UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

Evaluación Empírica y Comparativa de Herramientas Usadas para la Validación de Requisitos Funcionales a Nivel de Modelos Conceptuales Basados en UML

Trabajo de titulación previo a la obtención del título de Ingeniero en Sistemas

Autor:

Carlos Darwin Vera Mejía

CI:1400896435

Correo electrónico: carlosvinf@hotmail.com

Directora:

Ing. María Fernanda Granda Juca PHD

CI:0702952441

Cuenca, Ecuador

17 de Julio de 2020

Resumen:

La validación de modelos conceptuales es una actividad clave para asegurar la calidad de software y ahorrar costos, especialmente cuando se adopta cualquier tipo de metodología de Ingeniería de Software Dirigido por Modelos. En este contexto, los lenguajes de modelado estándar, como el Lenguaje de Modelado Unificado -UML, y las herramientas para apoyar las actividades de validación se vuelven esenciales. El objetivo de este estudio es analizar qué tan buenas son las herramientas de modelado para apoyar la tarea de validación basada en pruebas. Nuestro objetivo es extraer las principales características de las herramientas que soportan este tipo de validación, de manera que en este estudio se comparen herramientas que ayuden a mejorar el proceso de detección temprana de defectos en el modelado conceptual. Nuestro estudio emplea dos enfoques de investigación: (1) el principal es una evaluación empírica que analiza eficacia, eficiencia y satisfacción de usuario de dos herramientas seleccionadas que permiten validar requisitos en modelos conceptuales UML y (2) un análisis teórico que complementa el experimento anterior. El estudio se enfoca en el tipo más frecuente de diagrama UML, el diagrama de clase y en dos herramientas ampliamente utilizadas por la comunidad de modelado para realizar una validación basada en pruebas: USE y OCLE.

Palabras claves: Diagrama de Clase. Modelo conceptual. Validación. Verificación.

Abstract:

Conceptual model validation is a key activity to ensure software quality and save costs, especially when adopting any type of Model Driven Software Engineering methodology. In this context, standard modeling languages, such as the Unified Modeling Language -UML, and tools to support validation activities become essential. The objective of this study is to analyze how good the modeling tools are to support the evidence-based validation task. Our objective is to extract the main characteristics of the tools that support this type of validation, so that this study compares tools that help improve the process of early detection of defects in conceptual modeling. Our study uses two research approaches: (1) the main one is an empirical evaluation that analyzes effectiveness, efficiency and user satisfaction of two selected tools that allow validating requirements in UML conceptual models and (2) a theoretical analysis that complements the previous experiment. The study focuses on the most common type of UML diagram, the class diagram, and two tools widely used by the modeling community to perform evidence-based validation: USE and OCLE.

Keywords: Class Diagram. Conceptual model. Validation. Verification.

Índice

Índice de Tablas

Índice de Figuras

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo Carlos Darwin Vera Mejía en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación *"EVALUACIÓN EMPÍRICA Y COMPARATIVA DE HERRAMIENTAS USADAS PARA LA VALIDACIÓN DE MODELOS CONCEPTUALES BASADOS EN UML"*, de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 17 de Julio de 2020

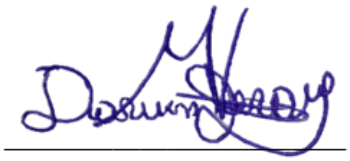
Carlos Darwin Vera Mejía

C.I: 1400896435

Cláusula de Propiedad Intelectual

Yo Carlos Darwin Vera Mejía, autor del trabajo de titulación "EVALUACIÓN EMPÍRICA Y COMPARATIVA DE HERRAMIENTAS USADAS PARA LA VALIDACIÓN DE MODELOS CONCEPTUALES BASADOS EN UML", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 17 de julio de 2020



Carlos Darwin Vera Mejía

C.I: 1400896435

Cláusula de Dirección

CERTIFICO:

Que el presente proyecto de Trabajo de Titulación: " EVALUACIÓN EMPÍRICA Y COMPARATIVA DE HERRAMIENTAS USADAS PARA LA VALIDACIÓN DE MODELOS CONCEPTUALES BASADOS EN UML ", fue dirigido por mi persona.

.....

Ing. María Fernanda Granda Juca, PhD

C.I. 0702952441

Directora de Tesis

DEDICATORIA

Este trabajo dedico a mi familia.

A mis padres Narcisa y Carlos quienes me han apoyado en todo momento a pesar de todas las dificultades que se pudieron presentar. A mis abuelos quienes estuvieron pendientes de este logro. A mi tía Margarita quien con sus consejos, preocupación y apoyo me ha alentado a seguir adelante.

AGRADECIMIENTO

Quiero agradecer a mis padres Narcisa y Carlos quienes siempre estuvieron presentes en mis logros con su apoyo constante, por haber confiado en mí y haberse esforzado para darme lo necesario y poder continuar con mis metas. Agradezco a mi directora la ingeniera María Fernanda Granda por apoyarme y ayudarme en base a sus conocimientos profesionales a solucionar problemas que se pudieron presentar en el desarrollo de este trabajo. Agradezco también al ingeniero Otto Parra por haber aportado significativamente a la realización de este trabajo. Agradezco a los docentes quienes trasmitieron sus amplios conocimientos y compartieron experiencias profesionales, las cuales me ayudaron a tener una visión más amplia en el ámbito laboral relacionado a la carrera.

CAPÍTULO 1: INTRODUCCIÓN

1.1 Antecedentes

Dentro de las etapas del ciclo de vida de desarrollