partir de los estudios obtenidos, que la obtención de información de trazabilidad, se suele automatizar por medio de los algoritmos de recuperación de información, así como también la aplicación de mejoras sobre ellos para aumentar la precisión de esta información de trazabilidad.

- ***PI2: De los estudios obtenidos, ¿Alguno de ellos utiliza Desarrollo guiado por comportamiento (BDD)?***
  Hay dos estudios que utilizan el enfoque de desarrollo guiado por comportamiento (BDD) para obtener la información de trazabilidad.

  En *(De Carvalho et al., 2013),* se propone vincular los WorkFlows a pruebas automatizadas usando BDD. Esta vinculación permitirá obtener automáticamente la trazabilidad de requerimientos.

  En *(Lucassen et al., 2017)*, se propone el “Método de trazabilidad guiado por comportamiento” (BDT por sus siglas en inglés). Este método establece una trazabilidad entre los requerimientos y el código fuente, tomando ventaja de las pruebas automatizadas que son generadas como parte del proceso de BDD. Así mismo se menciona el uso de herramientas de trazabilidad de ejecución del código, con el fin de poder obtener los elementos de código fuente invocados al momento de ejecutar los casos de prueba, y con esto obtener la trazabilidad.

  Ambos estudios proponen utilizar las pruebas en BDD como un mecanismo para obtener los elementos de trazabilidad, sin embargo, en *(De Carvalho et al., 2013)* se buscaba obtener la trazabilidad entre los workflow y los requerimientos funcionales, y no se observa cómo se podría utilizar BDD para obtener la trazabilidad entre requerimientos funcionales y código fuente.

5.3.5. Conclusiones:

El objetivo de esta revisión fue conocer el estado del arte, referente a los métodos y técnicas utilizados para obtener automáticamente la información de trazabilidad entre requerimientos y código fuente. A partir de los estudios encontrados, se revisó si existen algunos que usen desarrollo guiado por comportamiento (BDD).

De los estudios encontrados, se pudo observar que más de la mitad (S1, S2, S3, S4, S6, S8, S9, S12, S13, S14, S16, S18, S25, S27, S28, S29) se enfocan en técnicas de "*Retrieval Information*". Estos estudios básicamente proponen algoritmos o mejoras a los existentes, con la finalidad de incrementar la precisión de la información de trazabilidad. Pues muchas de éstas, dan falsos positivos, es decir vínculos de trazabilidad incorrectos.

En lo que respecta a BDD, se encontraron pocos estudios, sólo dos (S5 y S17), que coinciden en el uso de las pruebas automatizadas generadas, como medio para obtener la información de trazabilidad entre requerimientos y código fuente.

6. Propuesta para garantizar la trazabilidad

Antes de describir la propuesta para garantizar la trazabilidad, se explicará como actualmente se trabaja en la organización gubernamental donde se llevará a cabo el experimento, a esta forma de trabajo la llamaremos “método tradicional”.

6.1. Método tradicional para obtener la trazabilidad:

La organización utiliza un documento llamado “Especificación de requerimientos”, que contiene la documentación del análisis, diseño y especificaciones de construcción de los requerimientos que se implementarán.

A nivel de análisis, se define los requerimientos funcionales y casos de uso, que son detallados por medio de la especificación de casos de uso. También se define una matriz de trazabilidad que vincula los requerimientos funcionales con los casos de uso.

En el diseño, se incluye por lo menos los siguientes diagramas UML: Diagrama de casos de uso, diagrama de paquetes, diagrama de componentes, diagrama de clases de diseño, diagrama de despliegue. Sin embargo, no existe una trazabilidad entre los requerimientos funcionales - o casos de uso - con los artefactos del diseño.

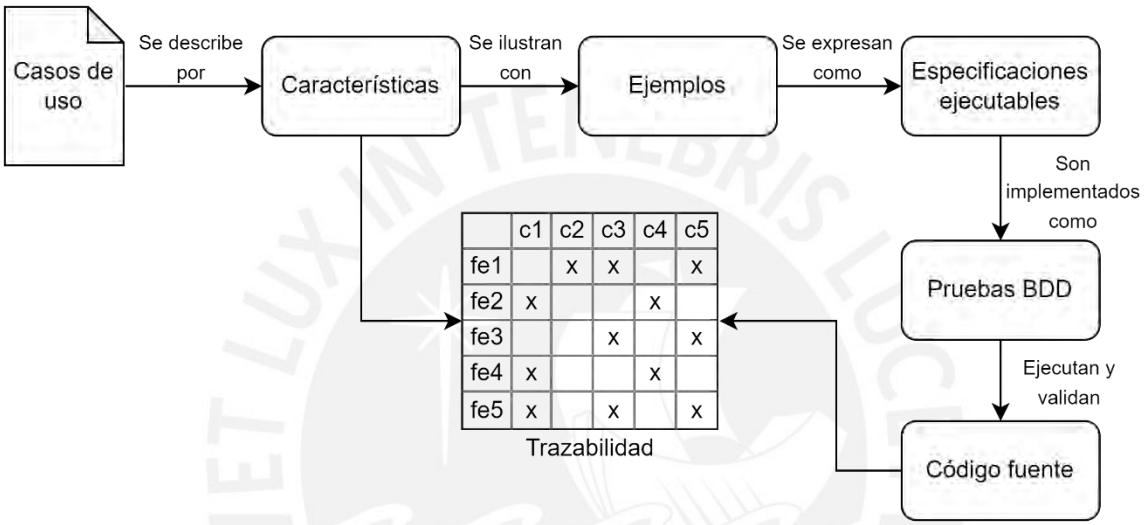
En la especificación de construcción, se indica las clases y métodos necesarios para implementar un caso de uso. Los métodos son especificados por medio de pseudocódigo.

Sin embargo, un caso de uso suele ser implementado por muchas clases y métodos. La trazabilidad no permite identificar fácilmente que clases y métodos implementan ciertas secciones de la especificación del caso de uso. Por esta razón, se suele hacer una revisión documental, que incluye revisar el documento de “Especificación de requerimientos” y el código fuente, con el fin de identificar la trazabilidad hacia el código fuente.

6.2. Definición de la propuesta para garantizar trazabilidad:

Como se ha indicado en la presente investigación, existe una serie de problemas para obtener la información de trazabilidad y que ésta a su vez sea fiable. Es por esto, que la propuesta de esta investigación se orienta al uso de las prácticas de desarrollo guiado por comportamiento (BDD) para garantizar la trazabilidad entre los requerimientos funcionales y el código fuente.

En la “Figura 6.1” se muestra un diagrama en donde se explica la propuesta.



	c1	c2	c3	c4	c5
fe1		x	x		x
fe2	x			x	
fe3			x		x
fe4	x			x	
fe5	x		x		x

Figura 6.1: Propuesta para garantizar trazabilidad con BDD

Como se observa en la Figura 6.1, se parte primero de los casos de uso, estos son luego detallados en características, que permiten desglosar los casos de uso en una o más funcionalidades que el sistema debe ofrecer. Estas características se pueden expresar como historias de usuario.

Las características se ilustran con ejemplos, que definen cómo se debe comportar el sistema al momento de implementar una característica, estos son expresados por medio del lenguaje Gherkin, que es un lenguaje cercano al natural y que facilita la comunicación con los Stakeholders. Los ejemplos son expresados como especificaciones ejecutables, que son generados de forma automática al utilizar *frameworks* como Cucumber o JBehave. Estas especificaciones son la base para continuar con la implementación de las pruebas BDD.

Estas pruebas BDD permitirán validar, por medio de la invocación de las clases y métodos del código fuente, el cumplimiento de los escenarios, representados como pruebas de aceptación para verificar el cumplimiento de las características o requerimientos del sistema.

Para poder entender mejor la propuesta, se plantea el siguiente caso:

Tenemos el caso de uso "CUS01 – Registrar declaración de mercancía (DAM)", como se sabe un caso de uso describe un escenario donde hay una interacción entre el actor (usuario) y el sistema. Como parte de la especificación del caso de uso, hay una sección donde se valida el País destino de la mercancía, y se desea que el sistema al momento de ingresar el país, si es correcto, permita continuar con el registro, caso contrario muestre un mensaje de error.

Lo mencionado da lugar a que el caso de uso "*CUS01 – Registrar declaración de mercancía (DAM)*", tenga la característica "Validar País destino". En la Figura 6.2 se puede ver la característica expresada como una historia de uso. Esta forma de expresar la característica es por medio del lenguaje Gherkin, el cual se puede encontrar disponible en muchos plugins para IDE de desarrollo, en este caso se ha hecho uso de Cucumber para Eclipse.

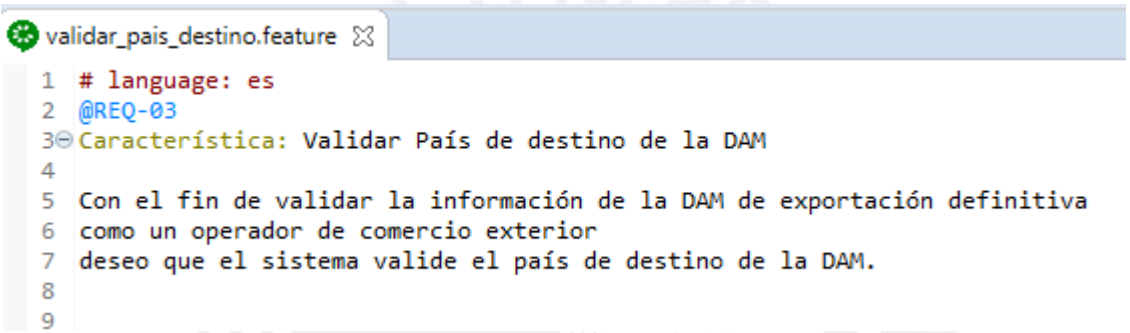


Figura 6.2: característica expresada en Gherkin

El siguiente paso es definir ejemplos que describen el comportamiento de las características en diferentes situaciones, para este caso existen dos situaciones, una donde el país destino es correcto, y otro donde no lo es. En la Figura 6.3 se observa la descripción completa de la característica, con dos escenarios y cada escenario ilustrado con diferentes ejemplos.

```
# language: es
@REQ-03
Característica: Validar País de destino de la DAM

Con el fin de validar la información de la DAM de exportación definitiva
como un operador de comercio exterior
deseo que el sistema valide el país de destino de la DAM.


  Esquema del escenario: El país de destino de la DAM es válido

    Cuando la DAM tiene país de destino con código "<codigo_pais>"
    Y se valida el país de destino de la DAM
    Entonces no hay mensajes de error

    Ejemplos:

      | codigo_pais |
      | PE          |
      | HN          |
      | JP          |

  Esquema del escenario: El país de destino de la DAM no es válido

    Cuando la DAM tiene país de destino con código "<codigo_pais>"
    Y se valida el país de destino de la DAM
    Entonces se obtiene el mensaje de error "País de destino de la DAM no válido"

    Ejemplos:

      | codigo_pais |
      | TA          |
      | HH          |
      | UU          |
```

Figura 6.3: Descripción completa de la característica (feature)

La descripción de las características es transformada a especificaciones ejecutables que se implementan como pruebas BDD. Todo esto puede ser realizado automáticamente usando frameworks BDD, como por ejemplo Cucumber, donde a partir de una definición como la mostrada en la Figura 6.3 se genera una estructura de código fuente, lista para ser implementada y posteriormente probada, en la Figura 6.4 se puede observar como la especificación de las características se ha convertido en una serie de métodos.

```java
@Cuando("^la DAM tiene país de destino con código \"(.*?)\"$")
public void la_DAM_tiene_país_de_destino_con_código(String arg1) throws Throwable {
    Declaracion declaracion = new Declaracion();
    declaracion.setCodPaisDestino(arg1);
    this.validadorDatosDeclaracionSupport.setDeclaracion(declaracion);
}

@Cuando("^se valida el país de destino de la DAM$")
public void se_valida_el_país_de_destino_de_la_DAM() throws Throwable {
    String mensajeError = this.validadorDatosDeclaracionSupport.validarPaisDestino();
    this.manageErrorHelper.addMensajeError(mensajeError);
}


@Entonces("^se obtiene el mensaje de error \"(.*?)\"$")
public void se_obtiene_el_mensaje_de_error(String mensajeError) throws Throwable {
    Assert.assertTrue(this.manageErrorHelper.contieneMensajeError(mensajeError));
}

@Entonces("^no hay mensajes de error$")
public void no_hay_mensajes_de_error() throws Throwable {
    Assert.assertTrue(this.manageErrorHelper.getMensajesError().isEmpty());
}
```

Figura 6.4: Especificación de características en código fuente

Los métodos mostrados en la Figura 6.4 representan cada uno de los pasos descritos en los escenarios, estos se encuentran con sus respectivas anotaciones, y se ejecutaran según el orden en que hayan sido descritos en las características.

Estos métodos inicialmente estarán vacíos, y es el desarrollador, quien se encargará de implementar la lógica, que finalmente llamará los métodos de negocio o propios del sistema.

Hasta el momento se ha definido las características, escenarios y casos de prueba BDD. Con esta información es factible obtener la trazabilidad entre requerimientos funcionales (RF), definidos a través de las características, y los elementos de código fuente invocados por los casos de prueba BDD.

Para el presente estudio, se ha hecho uso de *test coverage*, para identificar los elementos de código fuente que son llamados cuando se ejecutan los casos de prueba BDD.

Para establecer de cual requerimiento queremos obtener su trazabilidad, se hace uso de las anotaciones (tags) provistas por el framework cucumber, esto se aprecia en la Figura 6.5, en donde se busca validar el RF 03:

```java
package pe.gob.sunat.bdd.steps.execution;

import org.junit.runner.RunWith;

import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(tags="@REQ-03", plugin={"pretty", "html:resultados_pruebas"},
                 features ="src/test/resources/features")
public class RunCukesTest {
}
```

Figura 6.5: Clase que inicia la ejecución de Pruebas BDD

Lo mostrado en la Figura 6.5 inicia la ejecución de las pruebas para aquellas características etiquetadas como @REQ-03; y por medio de *Test Coverage*, se puede identificar los elementos de código fuente invocados durante su ejecución. El resultado se muestra en la Figura 6.6 donde se puede observar los elementos de código fuente que estarían vinculados al RF 03.

Figura 6.6: Elementos de código fuente identificados

En resumen, la propuesta se centra en aprovechar la información generada al momento de definir las pruebas BDD, ya que en ellas se tiene información de trazabilidad entre los requerimientos funcionales y los casos de prueba. Así mismo, estas pruebas invocan clases y métodos del código fuente. En base a ello, es posible tener la información de trazabilidad entre requerimientos funcionales y código fuente.

7. Planeamiento del experimento:

Para llevar a cabo la planificación del experimento, se van a seguir los pasos indicados en *(Wohlin et al., 2012, p. 90)*, que pueden ser vistos en la “Figura 7.1”.

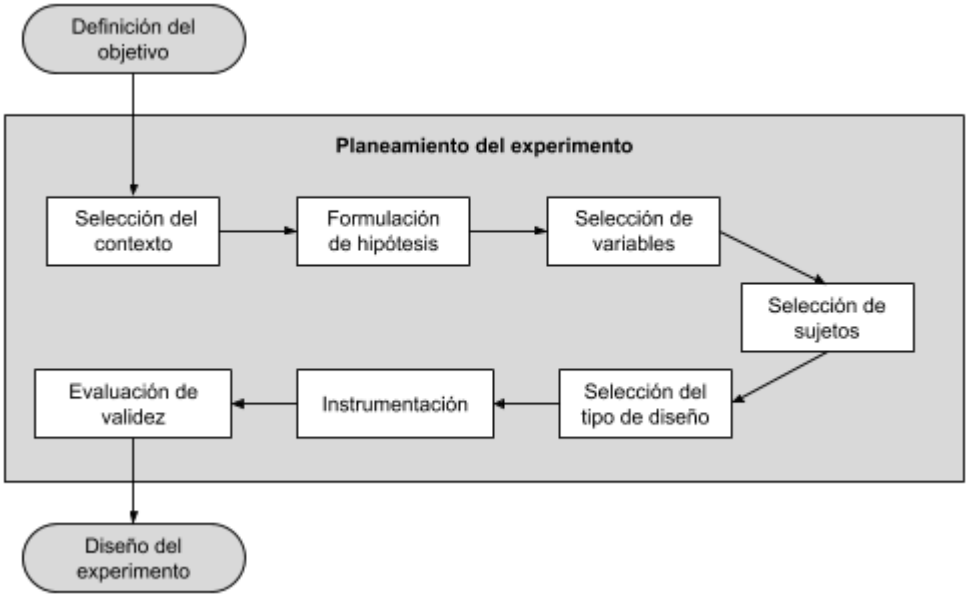


Figura 7.1: Planeamiento del experimento según Wohlin

Como se puede apreciar en la “Figura 7.1”, la fase del planeamiento del experimento se divide en siete pasos. La entrada a la fase de planeamiento es la definición del objetivo del experimento, y la salida es el diseño del experimento.

7.1. Definición del objetivo del experimento

El objetivo del experimento, alineado con lo mencionado en la matriz de marco lógico (Ver “Tabla 3.1”), es evaluar si el desarrollo guiado por comportamiento (BDD) facilita la trazabilidad entre requerimientos funcionales y código fuente.

Para poder realizar esta evaluación, se aplicará el modelo de evaluación de métodos (MEM) propuesto por *(Moody, Sindre, Brasethvik, & Sølvberg, 2002)*. Este modelo (Ver Figura 7.2), a diferencia de otros modelos de evaluación, incorpora dos aspectos: Eficacia actual y adopción en la práctica. Ambos aspectos deben ser tomados en cuenta para evaluar la aceptación de nuevos métodos, en este caso el uso de BDD para facilitar la trazabilidad.

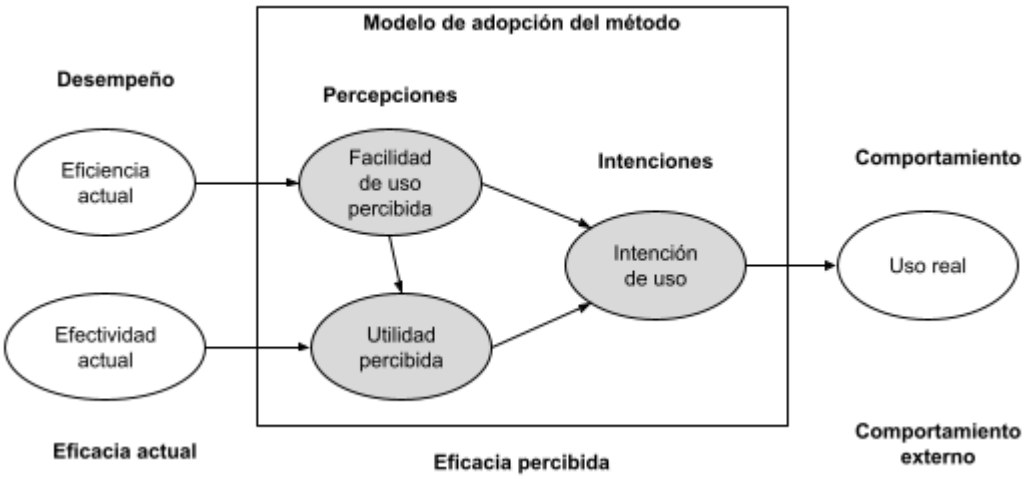


Figura 7.2: Modelo de evaluación de métodos (MEM)

Teniendo en cuenta el MEM, se evalúa la eficacia percibida al utilizar BDD para obtener información de trazabilidad, para esto se debe determinar la: Facilidad de uso percibida, utilidad percibida e intención de uso.

Además, se evaluará si BDD facilita la actualización de los elementos de trazabilidad, determinando:

- El tiempo necesario para actualizar los elementos de trazabilidad.
- Cantidad de omisiones al momento de actualizar los elementos de trazabilidad.

7.2. Selección del contexto

El contexto del experimento es un proyecto que actualmente se encuentra siendo desarrollado por una institución del gobierno. Se pretende aplicar el experimento sobre algunos requerimientos funcionales solicitados en el proyecto. Todos estos requerimientos deben tener similar nivel de complejidad.

El experimento será llevado a cabo por trabajadores de la institución, que son responsables del desarrollo de aplicaciones. La posibilidad de generalizar este experimento a partir de este contexto específico será discutida en mayor detalle en las amenazas a la validez.

7.3. Formulación de la hipótesis:

La institución del gobierno donde se llevará a cabo el experimento utiliza la matriz de trazabilidad de requerimientos (MTR) y revisión documentaria; como técnicas o métodos para registrar, obtener y mantener la información de trazabilidad entre requerimientos funcionales y código fuente.

Se espera que al aplicar BDD; la obtención de información de trazabilidad entre requerimientos funcionales y código fuente, sea más fácil que utilizando el método tradicional.

A partir de las declaraciones anteriores y los objetivos del experimento, se ha planteado las siguientes hipótesis:

H_{10}: No existe diferencia en el tiempo necesario para obtener los elementos de trazabilidad entre el método tradicional y el utilizado usando BDD.
H_{1a}: Existe diferencia en el tiempo necesario para obtener los elementos de trazabilidad entre el método tradicional y el utilizado usando BDD.

H_{20}: No existe diferencia en el número de omisiones al obtener los elementos de trazabilidad entre el método tradicional y el utilizado usando BDD.
H_{2a}: Existe diferencia en el número de omisiones al obtener los elementos de trazabilidad entre el método tradicional y el utilizado usando BDD.

H_{30}: No existe diferencia en la percepción de facilidad de uso entre el método tradicional y el utilizado usando BDD.
H_{3a}: Existe diferencia en la percepción de facilidad de uso entre el método tradicional y el utilizado usando BDD.

H_{40}: No existe diferencia en la percepción de utilidad entre el método tradicional y el utilizado usando BDD.
H_{4a}: Existe diferencia en la percepción de utilidad entre el método tradicional y el utilizado usando BDD.

H_{50}: No existe intención de utilizar el método propuesto en el futuro.
H_{5a}: Existe intención de utilizar el método propuesto en el futuro.

7.4. Selección de variables:

La variable independiente para cada hipótesis es el método utilizado para trabajar la trazabilidad entre requerimientos funcionales y código fuente. Este método puede ser uno de los siguientes:

- Método tradicional: El que actualmente es usado por la institución donde se llevará a cabo el experimento. Este método contempla la técnica de matriz de trazabilidad de requerimientos y revisión documentaria de las diversas fuentes de información.
- Método utilizando BDD: El método que se propone que es la obtención automática de trazabilidad de requerimientos usando el enfoque de desarrollo guiado por comportamiento (BDD).

La variable dependiente para cada hipótesis es detallada en la “Tabla 7.1”.

Hipótesis	Variable dependiente
H_1	Tiempo promedio para completar la información de trazabilidad.
H_2	Número omisiones al momento de completar la información de trazabilidad.
H_3	Facilidad de uso percibida.
H_4	Utilidad percibida.
H_5	Intención de uso.

Tabla 7.1: Variables dependientes para las hipótesis

Para medir las variables dependientes de las hipótesis H_1y H_2se realizará un experimento en donde se identificará el tiempo promedio y el número de omisiones al completar la información de trazabilidad de una lista de requerimientos funcionales.

El tiempo promedio, sería el tiempo promedio en minutos que se necesita para completar la información de trazabilidad. Por cada requerimiento funcional, se anotará la hora de inicio y fin al registrar la información de trazabilidad, luego se obtendrá la diferencia en minutos, que será el tiempo por promediar.

Las omisiones será el número de registros de información de trazabilidad que no fueron completados durante el desarrollo del experimento.

Para medir las variables dependientes de las hipótesis H_3, H_4 y H_5 se utilizará un cuestionario para evaluar la percepción, que puede ser visto en el Anexo 7. Cada una de estas preguntas evalúa un aspecto; que puede ser facilidad de uso percibida (PEOU), utilidad percibida (PU) e intención de uso (ITU). La distribución de las preguntas según estos criterios se observa en la Tabla 7.2

Criterio de percepción	ID de las Preguntas
Facilidad de uso percibida (PEOU)	P1, P4, P7, P10 y P12
Utilidad percibida (PU)	P14, P2, P5, P6, P9 y P11
Intención de uso (ITU)	P3, P8 y P13

Tabla 7.2: Distribución de preguntas por criterio de percepción

7.5. Selección de sujetos:

Los sujetos del estudio son los trabajadores de una de las supervisiones de la institución del gobierno que está llevando a cabo un proyecto de desarrollo e implementación de software.

Los trabajadores seleccionados son los encargados del diseño y la construcción de los diferentes requerimientos funcionales que llegan a la supervisión. Estos trabajadores tienen en promedio 5 años de experiencia en análisis, diseño y desarrollo de software.

7.6. Selección del tipo de diseño

En este paso del planeamiento del experimento se discute los métodos de análisis estadístico a utilizar para determinar la aceptación o rechazo de las hipótesis planteadas.

Los datos recabados serán evaluados para verificar si estos se ajustan a una distribución normal. Para esto, se hará uso del test de Shapiro-Wilk *(Shapiro & Wilk, 1965)*, el cual resulta conveniente para nuestro caso, ya que la muestra será pequeña (menor a 30).

Para cada hipótesis planteada, si el conjunto de datos sigue una distribución normal, se hará uso de pruebas paramétricas, caso contrario, se hará uso de pruebas no paramétricas.

Para las hipótesis H_1 y H_2, en caso el conjunto de datos siga una distribución normal, se usará la prueba t-student pareada *(Student, 1908)*, caso contrario, se usará la prueba Wilcoxon *(Wilcoxon, 1945)*.

Para las hipótesis H_3, H_4 y H_5, se desarrollará un cuestionario con respuestas simples basadas en una escala Likert de 5 niveles. La confiabilidad y la validez será evaluada en este cuestionario. Para la confiabilidad se hará uso del coeficiente *Alfa de Cronbach (Cronbach, 1951)*, y para la validez se hará uso de la validez convergente y divergente *(Campbell & Fiske, 1959)*.

Una vez evaluada la confiabilidad y validez del cuestionario, se procederá a evaluar las hipótesis H_3, H_4 y H_5, según sea, si el conjunto de datos sigue una distribución normal, se hará uso de la prueba t-student para una muestra *(Student, 1908)*, caso contrario, se usará la prueba Wilcoxon *(Wilcoxon, 1945)*.

7.7. Instrumentación:

Para el desarrollo del experimento, se han utilizado los siguientes instrumentos:

- Cuestionario de autoevaluación: El objetivo de este cuestionario es recabar, por cada participante, información relacionada a la experiencia laboral en el diseño y desarrollo de aplicaciones, así como el conocimiento en las prácticas de desarrollo guiado por comportamiento (BDD). Este cuestionario puede ser visto en el Anexo 4.

- Presentación de casos para actualización de trazabilidad: Se presentará dos casos prácticos, basados en requerimientos previamente implementados. Cada uno de estos casos, plantea identificar la información de trazabilidad. Un caso, será utilizando los métodos tradicionales y el otro utilizando BDD. Los casos prácticos pueden ser vistos en los Anexos 5 y 6.

- Formulario de resultados en la actualización de trazabilidad: Con el fin de registrar el tiempo necesario y los elementos que han sido omitidos durante la actualización de la información de trazabilidad. Este formulario puede ser visto en el Anexo 8.

- Cuestionario para el análisis de percepción: Se van a utilizar cuestionarios para la evaluación de la percepción de facilidad de uso, percepción de uso e intención de uso de cada uno de los métodos para trabajar la trazabilidad de requerimientos. Los cuestionarios estarán conformados por preguntas con respuestas de selección simple basado en una escala de Likert de 5 niveles. Este cuestionario se encuentra en el Anexo 7.

La evaluación de la validez, se indica en la sección 11 del presente documento. El diseño del experimento se encuentra detallado en la sección 9, en esta parte se aplica los métodos de análisis estadístico, definidos en el punto 7.6, lo que llevará al rechazo o aceptación de las hipótesis planteadas.

8. Procedimiento:

Para el desarrollo del experimento, se seguirá el siguiente procedimiento:

Preparación (1 semana):

- Seleccionar a los participantes del experimento.

- Presentar a los participantes la hoja informativa (ver Anexo 2) donde se explica el propósito de la investigación, y las actividades a desarrollar.

- Solicitar a los participantes la firma del formulario del consentimiento (Ver Anexo 3), donde manifiestan su voluntad de participar en el experimento.

- Elaborar un primer cuestionario para recabar información referente a conocimientos y experiencia profesional de cada uno de los participantes en el experimento. En este cuestionario también se busca identificar si el participante tiene conocimiento en BDD, así como su opinión sobre cuáles son los elementos más complicados de obtener en la información de trazabilidad.

- Definir una lista de requerimientos funcionales, para los que se desea obtener la información de trazabilidad hacia el código fuente (clases y métodos asociados).

- Preparar dos casos prácticos, cada uno con la misma cantidad de requerimientos funcionales, así mismo los requerimientos utilizados en un caso práctico no pueden ser utilizados en el otro.

- En el primer caso práctico (Anexo 5), el participante buscará obtener la información de trazabilidad hacia el código fuente utilizando la forma actual de trabajo, que es mediante la matriz de trazabilidad y revisión documental.

- En el segundo caso práctico (Anexo 6), el participante buscará obtener la información de trazabilidad hacia el código fuente utilizando la propuesta de la presente investigación, que es por medio de BDD.

- Elaborar un segundo cuestionario para evaluar (Anexo 7): facilidad de uso, utilidad percibida e intención de uso; para la propuesta de la presente investigación.

Pre-prueba (1 hora):

- Cada participante responde el primer cuestionario para recabar datos relevantes a sus conocimientos y experiencia profesional.

- Cada participante desarrollará el primer caso práctico mostrado en el Anexo 5, con un tiempo límite de una hora. En cada requerimiento funcional, el participante apuntará la hora de inicio y la hora en que finalizó de identificar la información de trazabilidad. Se espera que por cada requerimiento se identifique todos los métodos involucrados en su construcción, así como el tiempo que tardo en identificarlos.

Aplicación de tratamiento (1 hora):

- Cada participante recibirá una capacitación sobre desarrollo guiado por comportamiento (BDD). Luego, como parte de la capacitación, se desarrollará un caso práctico utilizando BDD. El objetivo es que los participantes tengan el conocimiento necesario y se encuentren familiarizarlos con BDD.

Post-prueba (1 hora):

- Al igual que en la pre-prueba, cada participante tendrá una hora para desarrollar el segundo caso práctico, que contiene la otra lista de requerimientos funcionales, por cada uno de estos, el participante apuntará la hora de inicio y la hora en que finalizó de identificar la información de trazabilidad.

- Se completa el segundo cuestionario, teniendo en cuenta la propuesta con BDD. Esto con el fin de evaluar su: facilidad de uso, utilidad percibida e intención de uso.

9. Ejecución del experimento:

A continuación, se detalla cada uno de los resultados obtenidos durante el desarrollo del experimento.

9.1. Cuestionario de autoevaluación

Inicialmente se tenía previsto que fuesen 9 participantes, pero uno no pudo presentarse, por lo que sólo se pudo trabajar con 8 participantes. A cada uno de ellos se aplicó el cuestionario mostrado en el Anexo 4.

En la Figura 9.1 se pueden ver los resultados a la primera pregunta, donde todos los participantes tienen una experiencia mayor a 5 años. Hay 5 participantes (62.5%) que tienen una experiencia mayor a 10 años.

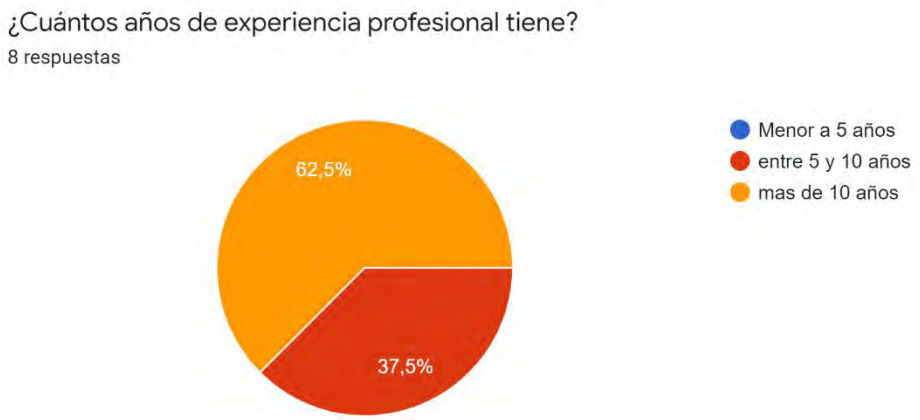


Figura 9.1: Años de experiencia profesional

Ya que los casos prácticos eran de requerimientos implementados con Java, se deseaba conocer la experiencia con Java, los resultados se muestran en la Figura 9.2, donde un participante (12.5%) tiene una experiencia menor a cinco años.

Figura 9.2: Experiencia en Java

En la Figura 9.3, donde el eje “X” muestra el número de participantes y el eje “Y” la etapa del desarrollo de software; se observa que todos tienen experiencia en la implementación (Construcción). Esto resulta de importancia, pues en BDD se necesita codificar las pruebas, lo cual requiere de experiencia en la construcción de software.

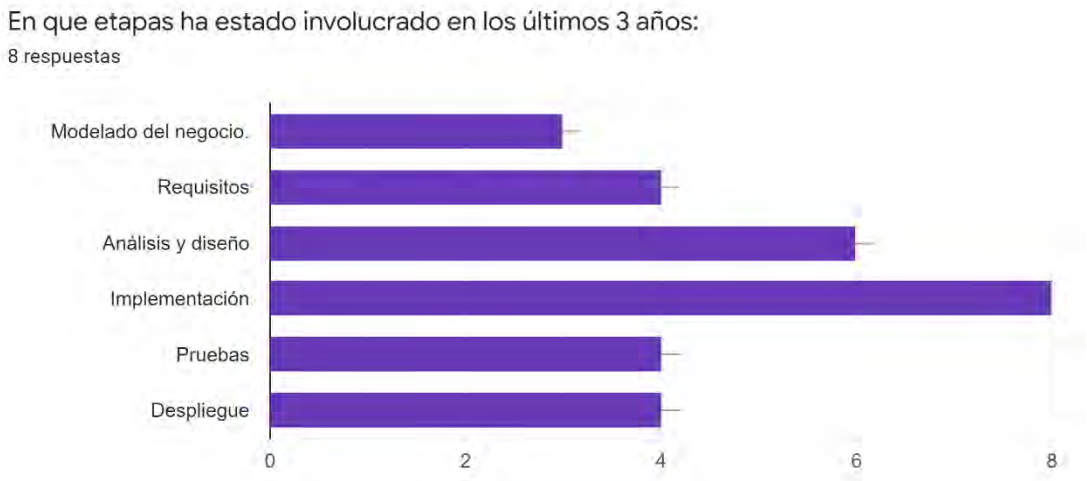


Figura 9.3: Etapas involucradas en el proceso de software

BDD resulta en cierto aspecto similar a TDD, por esta razón se planteó la pregunta mostrada en la Figura 9.4, donde se aprecia que un participante nunca había escuchado de TDD.

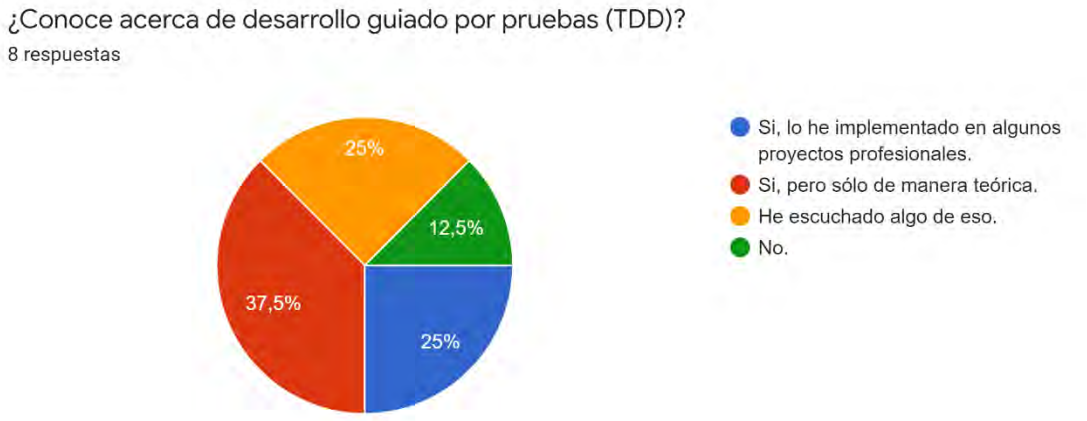


Figura 9.4: Conocimiento sobre TDD

En la Figura 9.5, se observa que ninguno tiene experiencia implementando BDD, la mayoría sólo ha escuchado el término o tiene algún conocimiento teórico. Durante el desarrollo del experimento, con el fin de ofrecer un conocimiento general, se ofreció una breve capacitación sobre como implementar BDD.

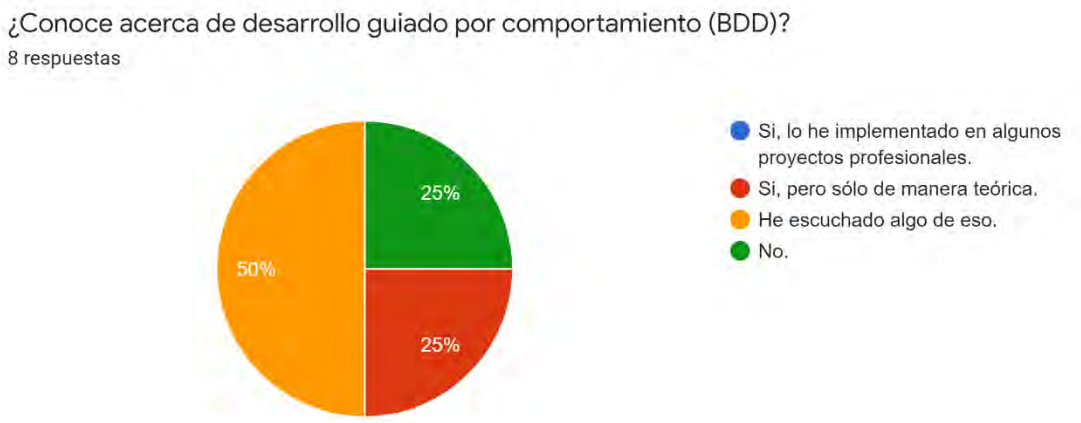


Figura 9.5: Conocimiento sobre BDD

La preguntada mostrada en la Figura 9.6, se realizó para conocer la experiencia de los participantes con algunos frameworks para el desarrollo de pruebas, pues resulta de importancia al momento de querer implementar BDD en un proyecto.

Figura 9.6: Conocimiento sobre Herramientas

Figura 9.7: Elementos de trazabilidad fáciles de obtener

En la Figura 9.7 se observa que, la especificación de casos de uso y los componentes, son los artefactos más fáciles de obtener al querer tener una trazabilidad con los requerimientos funcionales. Esto resulta así, ya que en la entidad donde se realizó el experimento, esta información siempre es registrada como parte del entregable, y es guardada en un repositorio.

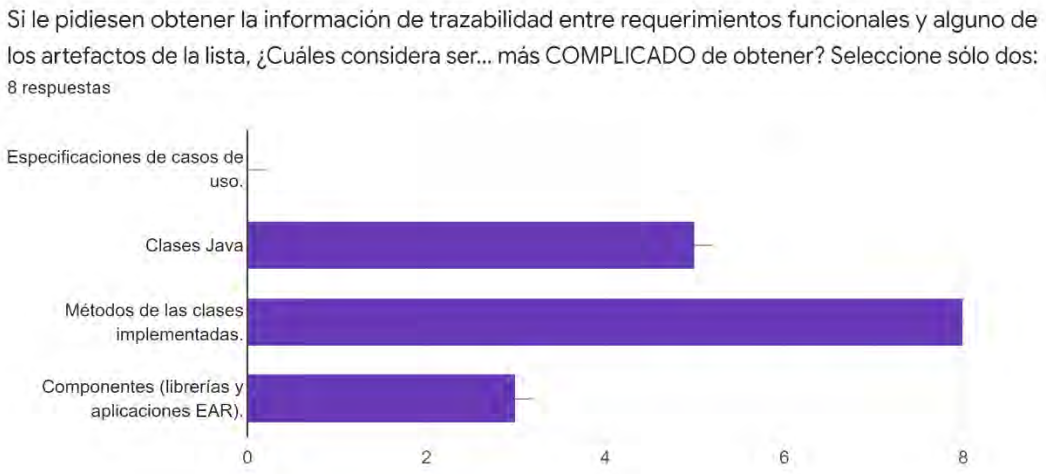


Figura 9.8: Elementos de trazabilidad complicados de obtener

En la Figura 9.8 se observa que la mayoría de los participantes coincide en que los elementos de trazabilidad más complicados de obtener son las clases y los métodos Java, que son los elementos que se buscan identificar con la propuesta de trazabilidad.

9.2. Evaluación de las variables de percepción

Teniendo en cuenta lo mencionado en la sección 7.3, se han planteado las siguientes hipótesis para evaluar la percepción sobre la facilidad de uso, utilidad e intención de uso.

H_{30}: No existe diferencia en la percepción de facilidad de uso entre el método tradicional y el utilizado usando BDD.
H_{3a}: Existe diferencia en la percepción de facilidad de uso entre el método tradicional y el utilizado usando BDD.

H_{40}: No existe diferencia en la percepción de utilidad entre el método tradicional y el utilizado usando BDD.
H_{4a}: Existe diferencia en la percepción de utilidad entre el método tradicional y el utilizado usando BDD.

H_{50}: No existe intención de utilizar el método propuesto en el futuro.

H_{5a}: Existe intención de utilizar el método propuesto en el futuro.

Cada una de estas hipótesis busca evaluar los tres criterios de percepción indicados en la Tabla 7.2. Para esto se utilizó el cuestionario de 14 preguntas del Anexo 7. Los resultados de aplicar el cuestionario de percepción se pueden ver en el Anexo 9.

Previo al análisis de los resultados del cuestionario, se evaluó su validez, para verificar si las preguntas planteadas son útiles para valorar los criterios de percepción. Para este fin, se hará uso de la validez convergente y divergente *(Campbell & Fiske, 1959)*.

La validez convergente señala que la correlación entre los indicadores usados para medir el mismo constructo debe ser lo más alta posible. La validez divergente señala que la correlación entre los indicadores usados para medir diferentes constructos deberá ser lo más baja posible. Según *(Campbell & Fiske, 1959)*, un ítem será válido si la validez convergente es mayor que la divergente.

	PEOU					PU						ITU					
	P1	P4	P7	P10	P12	P14	P2	P5	P6	P9	P11	P3	P8	P13	CV	DV	VALID
P1	1.000	0.143	-0.218	-0.267	0.293	0.655	0.488	-0.267	0.488	0.378	0.204	-0.079	0.293	0.143	0.190	0.256	NO
P4	0.143	1.000	0.218	0.267	0.488	0.218	-0.488	0.267	-0.488	-0.378	-0.747	0.079	0.488	-0.143	0.423	-0.132	SI
P7	-0.218	0.218	1.000	0.816	0.447	0.333	0.149	-0.408	0.149	0.000	-0.104	0.361	-0.149	0.655	0.453	0.110	SI
P10	-0.267	0.267	0.816	1.000	0.183	0.408	0.183	0.000	0.183	0.354	-0.127	0.737	0.183	0.535	0.400	0.273	SI
P12	0.293	0.488	0.447	0.183	1.000	0.447	0.067	-0.183	0.067	-0.258	-0.046	-0.269	0.467	0.488	0.482	0.087	SI
P14	0.655	0.218	0.333	0.408	0.447	1.000	0.745	-0.408	0.745	0.577	0.311	0.361	0.447	0.655	0.495	0.441	SI
P2	0.488	-0.488	0.149	0.183	0.067	0.745	1.000	-0.548	1.000	0.775	0.788	0.269	0.067	0.683	0.627	0.177	SI
P5	-0.267	0.267	-0.408	0.000	-0.183	-0.408	-0.548	1.000	-0.548	0.000	-0.381	0.147	0.548	-0.535	-0.147	-0.054	NO
P6	0.488	-0.488	0.149	0.183	0.067	0.745	1.000	-0.548	1.000	0.775	0.788	0.269	0.067	0.683	0.627	0.177	SI
P9	0.378	-0.378	0.000	0.354	-0.258	0.577	0.775	0.000	0.775	1.000	0.539	0.626	0.258	0.378	0.611	0.170	SI
P11	0.204	-0.747	-0.104	-0.127	-0.046	0.311	0.788	-0.381	0.788	0.539	1.000	0.112	-0.046	0.611	0.508	-0.018	SI
P3	-0.079	0.079	0.361	0.737	-0.269	0.361	0.269	0.147	0.269	0.626	0.112	1.000	0.162	0.394	0.519	0.238	SI
P8	0.293	0.488	-0.149	0.183	0.467	0.447	0.067	0.548	0.067	0.258	-0.046	0.162	1.000	0.098	0.420	0.238	SI
P13	0.143	-0.143	0.655	0.535	0.488	0.655	0.683	-0.535	0.683	0.378	0.611	0.394	0.098	1.000	0.497	0.377	SI

Tabla 9.1: Matriz de correlaciones entre elementos

Se aprecia en la Tabla 9.1, para las preguntas P1 y P5, la validez convergente (CV) es menor que la validez divergente (DV), por tanto, estas serán excluidas del análisis.

El análisis continúa para determinar la confiabilidad del cuestionario, es decir que si al aplicar dos o más veces este cuestionario al mismo grupo de individuos se obtendrán resultados similares. Para esto, se hará uso del valor α de Cronbach.

	Alfa de Cronbach	Número de elementos
PEOU	0.705	4
PU	0.910	5
ITU	0.469	3
Total	**0.833**	**12**

Tabla 9.2: Confiabilidad del cuestionario de percepción

En la Tabla 9.2 se observa que el valor α de Cronbach para el cuestionario de percepción es 0.8333, que es mayor al valor mínimo de 0.7 recomendado por *(Nunnally, 1978)* para estudios exploratorios. Por tanto, se concluye que el cuestionario es confiable.

Para verificar si los datos siguen una distribución normal, se utilizará la prueba de "Shapiro-wilk" (Shapiro & Wilk, 1965), que sirve para muestras pequeñas (menor a 30).

Para esta prueba, se tiene como hipótesis nula que el conjunto de datos sigue una distribución normal, y como hipótesis alternativa, que los datos no siguen una

distribución normal. La prueba se hizo teniendo en cuenta un nivel de significancia de 0.05.

Si el valor “p” es mayor que el nivel de significancia, entonces la hipótesis nula se acepta; sin embargo, si el valor “p” es menor que el nivel de significancia, entonces la hipótesis nula se rechaza.

En la Tabla 9.3, se pueden observar los resultados obtenidos.

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PEOU	,313	8	,020	,903	8	,304
PU	,304	8	,028	,735	8	,006
ITU	,226	8	,200*	,899	8	,283

Tabla 9.3: Pruebas de normalidad

Del resultado obtenido en la Tabla 9.3, se puede observar que el nivel de significancia obtenido es mayor a 0.05 para las variables PEOU e ITU, por tanto, estas siguen una distribución normal mientras que la variable PU no sigue una distribución normal.

Habiendo realizado las pruebas de normalidad, se procede a probar las hipótesis H_3, H_4, H_5. Para aquellos datos que sigan una distribución normal, se hará uso de la Prueba-t para una muestra, caso contrario se hará uso de la prueba de Wilcoxon. (Wilcoxon, 1945)

Para las pruebas se ha asumido un valor de la media poblacional de referencia igual a 3, ya que este es el valor neutral según la escala de Likert de 5 niveles.

Se realiza la prueba t-student para verificar la hipótesis H_3 y H_5 , los resultados se muestran en la Tabla 9.4

Prueba para una muestra

	Valor de prueba = 3					
					95% de intervalo de confianza de la diferencia	
	t	gl	Sig. (bilateral)	Diferencia de medias	Inferior	Superior
PEOU	6,708	7	,000	,937500	,60703	1,26797
ITU	7,561	7	,000	1,16667	,8018	1,5315

Tabla 9.4: Resultados de la prueba t para una muestra

En la Tabla 9.4, se observa que el valor de significancia es 0, siendo menor al valor 0.05, por tanto, para la muestra se rechaza la hipótesis nula en H_3 y H_5, lo que indica que si existe una diferencia en la percepción de facilidad de uso e intención de uso. También se observa que hay una probabilidad del 95%, que el valor de la media para la facilidad de uso percibida (PEOU) se encuentre entre 3.607 y 4.26, siendo esto mayor que 3, por tanto, en la muestra hay evidencia que la facilidad de uso percibida (PEOU) con el método propuesto es mayor, esto nos lleva aceptar la hipótesis alternativa de H_3.

Para la percepción de intención de uso (ITU) se observa que hay un 95% de probabilidad que el valor de la media se encuentre entre 3.801 y 4.531, siendo mayor que 3, por tanto, en la muestra hay evidencia que la intención de uso percibida (ITU) con el método propuesto es mayor, esto nos lleva aceptar la hipótesis alternativa de H_5.

Se realiza la prueba Wilcoxon para verificar la hipótesis H_4, los resultados se muestran en la Tabla 9.5

Resumen de prueba de hipótesis

	Hipótesis nula	Prueba	Sig.	Decisión
1	La mediana de PU es igual a 3,00.	Prueba de rangos con signo de Wilcoxon para una muestra	,011	Rechazar la hipótesis nula.

Se muestran significaciones asintóticas. El nivel de significación es de ,05.

Tabla 9.5: Resultados de la prueba de Wilcoxon

En la Tabla 9.5 se observa que se rechaza la hipótesis nula, por tanto, el valor de la mediana para Utilidad percibida (PU) es diferente a 3. En la Figura 9.9 se observa que la mediana observada es 4.8, lo cual nos lleva a decir que para la muestra se observa que existe intención de uso de utilizar el método propuesto en el futuro, por tanto, se acepta la hipótesis alternativa de H_4 .

La hipótesis nula, indica que no habrá una diferencia de percepción, esto si los valores de la muestra tienen una media igual a 3, sino es igual, entonces se puede afirmar que existe una diferencia en la utilidad percibida.

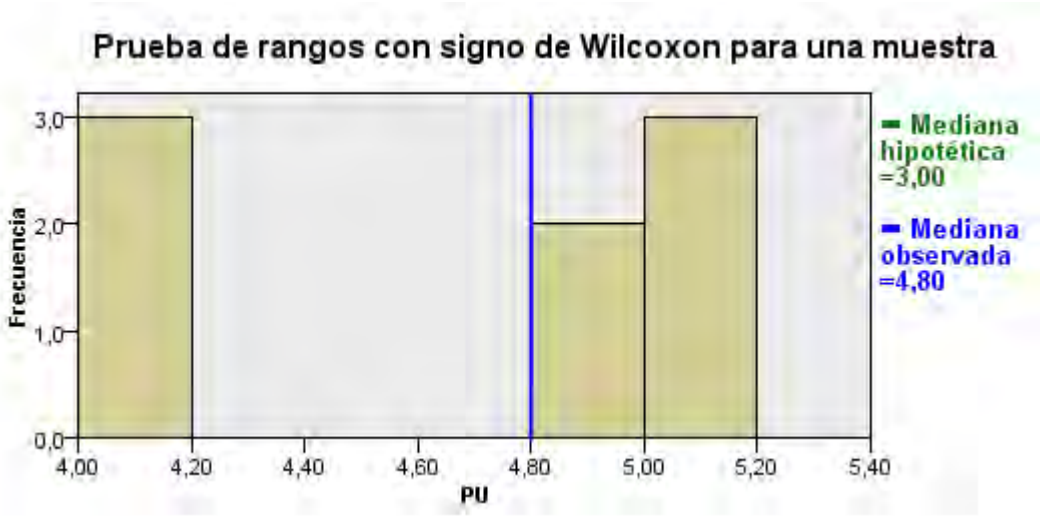


Figura 9.9: Mediana observada

9.3. Evaluación del tiempo promedio

Se había planteado la siguiente hipótesis:

H_{10}: No existe diferencia en el tiempo necesario para obtener los elementos de trazabilidad entre el método tradicional y el utilizado usando BDD.
H_{1a}: Existe diferencia en el tiempo necesario para obtener los elementos de trazabilidad entre el método tradicional y el utilizado usando BDD.

Con el fin de determinar si existe una diferencia significativa en el tiempo utilizado para obtener los elementos de trazabilidad entre la propuesta utilizando BDD frente a la forma tradicional.

En la Tabla 9.6 se observa el tiempo en minutos, sin y con la propuesta, necesario para actualizar los elementos de trazabilidad.

Participante	Tiempo utilizado (minutos)	
	Sin la propuesta	Con la propuesta
1	60	15
2	42	13
3	38	6
4	43	23
5	60	12
6	47	10
7	58	5
8	60	20

Tabla 9.6: Tiempo utilizado

Antes de evaluar las hipótesis, será necesario determinar si los elementos de la muestra siguen una distribución normal. Ya que la muestra es pequeña (menor a 30), se utilizará la prueba de Shapiro-Wilk, en la Tabla 9.7 se pueden observar los resultados.

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Sin la propuesta	,271	8	,086	,821	8	,048
Con la propuesta	,125	8	,200*	,957	8	,780

Tabla 9.7: Prueba de normalidad

De los resultados de la Tabla 9.7, se puede observar que los datos obtenidos al aplicar la propuesta siguen una distribución normal, ya que el valor p es 0.78, que es mayor a 0.05. Para los datos obtenidos sin aplicar la propuesta, se observa que no siguen una distribución normal, pues el valor p es 0.048, que es menor a 0.05.

Ya que ambos datos no siguen una distribución normal, entonces para contrastar la hipótesis será necesario utilizar pruebas no paramétricas, en este caso se usará la prueba de Wilcoxon para muestras relacionadas.

	Con la propuesta - Sin la propuesta
Z	-2,521
Sig. asintótica(bilateral)	,012

Tabla 9.8: Prueba de Wilcoxon (Tiempo utilizado)

En la tabla 9.8 se observan los resultados de la prueba de Wilcoxon, en donde se ha obtenido un valor p=0.012 siendo menor al nivel de significancia 0.05, esto nos lleva a rechazar la hipótesis nula. Por tanto, se puede decir que para la muestra hay evidencia de una diferencia significativa entre el tiempo necesario para actualizar los elementos de trazabilidad al utilizar y no la propuesta.

En la Tabla 9.9 se observan los rangos, en donde para todos los casos el tiempo al utilizar la propuesta fue menor.

		N	Rango promedio	Suma de rangos
Con la propuesta - Sin la propuesta	Rangos negativos	8[a]	4,50	36,00
	Rangos positivos	0[b]	,00	,00
	Empates	0[c]		
	Total	8		

a. Con la propuesta < Sin la propuesta
b. Con la propuesta > Sin la propuesta
c. Con la propuesta = Sin la propuesta

Tabla 9.9: Rangos (Tiempo utilizado)

9.4. Evaluación del número de omisiones

Se ha planteado la siguiente hipótesis:

H_{20}: No existe diferencia en el número de omisiones al obtener los elementos de trazabilidad entre el método tradicional y el utilizado usando BDD.
H_{2a}: Existe diferencia en el número de omisiones al obtener los elementos de trazabilidad entre el método tradicional y el utilizado usando BDD.

Para determinar si existe una diferencia significativa en el número de omisiones al obtener los elementos de trazabilidad entre la propuesta utilizando BDD frente a la forma tradicional.

En la Tabla 9.10 se observa el número de omisiones obtenidas, sin y con la propuesta, durante la actualización de los elementos de trazabilidad.

Participante	Número de omisiones	
	Sin la propuesta	Con la propuesta
1	5	0
2	6	0
3	4	0
4	7	1
5	7	3
6	3	0
7	2	2
8	0	0

Tabla 9.10: Número de omisiones

Antes de evaluar las hipótesis, será necesario determinar si los elementos de la muestra siguen una distribución normal. Ya que la muestra es pequeña (menor a 30), se utilizará la prueba de Shapiro-Wilk, en la Tabla 9.11 se pueden observar los resultados.

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Sin la propuesta	,135	8	,200*	,941	8	,623
Con la propuesta	,365	8	,002	,724	8	,004

Tabla 9.11: Prueba de normalidad

De los resultados de la Tabla 9.11, se puede observar que los datos obtenidos sin aplicar la propuesta siguen una distribución normal, ya que el valor p es 0.623, que es mayor a 0.05. Para los datos obtenidos aplicando la propuesta, se observa que no sigue una distribución normal, pues el valor p es 0.004, que es menor a 0.05.

Ya que ambos datos no siguen una distribución normal, entonces para contrastar la hipótesis será necesario utilizar pruebas no paramétricas, en este caso se usará la prueba de Wilcoxon para muestras relacionadas.

	Con la propuesta - Sin la propuesta
Z	-2,214[b]
Sig. asintótica(bilateral)	,027

Tabla 9.12: Prueba de Wilcoxon (número de omisiones)

En la tabla 9.12 se observan los resultados de la prueba de Wilcoxon, en donde se ha obtenido un valor p=0.027 siendo menor al nivel de significancia 0.05, esto nos lleva a rechazar la hipótesis nula. Por tanto, se puede afirmar que para la muestra hay evidencia de una diferencia significativa entre el número de omisiones incurridos al utilizar y no la propuesta.

En la Tabla 9.13 se observan los rangos, en donde para 6 casos el número de omisiones al utilizar la propuesta fue menor, y dos casos donde no hubo diferencia en número de omisiones encontradas.

		N	Rango promedio	Suma de rangos
Con la propuesta - Sin la propuesta	Rangos negativos	6[a]	3,50	21,00
	Rangos positivos	0[b]	,00	,00
	Empates	2[c]		
	Total	8		

a. Con la propuesta < Sin la propuesta
b. Con la propuesta > Sin la propuesta
c. Con la propuesta = Sin la propuesta

Tabla 9.13: Rangos (número de omisiones)

9.5. Análisis de las relaciones causales

El objetivo de esta sección es validar las relaciones entre los constructos definido en el modelo de evaluación de métodos (MEM), con excepción del uso real. Para esto, se ha realizado un análisis de regresión lineal, ya que esta permite identificar si existe una relación de causalidad entre las variables. Para realizar el análisis se utilizará los siguientes niveles de significancia definidos por *(Moody, 2001)*

Valor de significancia	Rango
No significativo	$p > 0.1$
Baja significancia	$p < 0.1$
Media significancia	$p < 0.05$
Alta significancia	$p < 0.01$
Muy alta significancia	$p < 0.001$

Tabla 9.14: Niveles de significancia

- **Eficiencia vs. Facilidad de Uso Percibida (PEOU)**

  La eficiencia se ha definido por medio del tiempo utilizado para actualizar los elementos de trazabilidad planteado en la hipótesis H_1. Para analizar la causalidad se ha definido un modelo de regresión lineal simple, en donde el tiempo utilizado ha sido definido como la variable predictora (independiente) y PEOU como variable dependiente (predicha), el resultado se observa en la Tabla 9.15.

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados			
		B	Desv. Error	Beta	t	Valor-p.	R2
1	(Constante)	4,043	,364		11,123	,000	
	tiempo	-,008	,025	-,129	-,320	,760	,017

a. Variable dependiente: PEOU

Tabla 9.15: Regresión lineal (Tiempo / PEOU)

Según los resultados mostrados en la Tabla 9.15 se observa que el modelo de regresión fue encontrado no significativo, pues el valor p es de 0.760, siendo mayor a 0.1. También se observa un valor $R^2 = 0.017$ lo cual quiere decir que sólo el 1.7% de la variabilidad en la facilidad de uso percibida es explicado por el modelo de regresión lineal. Por tanto, se puede afirmar que la facilidad de uso percibida (PEOU) no se ve influenciada por el tiempo utilizado para actualizar los elementos de trazabilidad.

- **Efectividad vs Utilidad Percibida**

  Para evaluar la efectividad se ha definido el número de omisiones planteado en la hipótesis H_2. Para analizar la causalidad se ha definido un modelo de regresión lineal simple, en donde el número de omisiones ha sido definido como la variable predictora (independiente) y la utilidad percibida (PU) como variable dependiente (predicha), el resultado se observa en la Tabla 9.16.

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados			
		B	Desv. Error	Beta	t	Valor-p.	R2
1	(Constante)	4,626	,221		20,935	,000	
	Omisiones	-,068	,167	-,165	-,410	,696	,027

a. Variable dependiente: PU

Tabla 9.16: Regresión lineal (Omisiones / PU)

Según los resultados mostrados en la Tabla 9.16 se observa que el modelo de regresión fue encontrado no significativo, pues el valor p es de 0.696, siendo mayor a 0.1. También se observa un valor $R^2 = 0.027$ lo cual quiere decir que sólo el 2.7% de la variabilidad en la utilidad percibida es explicado por el modelo de regresión lineal. Por tanto, se puede afirmar que la utilidad percibida (PU) no se ve influenciada por el número de omisiones obtenidos al momento de actualizar los elementos de trazabilidad.

- **PEOU vs Utilidad Percibida**

  Se analizará si la utilidad percibida (PU) es determinada por la facilidad de uso percibida (PEOU). Para esto se define un modelo de regresión lineal donde la variable predictora (independiente) es PEOU y la utilidad percibida (PU) es la variable dependiente, el resultado se observa en la Tabla 9.17.

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados			
		B	Desv. Error	Beta	t	Valor-p.	R2
1	(Constante)	4,440	1,973		2,250	,065	
	PEOU	-,034	,499	,028	,069	,947	,001

a. Variable dependiente: PU

Tabla 9.17: Regresión lineal (PEOU / PU)

Según los resultados mostrados en la Tabla 9.17 se observa que el modelo de regresión fue encontrado no significativo, pues el valor p es de 0.947, siendo mayor a 0.1. También se observa un valor $R^2 = 0.001$ lo cual quiere decir que sólo el 0.1% de la variabilidad en la utilidad percibida (PU) es explicado por el modelo de regresión lineal. Por tanto, se puede afirmar que la utilidad percibida (PU) no se ve influenciada por la facilidad de uso percibida (PEOU).

- **Intención de Uso Vs Utilidad Percibida**

Se analizará si la intención de uso (ITU) es determinada por la utilidad percibida (PU). Para esto se define un modelo de regresión lineal donde la variable predictora (independiente) es PU y la intención de uso (ITU) es la variable dependiente, el resultado se observa en la Tabla 9.18.

Modelo		Coeficientes no estandarizados B	Desv. Error	Coeficientes estandarizados Beta	t	Valor-p.	R2
1	(Constante)	1,633	1,347		1,212	,271	
	PU	,554	,293	,611	1,890	,108	,373

a. Variable dependiente: ITU

Tabla 9.18: Regresión lineal (PU / ITU)

Según los resultados mostrados en la Tabla 9.18 se observa que el modelo de regresión fue encontrado no significativo, pues el valor p es de 0.108, siendo mayor a 0.1. También se observa un valor $R^2 = 0.373$ lo cual quiere decir que el 37.3% de la variabilidad en la intención de uso (ITU) es explicado por el modelo de regresión lineal. Por tanto, se puede afirmar que la intención de uso (ITU) no se ve influenciada por la utilidad percibida (PU).

- **Intención de Uso vs. Facilidad de Uso Percibida**

Se analizará si la intención de uso (ITU) es determinada por la facilidad de uso percibida (PEOU). Para esto se define un modelo de regresión lineal donde la variable predictora (independiente) es PEOU y la intención de uso (ITU) es la variable dependiente, el resultado se observa en la Tabla 9.19.

Modelo		Coeficientes no estandarizados B	Desv. Error	Coeficientes estandarizados Beta	t	Valor-p.	R2
1	(Constante)	1,455	1,402		1,038	,339	
	PEOU	,689	,355	,621	1,942	,100	,386

a. Variable dependiente: ITU

Tabla 9.19: Regresión lineal (PEOU / ITU)

Según los resultados mostrados en la Tabla 9.19 se observa que el modelo de regresión tiene una baja significancia, pues el valor p es de 0.1. También se observa un valor $R^2 = 0.386$ lo cual quiere decir que el 38.6% de la variabilidad en la intención de uso (ITU) es explicado por el modelo de regresión lineal. Por tanto, se puede afirmar que la intención de uso (ITU) no se ve influenciada por la facilidad de uso percibida (PEOU).

10. Análisis de los resultados:

En esta sección se resumen los resultados del experimento obtenidos en el punto anterior. Se ha evaluado cada una de las hipótesis planteadas en el punto 7.3 del presente estudio. En base a los resultados obtenidos se ha tenido los siguientes resultados para cada una de las hipótesis:

- **Para la hipótesis H_1:**

  En base a los resultados de la prueba de Wilcoxon se rechaza la hipótesis nula, por tanto, se puede afirmar que para la muestra estudiada existe una diferencia significativa en el tiempo utilizado para obtener los elementos de trazabilidad.

  Se pudo observar que, al utilizar la propuesta, en todos los casos se necesitó un menor tiempo para obtener los elementos de trazabilidad. Revisando los datos descriptivos, se observó que tanto la media y la mediana fue de 13 y 12.5 respectivamente al utilizar la propuesta, lo cual es mucho menor a 51 y 52.5 obtenidos al no utilizar la propuesta.

  Por lo tanto, se puede afirmar que, para la muestra, al utilizar la propuesta se incurre en un menor tiempo, con un valor mediana de 12.5 minutos.

- **Para la hipótesis H_2:**

  En base a los resultados de la prueba de Wilcoxon se rechaza la hipótesis nula, por tanto, se puede afirmar que para la muestra estudiada existe una diferencia significativa en el número de omisiones al obtener los elementos de trazabilidad.

  Con la prueba de rangos Wilcoxon se pudo observar que, al utilizar la propuesta, hubo seis casos donde se obtuvo un menor número de omisiones al obtener los elementos de trazabilidad, no hubo ningún caso donde el número de omisiones al obtener la propuesta sea mayor, y en dos casos el número de omisiones fue el mismo. Revisando los datos descriptivos, se observó que tanto la media y la mediana fue de 0.75 y 0 respectivamente al utilizar la propuesta, lo cual es menor a 4.25 y 4.5 obtenidos al no utilizar la propuesta.

  Por lo tanto, se puede afirmar que, para la muestra, al utilizar la propuesta se incurre en un menor número de omisiones, con un valor mediana de 0.

- **Para la hipótesis H_3:**

  En base a los resultados se rechaza la hipótesis nula, por tanto, se puede afirmar que existe diferencia en la percepción de la facilidad de uso al utilizar el método propuesto y el tradicional. Al utilizar el método propuesto se pudo observar que hay una mejor percepción en la facilidad de uso, donde se obtuvo un valor entre 3.607 y 4.26.

- **Para la hipótesis H_4:**

  En base a los resultados se rechaza la hipótesis nula, por tanto, se puede afirmar que existe diferencia en la intención de uso al utilizar el método propuesto y el tradicional. Al utilizar el método propuesto se pudo observar que hay una mejor percepción en la intención de uso, donde se obtuvo un valor entre 3.801 y 4.531.

- **Para la hipótesis H_5:**

  En base a los resultados se rechaza la hipótesis nula, por tanto, se puede afirmar que existe diferencia en la utilidad percibida al utilizar el método propuesto y el tradicional. Al utilizar el método propuesto se pudo observar que hay una mejor percepción en la utilidad, donde la mediana observada fue de 4.8, siendo mayor al valor neutro de 3.

Se analizó las relaciones causales del MEM entre eficacia actual y la eficacia percibida, la eficacia actual definida por la eficiencia y la efectividad actual. La eficiencia fue medida por la variable tiempo y la efectividad por el número de omisiones. Con respecto a la eficacia percibida, se utilizaron las variables de percepción (PEOU, PU, ITU).

Para determinar si existe relación de causalidad entre los entre los constructos definido en el modelo de evaluación de métodos (MEM), se utilizó el modelo de regresión lineal, y como resultado se obtuvo lo siguiente:

- La facilidad de uso percibida (PEOU) no se ve determinada por eficiencia actual (tiempo utilizado).
- La utilidad percibida (PU) no está determinada por la efectividad actual (número de omisiones).
- La intención de uso (ITU) no está determinada por PEOU o PU.
- PU no está determinada por PEOU.

11. Evaluación de validez:

En *(Wohlin et al., 2012, p. 102)* se indica que es importante considerar las cuestiones de validez en la fase de planeamiento para planear una adecuada validez del experimento. Una adecuada validez se refiere que los resultados deben ser válidos para la población de interés. En primer lugar, los resultados deben ser válidos para la población de la que se extrae la muestra. En segundo lugar, puede ser de interés generalizar los resultados a una población más amplia. Se definen 4 tipos de amenazas a la validez *(Cook & Campbell, 1979)*:

11.1. Validez del constructo:

Al momento de realizar el cuestionario para evaluar las variables de percepción, se les mencionó a los participantes que respondan de la forma más neutral posible, y que la propuesta no necesariamente tiene que dar mejores resultados a la forma en como se viene trabajando. Esto se hizo con el fin de mitigar, que los participantes de respuestas en favor de la propuesta presentada en el estudio.

Para poder evaluar la propuesta, se ha hecho uso del modelo de evaluación de métodos (MEM) el cual proporciona un mecanismo para evaluar tanto la eficacia actual y percibida, así como la intención de uso. Con esto se ha podido determinar que se debe evaluar para determinar si el método propuesto facilita la trazabilidad entre requerimientos funcionales y código fuente.

11.2. Validez interna:

Para evitar que el resultado del experimento se vea afectado por el cansancio de los participantes, se definió dos etapas, cada uno con un tiempo máximo de una hora. Para la ejecución del experimento, se utilizó un cuestionario de 14 preguntas, el cual fue evaluado tanto en su validez como en su confiabilidad, de esto se pudo ver que el cuestionario no era muy confiable para evaluar la intención de uso.

Con el fin de tener participantes más homogéneos, se realizó una invitación a un grupo de 20 participantes, con experiencia profesional similar, para que de forma voluntaria colaborasen en el experimento, de estos 9 aceptaron participar, donde uno no llegó a presentarse. Sin embargo, dado que el grupo es homogéneo, la no presencia de este participante, no afectaría los resultados del experimento.

11.3. Validez externa:

Será complicado poder generalizar los resultados fuera del contexto de la institución donde se ha realizado el experimento, ya que la muestra es pequeña. Sin embargo, se ha descrito las características del entorno, tales como experiencia del personal, herramientas, métodos con el fin de evaluar la aplicabilidad en un contexto específico.

11.4. Validez de las conclusiones:

Para evaluar las hipótesis planteadas en el presente estudio, se hizo uso de estadísticos de prueba, tanto paramétricos como no paramétricos, en base a si la distribución de los datos era normal o no.

12. Conclusiones y trabajos futuros:

12.1. Conclusiones

En el presente estudio se llevó a cabo una evaluación empírica para evaluar una propuesta que busca utilizar BDD como un mecanismo para garantizar la trazabilidad entre requerimientos funcionales y código fuente.

Antes de llevar a cabo el experimento, se realizó una revisión sistemática de la literatura (RSL), donde se plantearon 2 preguntas, la primera para determinar que estudios existen referentes a la obtención automática de trazabilidad entre requerimientos funcionales y código fuente. Se obtuvieron los estudios, donde más de la mitad utilizan algoritmos de recuperación de información ("*Information Retrieval*") como mecanismo para automatizar la obtención de información de trazabilidad.

En la segunda pregunta de la RSL, se buscó identificar que estudios utilizan BDD como mecanismo para obtener la información de trazabilidad. En este caso, se obtuvo una cantidad muy reducida de estudios, sólo 2, lo cual nos indica que BDD aún no ha sido muy explotado como mecanismo para garantizar la trazabilidad.

Una vez realizada esta RSL, se procedió con la evaluación empírica, para lo cual se diseñó un experimento, donde se buscaba comparar la forma tradicional y la propuesta para obtener la información de trazabilidad.

El experimento se hizo con una muestra muy pequeña, sólo 8 participantes, por lo que los resultados de estos no serían concluyentes, y haría falta realizar experimentos con muestras más grandes.

Se plantearon 5 hipótesis basadas en el modelo de evaluación de métodos (MEM), se pudo observar, para la muestra, que la propuesta permitía obtener la información de trazabilidad en un tiempo menor, así como también se incurrían en un número menor de omisiones. El tiempo menor indicaba que se tenía una mayor eficiencia con el método propuesto. El menor número de omisiones indicaba que se tenía una mayor efectividad al usar el método propuesto.

También se evaluaron las variables de percepción, las cuales se corresponden con las hipótesis $H_{3,}\ H_4\ y\ H_5$. En todas ellas se observó que la percepción en la facilidad, utilidad e intención de uso era mejor al utilizar la propuesta para garantizar la trazabilidad. Sin embargo, cuando se evaluaron las relaciones causales del MEM, no se pudo observar alguna dependencia entre ellas. Lo que quiere decir que no es posible explicar la variación en la percepción de intención de uso a partir de la variación en percepción de la facilidad de uso y utilidad percibida. Este resultado es probable que se haya obtenido por el pequeño número de participantes en el experimento.

Con respecto a los objetivos específicos planteados en la sección 3.1 del presente estudio, se pudo demostrar que se puede automatizar en cierta forma la obtención de trazabilidad entre requerimientos funcionales y el código fuente, esto se explicó detalladamente en la sección 6.2, ya que en BDD se definen los requerimientos y escenarios, y estos se encuentran asociados con pruebas ejecutables, al combinarlos con el uso de frameworks como Cucumber y Test Coverage, se pudo obtener los elementos de código fuente asociados a los requerimientos.

Las pruebas BDD son el insumo para poder obtener la trazabilidad entre requerimientos funcionales y código fuente, así mismo la definición de estas pruebas forman parte del código fuente de toda la aplicación, por tanto, el repositorio de código fuente sería el lugar donde se centralizaría la información de trazabilidad, cumpliendo de esta forma con el segundo objetivo específico del presente estudio. Cada vez que se necesite obtener los elementos de trazabilidad, se podría consultar el repositorio de código fuente, observar las pruebas BDD que se hayan definido, y a través de su ejecución obtener los elementos de trazabilidad.

El tercer objetivo específico planteado era el de facilitar el registro de la información de trazabilidad, para lo cual se plantearon las hipótesis de la sección 7.3, al evaluar las variables de le eficacia actual y la percibida del modelo de evaluación de métodos, se pudo comprobar que este objetivo se cumplía al utilizar la propuesta. Sin embargo, los resultados no se podrían generalizar, pues la muestra estudiada es muy pequeña, y sería necesario replicar más experimentos.

Con el cumplimiento de los objetivos específicos del presente estudio, es posible afirmar que el método propuesto puede facilitar la trazabilidad entre requerimientos funcionales y código fuente.

12.2. Trabajos futuros

El presente estudio se ha llevado a cabo en una muestra muy pequeña, por lo que es necesario aplicar el experimento en muestras mayores, con el fin poder seguir verificando la validez del estudio, así como tener una mayor información para analizar las relaciones causales en el modelo de evaluación de métodos. Esto último nos permitiría verificar si realmente existe una intención de uso en base a la utilidad o facilidad de uso percibida.

También sería necesario mejorar el cuestionario utilizado para evaluar las variables de percepción, en concreto las preguntas referentes a intención de uso, pues se observa que tienen un valor α de Cronbach de 0.469, siendo el recomendado un valor mínimo de 0.7, por lo que las preguntas actuales no resultan muy confiables para evaluar la intención de uso.

Para el tema de los casos prácticos, se podría realizar el experimento con repositorios públicos disponibles para la investigación referente a trazabilidad, un ejemplo de estos repositorios se encuentra http://coest.org/. En estos repositorios ya se tiene previamente mapeado la trazabilidad entre requerimientos funcionales y código fuente, esto resulta de utilidad al momento de querer realizar estudios con nuevos enfoques para obtener la información de trazabilidad. Estos repositorios podrían ayudar para poder aplicar el estudio en contextos más genéricos, y que los casos prácticos no necesariamente se encuentren vinculados al dominio de negocio de la entidad en la que se está aplicando el estudio.

La obtención de los elementos de trazabilidad al utilizar BDD han sido logrados haciendo uso de *plugins* en Eclipse, en donde se ha utilizado *Test Coverage* junto con los casos de prueba bajo el enfoque BDD, esto ha permitido obtener los elementos de trazabilidad, en un futuro se podría plantear el desarrollo de un *plugin* que genere automáticamente la matriz de trazabilidad.

13. Referencias bibliográficas:

Aizenbud-Reshef, N., Nolan, B. T., Rubin, J., & Shaham-Gafni, Y. (2010). Model traceability. IBM Systems Journal, 45(3), 515–526. https://doi.org/10.1147/sj.453.0515

Piattini, M. G., Bocco, M. F. G., & Lemus, J. A. C. (2014). Métodos de investigación en ingeniería del software. Editorial Ra-Ma.

Bourque, P., IEEE Computer Society, & Fairley, R. E. (2014). Guide to the Software Engineering Body of Knowledge Version 3. 0. USA: IEEE Computer Society Press.

Chemuturi, M. (2012). Requirements Engineering and Management for Software Development Projects. Springer New York.

Cleland-Huang, J., Chang, C. K., & Christensen, M. (2003). Event-based traceability for managing evolutionary change. IEEE Transactions on Software Engineering, 29(9), 796–810. https://doi.org/10.1109/TSE.2003.1232285

Cleland-Huang, J., Zemont, G., & Lukasik, W. (2004). A heterogeneous solution for improving the return on investment of requirements traceability. In Proceedings of the IEEE International Conference on Requirements Engineering (pp. 230–239). https://doi.org/10.1109/ICRE.2004.1335680

Gotel, O. C. Z., & Finkelstein, C. W. (2002). An analysis of the requirements traceability problem (pp. 94–101). Institute of Electrical and Electronics Engineers (IEEE).

Heindl, M., & Biffl, S. (2005). A case study on value-based requirements tracing (p. 60). Association for Computing Machinery (ACM). https://doi.org/10.1145/1081706.1081717

Kannenberg, A., & Saiedian, H. (2009). Why Software Requirements Traceability Remains a Challenge. The Journal of Defense Software Engineering, (July/August), 14–19.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in SE. EBSE Technical Report

Koelsch, G. (2016). Requirements Writing for System Engineering. Virginia, USA: Apress.

Moody, D. L., Sindre, G., Brasethvik, T., & Sølvberg, A. (2002). Evaluating the Quality of Process Models: Empirical Testing of a Quality Framework. Conceptual Modeling — ER 2002, 380–396. https://doi.org/10.1007/3-540-45816-6_36

Moody, D. L. (2001). A Practical Method for Representing Large Entity Relationship Models, P.h.D. Thesis. University of Melbourne, Australia.

North, D. (2006, September 20). Introducing BDD. Retrieved June 9, 2019, from https://dannorth.net/introducing-bdd/

Pressman, R. S., & Bruce R. Maxim, D. (2014). Software Engineering: A Practitioner's Approach (8th ed.). NY, USA: McGraw-Hill Education.

Rempel, P., & Mader, P. (2017). Preventing Defects: The Impact of Requirements Traceability Completeness on Software Quality. IEEE Transactions on Software Engineering, 43(8), 777–797. https://doi.org/10.1109/TSE.2016.2622264

Smart, J. F. (2014). BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle. NY: Manning Publications.
Sommerville, I. (2016). Software engineering (10th edition). Pearson Education Limited.

Standish Group. (1994). CHAOS report 1994. The Standish Group International, Inc. Retrieved from https://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf

Standish Group. (2015). CHAOS report 2015. The Standish Group International, Inc. Retrieved from https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). Experimentation in Software Engineering. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-29044-2

Ye, W. (2013). Instant Cucumber BDD How-to. Birmingham, UK: Packt Publishing.

Rodriguez, D. V., & Carver, D. L. (2019). Comparison of Information Retrieval Techniques for Traceability Link Recovery. 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT). https://doi.org/10.1109/INFOCT.2019.8710919

Ali, N., Cai, H., Hamou-Lhadj, A., & Hassine, J. (2019). Exploiting Parts-of-Speech for effective automated requirements traceability. Information and Software Technology, 106, 126–141. https://doi.org/10.1016/j.infsof.2018.09.009

Wang, W., Niu, N., Liu, H., & Niu, Z. (2018). Enhancing Automated Requirements Traceability by Resolving Polysemy. 2018 IEEE 26th International Requirements Engineering Conference (RE). https://doi.org/10.1109/RE.2018.00-53

Ghannem, A., Hamdi, M. S., Kessentini, M., & Ammar, H. H. (2017). Search-Based Requirements Traceability Recovery. Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016, 156–171. https://doi.org/10.1007/978-3-319-56994-9_11

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., Brinkkemper, S., & Zowghi, D. (2017). Behavior-Driven Requirements Traceability via Automated Acceptance Tests. 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). https://doi.org/10.1109/REW.2017.84

Ghannem, A., Hamdi, M. S., Kessentini, M., & Ammar, H. H. (2017). Search-based requirements traceability recovery: A multi-objective approach. 2017 IEEE Congress on Evolutionary Computation (CEC). https://doi.org/10.1109/CEC.2017.7969440

Zhang, Y., Wan, C., & Jin, B. (2016). An empirical study on recovering requirement-to-code links. 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). https://doi.org/10.1109/SNPD.2016.7515889

Mahmoud, A., & Niu, N. (2014). On the role of semantics in automated requirements tracing. Requirements Engineering, 20(3), 281–300. https://doi.org/10.1007/s00766-013-0199-y

Nunnally, J. C. (1978). Psychometric Theory (2nd ed.). Mcgraw-Hill College.

Ali, N., Sharafi, Z., Guéhéneuc, Y.-G., & Antoniol, G. (2014). An empirical study on the importance of source code entities for requirements traceability. Empirical Software Engineering, 20(2), 442–478. https://doi.org/10.1007/s10664-014-9315-y

Tsuchiya, R., Washizaki, H., Fukazawa, Y., Oshima, K., & Mibe, R. (2015). Interactive Recovery of Requirements Traceability Links Using User Feedback and Configuration Management Logs. Advanced Information Systems Engineering, 247–262. https://doi.org/10.1007/978-3-319-19069-3_16
Vinárek, J., Hnětynka, P., Šimko, V., & Kroha, P. (2014). Recovering Traceability Links Between Code and Specification Through Domain Model Extraction. Lecture Notes in Business Information Processing, 187–201. https://doi.org/10.1007/978-3-662-44860-1_11

Jyoti, & Chhabra, J. K. (2017). Requirements Traceability Through Information Retrieval Using Dynamic Integration of Structural and Co-change Coupling. Communications in Computer and Information Science, 107–118. https://doi.org/10.1007/978-981-10-5780-9_10

Zhou, J., Lu, Y., & Lundqvist, K. (2013). A Context-based Information Retrieval Technique for Recovering Use-Case-to-Source-Code Trace Links in Embedded Software Systems. 2013 39th Euromicro Conference on Software Engineering and Advanced Applications. https://doi.org/10.1109/SEAA.2013.30

Ali, N., Jaafar, F., & Hassan, A. E. (2013). Leveraging historical co-change information for requirements traceability. 2013 20th Working Conference on Reverse Engineering (WCRE). https://doi.org/10.1109/WCRE.2013.6671311

Delater, A., & Paech, B. (2013). Tracing Requirements and Source Code during Software Development: An Empirical Study. 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement. https://doi.org/10.1109/ESEM.2013.16

Ali, N., Gueheneuc, Y.-G., & Antoniol, G. (2013). Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links. IEEE Transactions on Software Engineering, 39(5), 725–741. https://doi.org/10.1109/TSE.2012.71

De Carvalho, R. A., de Carvalho e Silva, F. L., Manhães, R. S., & de Oliveira, G. L. (2013). Implementing Behavior Driven Development in an Open Source ERP. Lecture Notes in Business Information Processing, 242–249. https://doi.org/10.1007/978-3-642-36611-6_22

Niu, N., & Mahmoud, A. (2012). Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited. 2012 20th IEEE International Requirements Engineering Conference (RE). https://doi.org/10.1109/RE.2012.6345842

EGYED, A., & GRÜNBACHER, P. (2005). SUPPORTING SOFTWARE UNDERSTANDING WITH AUTOMATED REQUIREMENTS TRACEABILITY. International Journal of Software Engineering and Knowledge Engineering, 15(05), 783–810. https://doi.org/10.1142/S0218194005002464

Cleland-Huang, Jane, Berenbach, B., Clark, S., Settimi, R., & Romanova, E. (2007). Best Practices for Automated Traceability. Computer, 40(6), 27–35. https://doi.org/10.1109/MC.2007.195

Naslavsky, L., Alspaugh, T. A., Richardson, D. J., & Ziv, H. (2005). Using scenarios to support traceability. Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering - TEFSE '05. https://doi.org/10.1145/1107656.1107663

Ghabi, A., & Egyed, A. (2015). Exploiting traceability uncertainty among artifacts and code. Journal of Systems and Software, 108, 178–192. https://doi.org/10.1016/j.jss.2015.06.037

Omoronyia, I., Sindre, G., & Stålhane, T. (2011). Exploring a Bayesian and linear approach to requirements traceability. Information and Software Technology, 53(8), 851–871. https://doi.org/10.1016/j.infsof.2011.03.001

Zarour, M., Abran, A., Desharnais, J.-M., & Alarifi, A. (2015). An investigation into the best practices for the successful design and implementation of lightweight software process assessment methods: A systematic literature review. Journal of Systems and Software, 101, 180–192. https://doi.org/10.1016/j.jss.2014.11.041

Shapiro, S. S., & Wilk, M. B. (1965). An Analysis of Variance Test for Normality (Complete Samples). Biometrika, 52(3/4), 591. https://doi.org/10.2307/2333709

Student. (1908). The Probable Error of a Mean. Biometrika, 6(1), 1. https://doi.org/10.2307/2331554

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. Biometrics Bulletin, 1(6), 80. https://doi.org/10.2307/3001968

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. Psychometrika, 16(3), 297–334. https://doi.org/10.1007/bf02310555

Campbell, D. T., & Fiske, D. W. (1959). Convergent and discriminant validation by the multitrait-multimethod matrix. Psychological Bulletin, 56(2), 81–105. https://doi.org/10.1037/h0046016

Cook, T. D., & Campbell, D. T. (1979). Quasi-experimentation: Design & Analysis Issues for Field Settings. Boston: Houghton Mifflin.

14. Anexos:

Anexo 1

Lista de estudios obtenidos en la revisión sistemática de la literatura

ID	Referencia bibliográfica
S1	Rodriguez, D. V., & Carver, D. L. (2019). Comparison of Information Retrieval Techniques for Traceability Link Recovery. 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT). https://doi.org/10.1109/INFOCT.2019.8710919
S2	Ali, N., Cai, H., Hamou-Lhadj, A., & Hassine, J. (2019). Exploiting Parts-of-Speech for effective automated requirements traceability. Information and Software Technology, 106, 126–141. https://doi.org/10.1016/j.infsof.2018.09.009
S3	Wang, W., Niu, N., Liu, H., & Niu, Z. (2018). Enhancing Automated Requirements Traceability by Resolving Polysemy. 2018 IEEE 26th International Requirements Engineering Conference (RE). https://doi.org/10.1109/RE.2018.00-53
S4	Ghannem, A., Hamdi, M. S., Kessentini, M., & Ammar, H. H. (2017). Search-Based Requirements Traceability Recovery. Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016, 156–171. https://doi.org/10.1007/978-3-319-56994-9_11
S5	Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., Brinkkemper, S., & Zowghi, D. (2017). Behavior-Driven Requirements Traceability via Automated Acceptance Tests. 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). https://doi.org/10.1109/REW.2017.84
S6	Ghannem, A., Hamdi, M. S., Kessentini, M., & Ammar, H. H. (2017). Search-based requirements traceability recovery: A multi-objective approach. 2017 IEEE Congress on Evolutionary Computation (CEC). https://doi.org/10.1109/CEC.2017.7969440
S7	Zhang, Y., Wan, C., & Jin, B. (2016). An empirical study on recovering requirement-to-code links. 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). https://doi.org/10.1109/SNPD.2016.7515889
S8	Mahmoud, A., & Niu, N. (2014). On the role of semantics in automated requirements tracing. Requirements Engineering, 20(3), 281–300. https://doi.org/10.1007/s00766-013-0199-y
S9	Ali, N., Sharafi, Z., Guéhéneuc, Y.-G., & Antoniol, G. (2014). An empirical study on the importance of source code entities for requirements traceability. Empirical Software Engineering, 20(2), 442–478. https://doi.org/10.1007/s10664-014-9315-y
S10	Tsuchiya, R., Washizaki, H., Fukazawa, Y., Oshima, K., & Mibe, R. (2015). Interactive Recovery of Requirements Traceability Links Using User Feedback and Configuration Management Logs. Advanced Information Systems Engineering, 247–262. https://doi.org/10.1007/978-3-319-19069-3_16
S11	Vinárek, J., Hnětynka, P., Šimko, V., & Kroha, P. (2014). Recovering Traceability Links Between Code and Specification Through Domain Model Extraction. Lecture Notes in Business Information Processing, 187–201. https://doi.org/10.1007/978-3-662-44860-1_11
S12	Jyoti, & Chhabra, J. K. (2017). Requirements Traceability Through Information Retrieval Using Dynamic Integration of Structural and Co-change Coupling. Communications in Computer and Information Science, 107–118. https://doi.org/10.1007/978-981-10-5780-9_10
S13	Zhou, J., Lu, Y., & Lundqvist, K. (2013). A Context-based Information Retrieval Technique for Recovering Use-Case-to-Source-Code Trace Links in Embedded Software Systems. 2013 39th Euromicro Conference on Software Engineering and Advanced Applications. https://doi.org/10.1109/SEAA.2013.30
S14	Ali, N., Jaafar, F., & Hassan, A. E. (2013). Leveraging historical co-change information for requirements traceability. 2013 20th Working Conference on Reverse Engineering (WCRE). https://doi.org/10.1109/WCRE.2013.6671311
S15	Delater, A., & Paech, B. (2013). Tracing Requirements and Source Code during Software Development: An Empirical Study. 2013 ACM / IEEE International

	Symposium on Empirical Software Engineering and Measurement. https://doi.org/10.1109/ESEM.2013.16
S16	Ali, N., Gueheneuc, Y.-G., & Antoniol, G. (2013). Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links. IEEE Transactions on Software Engineering, 39(5), 725–741. https://doi.org/10.1109/TSE.2012.71
S17	De Carvalho, R. A., de Carvalho e Silva, F. L., Manhães, R. S., & de Oliveira, G. L. (2013). Implementing Behavior Driven Development in an Open Source ERP. Lecture Notes in Business Information Processing, 242–249. https://doi.org/10.1007/978-3-642-36611-6_22
S18	Niu, N., & Mahmoud, A. (2012). Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited. 2012 20th IEEE International Requirements Engineering Conference (RE). https://doi.org/10.1109/RE.2012.6345842
S19	EGYED, A., & GRÜNBACHER, P. (2005). SUPPORTING SOFTWARE UNDERSTANDING WITH AUTOMATED REQUIREMENTS TRACEABILITY. International Journal of Software Engineering and Knowledge Engineering, 15(05), 783–810. https://doi.org/10.1142/S0218194005002464
S20	Cleland-Huang, Jane, Berenbach, B., Clark, S., Settimi, R., & Romanova, E. (2007). Best Practices for Automated Traceability. Computer, 40(6), 27–35. https://doi.org/10.1109/MC.2007.195
S21	Naslavsky, L., Alspaugh, T. A., Richardson, D. J., & Ziv, H. (2005). Using scenarios to support traceability. Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering - TEFSE '05. https://doi.org/10.1145/1107656.1107663
S22	Ghabi, A., & Egyed, A. (2015). Exploiting traceability uncertainty among artifacts and code. Journal of Systems and Software, 108, 178–192. https://doi.org/10.1016/j.jss.2015.06.037
S23	Omoronyia, I., Sindre, G., & Stålhane, T. (2011). Exploring a Bayesian and linear approach to requirements traceability. Information and Software Technology, 53(8), 851–871. https://doi.org/10.1016/j.infsof.2011.03.001
S24	Florez, J. M. (2019). Automated Fine-Grained Requirements-to-Code Traceability Link Recovery. 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). Published. https://doi.org/10.1109/icse-companion.2019.00087
S25	Wang, S., Li, T., & Yang, Z. (2019). Exploring Semantics of Software Artifacts to Improve Requirements Traceability Recovery: A Hybrid Approach. 2019 26th Asia-Pacific Software Engineering Conference (APSEC). Published. https://doi.org/10.1109/apsec48747.2019.00015
S26	Rodriguez, D. V., & Carver, D. L. (2020). Multi-Objective Information Retrieval-Based NSGA-II Optimization for Requirements Traceability Recovery. 2020 IEEE International Conference on Electro Information Technology (EIT). Published. https://doi.org/10.1109/eit48999.2020.9208233
S27	Li, T., Wang, S., Lillis, D., & Yang, Z. (2020). Combining Machine Learning and Logical Reasoning to Improve Requirements Traceability Recovery. Applied Sciences, 10(20), 7253. https://doi.org/10.3390/app10207253
S28	Rodriguez, D. V., & Carver, D. L. (2020). An IR-Based Artificial Bee Colony Approach for Traceability Link Recovery. 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI). Published. https://doi.org/10.1109/ictai50040.2020.00174
S29	Wang, H., Shen, G., Huang, Z., Yu, Y., & Chen, K. (2021). Analyzing close relations between target artifacts for improving IR-based requirement traceability recovery. Frontiers of Information Technology & Electronic Engineering, 22(7), 957–968. https://doi.org/10.1631/fitee.2000126

Anexo 2

Hoja Informativa

Título: Propuesta de solución para garantizar la trazabilidad de requerimientos funcionales usando desarrollo guiado por comportamiento en una entidad del gobierno

Investigadores: Roger Armando Contreras Corrales

Destinatario: El participante

Este proyecto de investigación es llevado a cabo por Roger Armando Contreras Corrales, estudiante de la Maestría en Informática con mención en Ingeniería de Software. Esta investigación es parte de la tesis para obtener la Maestría en Informática otorgada por la Escuela de Postgrado de la Pontificia Universidad Católica del Perú.

El propósito de esta investigación es evaluar si el uso del desarrollo guiado por comportamiento facilitaría la obtención de información de trazabilidad entre requerimientos funcionales y código fuente.

La investigación requiere que el participante desarrolle dos casos prácticos, cada uno con una duración máxima de una hora, al finalizar se le pedirá completar un cuestionario.

1. Primer caso práctico (1 hora): El participante recibirá una lista de requerimientos funcionales, para cada uno de estos se debe identificar las clases y métodos involucrados, así mismo se debe colocar la hora en que inició y finalizó por cada requerimiento funcional trabajado.

   En este caso práctico, para identificar las clases y métodos, el participante utilizará su forma habitual de trabajo, que se basa en la revisión documental y exploración en el código fuente.

2. Segundo caso práctico (1 hora): La información a completar es la misma que en el primer caso práctico, pero con nuevos requerimientos funcionales. Para identificar las clases y métodos, el participante utilizará la forma de trabajo propuesta, que se basa en el uso de desarrollo guiado por comportamiento (BDD).

3. Cuestionario (5 minutos): Cuestionario que presentará preguntas para comparar la forma actual de trabajo y la propuesta basada en desarrollo guiado por comportamiento (BDD)

Los datos personales del participante, como los nombres y número de DNI, no serán incluidos dentro de los datos de la investigación. Toda la información que provea será anónima, los datos personales sólo serán accesibles para los miembros del proyecto y serán destruidas luego de tres años.

La participación es completamente voluntaria, y puede retirar su participación en cualquier momento; sin embargo, no podrá retirar la información ya entregada.

Para participar, por favor, llenar el formulario de consentimiento.

Si tuviera alguna consulta sobre la investigación, los datos de contacto de los investigadores son los siguientes:

Roger Armando Contreras Corrales (Estudiante de Maestría, Escuela de Posgrado)
Correo: roger.contreras@pucp.edu.pe

Anexo 3

Formulario de consentimiento

Yo, _____,
con DNI _____ doy mi consentimiento para participar de manera voluntaria en el estudio. He recibido información en forma verbal sobre el estudio mencionado anteriormente y he leído la información de la hoja informativa.

Al firmar este consentimiento, confirmo lo siguiente:

- Yo comprendo que la información de esta hoja será almacenada de manera independiente al cuestionario.
- Yo acepto participar en esta investigación.
- Yo comprendo que mi participación en esta investigación es voluntaria.
- Yo comprendo que puedo retirarme de la investigación en cualquier momento.
- Yo comprendo que mis datos personales no serán utilizados en la presente investigación.
- Yo comprendo que la información recabada, a través de los ejercicios y los cuestionarios, serán usados conforme lo detallado en la hoja informativa de la investigación en la que estoy participando.
- Yo comprendo que los datos que he entregado serán utilizados como parte de la investigación de la tesis de maestría del alumno (Roger Armando Contreras Corrales).

Si hubiese alguna consulta sobre la investigación, los datos de contacto del investigador son:

Roger Armando Contreras Corrales (Estudiante de Maestría, Escuela de Posgrado)
Correo: roger.contreras@pucp.edu.pe

_____	_____
Participante:	**Investigador:**
Fecha:	**Fecha:**

Anexo 4

Cuestionario de autoevaluación

Gracias por participar en el presente cuestionario, por favor responder cada una de las preguntas con la mayor sinceridad.

1. ¿Cuántos años de experiencia profesional tiene?

    a) Menor a 5 años
    b) Entre 5 y 10 años
    c) Mas de 10 años

2. ¿Cuántos años de experiencia profesional tiene desarrollando con Java?

    a) Menor a 5 años
    b) Entre 5 y 10 años
    c) Mas de 10 años

3. En que etapas ha estado involucrado en los últimos 3 años:

    a) Modelado del negocio.
    b) Requisitos
    c) Análisis y diseño
    d) Implementación
    e) Pruebas
    f) Despliegue

4. ¿Conoce acerca de desarrollo guiado por pruebas (TDD)?

    a) Si, lo he implementado en algunos proyectos profesionales.
    b) Si, pero sólo de manera teórica.
    c) He escuchado algo de eso.
    d) No.

5. ¿Conoce acerca de desarrollo guiado por comportamiento (BDD)?

    a) Si, lo he implementado en algunos proyectos profesionales.
    b) Si, pero sólo de manera teórica.
    c) He escuchado algo de eso.
    d) No.

6. Marque aquellas herramientas con los cuales usted alguna vez ha trabajado:

    a) TestNG
    b) JBehave
    c) Selenium WebDriver
    d) JUnit
    e) Mockito
    f) SoapUI
    g) EclEmma
    h) SonarQube

7. Si le pidiesen obtener la información de trazabilidad entre requerimientos funcionales y alguno de los artefactos de la lista, ¿Cuáles considera serían más fáciles de obtener? Seleccione sólo dos:

    a) Especificaciones de casos de uso.
    b) Clases Java.
    c) Métodos de las clases implementadas.
    d) Componentes (librerías y aplicaciones EAR).

8. Si le pidiesen obtener la información de trazabilidad entre requerimientos funcionales y alguno de los artefactos de la lista, ¿Cuáles considera serían más complicados de obtener? Seleccione sólo dos:

    a) Especificaciones de casos de uso.
    b) Clases Java.
    c) Métodos de las clases implementadas.
    d) Componentes (librerías y aplicaciones EAR).

Anexo 5

Primer caso práctico

Se pide identificar las clases y métodos involucrados en la implementación de los siguientes requerimientos:

Cod. Req.	Sistema	Sub-Sistema	Módulo	Caso de uso	Descripción requerimiento
REQ-19	34 - FAST Salida	3402 - Control de Ingreso y Salida de Mercancías	340202 - Control de Mercancía para Regímenes de Salida	CUS03 - Validar y Rectificar Recepción de mercancía.	El depósito temporal que rectifica debe ser el mismo que numeró la RM. De no cumplir registra el mensaje de error: “Depósito temporal que rectifica la RM debe ser el mismo que numera <ruc_deposito_numera>”. (RN5609) Si no se ha superado las validaciones se detiene el proceso.
REQ-16	30 - SDA DESPACHO	3007 - Rectificación de la Declaración	300702 - Rectificación de Declaración de Regímenes de Salida	CUS07-Atender solicitud de rectificación de DAM	El sistema muestra en una ventana con la siguiente información: - Sección: Sección de la DAM donde se realiza la rectificación. - Acción: Con los valores: Modificación, Eliminación, Adición. - Dato: Nombre del dato que se está modificando. Si se está eliminado o agregando, este dato se muestra en blanco. - Valor anterior: Valor anterior del dato que se está modificando. Si se está eliminado o agregando, este campo se muestra en blanco. - Nuevo valor: Nuevo valor del dato que se está modificando. Si se está eliminado o agregando, este campo se muestra en blanco.
REQ-06	30 - SDA DESPACHO	3007 - Rectificación de la Declaración	300702 - Rectificación de Declaración de Regímenes de Salida	CUS02- Rectificar DAM de exportación definitiva por SEIDA.	En caso de mercancías restringidas tipo de documento 21, se comunica a VUCE los datos de la DAM rectificada.
REQ-08	30 - SDA DESPACHO	3007 - Rectificación de la Declaración	300702 - Rectificación de Declaración de Regímenes de Salida	CUS02- Rectificar DAM de exportación definitiva por SEIDA.	Se envía un aviso al buzón SOL del exportador y despachador de aduana/OEA indicando que la DAM ha sido rectificada correctamente.
REQ-09	30 - SDA DESPACHO	3007 - Rectificación de la Declaración	300702 - Rectificación de Declaración de Regímenes de Salida	CUS04-Validar rectificación DAM exportación definitiva	El sistema valida que se haya enviado el número de la DAM a rectificar indicando la aduana, año, régimen y número, verificando que exista y no se encuentre legajada. De no cumplir, genera el mensaje de error: “DAM no existe” o “DAM se encuentra legajada”

Anexo 6

Segundo caso práctico

Se pide identificar las clases y métodos involucrados en la implementación de los siguientes requerimientos:

Cod. Req.	Sistema	Sub-Sistema	Modulo	Caso de uso	Descripción requerimiento
REQ-02	30 - SDA DESPACHO	3003 - Numeración de la Declaración	300302 - Regímenes de salida	CUS03-Validar reglas generales de exportación definitiva.	Si el código de país de destino es diferente de "Zonas francas del Perú", la DAM debe contar con dirección de consignatario de al menos cinco caracteres diferentes de espacios. De no cumplir, muestra el mensaje de error: “Se debe indicar la dirección del consignatario de la DAM (mínimo 5 caracteres)”. (RN5016).
REQ-03	30 - SDA DESPACHO	3003 - Numeración de la Declaración	300302 - Regímenes de salida	CUS03-Validar reglas generales de exportación definitiva.	El código del país de destino debe existir en el catálogo de países y ser diferente de Perú (código PE). De no cumplir, se muestra el mensaje: “País de destino de la DAM no válido”. (RN5023). (Esta verificación no aplica para rectificación web ya que el sistema mostrará únicamente los elementos válidos)
REQ-13	30 - SDA DESPACHO	3007 - Rectificación de la Declaración	300702 - Rectificación de Declaración de Regímenes de Salida	CUS05-Buscar DAM para rectificar WEB	2.5 El sistema valida que no exista una solicitud con evaluación previa pendiente (estados: 02-Por asignar, 03-Asignado con especialista o 04-En proceso de evaluar”) asociada a la DAM de los tipos: “Rectificación de DAM”, “Anulación de DAM” o “Reversión de anulación de DAM”. Caso contrario muestra el mensaje de error: “DAM cuenta con solicitud de evaluación previa pendiente”. (RN5265).
REQ-18	30 - SDA DESPACHO	3007 - Rectificación de la Declaración	300702 - Rectificación de Declaración de Regímenes de Salida	CUS08-Generar solicitud de BQ	1.3.2 Registra una solicitud con los siguientes datos: - Tipo de solicitud: Solicitud de BQ. - Número de solicitud: Se genera por aduana de la DAM, año y tipo de solicitud - Fecha de solicitud. Fecha del sistema. - Estado de solicitud. Pendiente de asignación.
REQ-20	34 - FAST Salida	3402 - Control de Ingreso y Salida de Mercancías	340202 - Control de Mercancía para Regímenes de Salida	CUS11 Registrar RIP/RIA mediante WebService	La aduana, año y número de la RCE debe existir con estado: Para el RIP con “03-Salida Autorizada del recinto con destino al puerto / puesto de control”, de no cumplir, registra el mensaje de error: “RCE <Aduana de Presentación-AñodeRCE-NúmerodeRCE> no existe” o “RCE no se encuentra en estado 03-Salida Autorizada del recinto con destino al puerto / puesto de control” ” y se detiene el procesamiento. (RN5505, RN5524) Para el RIA con estado ‘05’. De no cumplir, registra el mensaje de error: RCE <Aduana de Presentación-AñodeRCE-NúmerodeRCE> no existe o “RCE no se encuentra en estado de 05-Embarque Autorizado” y se detiene el procesamiento.

Anexo 7

Cuestionario para medir variables de percepción

Para cada una de las siguientes declaraciones, se pide marcar una alternativa en los casilleros correspondientes a los números entre 1 y 5, donde cada número significa lo siguiente:

1 – Totalmente en desacuerdo.
2 – En desacuerdo.
3 – Ni de acuerdo, ni desacuerdo.
4 – De acuerdo.
5 – Totalmente de acuerdo.

La información de trazabilidad se refiere a la información en donde, a partir de los requerimientos funcionales, se conoce cuales son las clases y métodos que lo implementan.

ID	Declaración	1	2	3	4	5
P1	Con el método propuesto, se ha requerido menos esfuerzo (tiempo) para obtener la información de trazabilidad.					
P2	En general, considero que el método propuesto para obtener la información de trazabilidad es útil.					
P3	En el trabajo, se debería utilizar el método propuesto para obtener la información de trazabilidad.					
P4	Es fácil de entender cómo utilizar el método propuesto para obtener la información de trazabilidad.					
P5	Considero que la información de trazabilidad, obtenida con el método propuesto, es fiable (puedo confiar en los resultados).					
P6	El método propuesto me ayudará para realizar el mantenimiento futuro de requerimientos funcionales.					
P7	Es fácil de entender como definir las características (según BDD) según el método propuesto.					
P8	Recomendaría el uso del método propuesto para obtener la información de trazabilidad.					
P9	La información de trazabilidad, obtenida con el método propuesto, me será de ayuda para identificar el impacto por cambios en el código fuente.					
P10	Es fácil de entender como especificar un caso de prueba según las prácticas de BDD.					
P11	El uso del método propuesto me ayudará a mantener actualizada la información de trazabilidad.					
P12	Considero que sería fácil adquirir las destrezas (capacidades/habilidades) para poder implementar los casos de prueba, según el método propuesto.					
P13	De ser necesario, utilizaría en el futuro el método propuesto.					
P14	Considero que el método propuesto me ayudará a reducir el tiempo necesario para obtener la información de trazabilidad.					

Anexo 8

Registro de información de trazabilidad

Para cada uno de los requerimientos, completar la información de trazabilidad hacia el código fuente. Se debe completar lo siguiente:

Cod. Req.: Es el código del requerimiento funcional del cual se desea obtener la información de trazabilidad.

Hora de inicio: Hora en formato (HH24:MI) al momento de iniciar el registro de información de trazabilidad.

Hora de fin: Hora en formato (HH24:MI) en que finaliza de registrar la información de trazabilidad.

Paquete: Nombre del paquete donde se encuentra la clase del código fuente.

Clase: Nombre de la clase vinculada al requerimiento funcional.

Método: Nombre del método de la clase que implementa la funcionalidad descrita por el requerimiento.

Cod. Req.			
Hora de inicio			
Hora de fin			
ID	**Paquete**	**Clase**	**Método**

Anexo 9

Respuestas al cuestionario de percepción

Participante	Preguntas													
	P1	P4	P7	P10	P12	P14	P2	P5	P6	P9	P11	P3	P8	P13
1	5	4	3	3	4	5	5	5	5	5	5	4	5	4
2	5	5	4	4	5	5	4	5	4	4	3	4	5	4
3	5	4	4	4	4	5	5	4	5	5	4	4	4	4
4	5	4	4	3	5	5	5	3	5	4	5	3	4	5
5	5	4	3	2	4	4	4	5	4	4	4	3	4	3
6	4	4	4	4	4	4	4	5	4	4	4	4	4	4
7	5	4	4	4	5	5	5	5	5	5	5	4	5	5
8	5	4	4	4	4	5	5	4	5	5	5	5	4	5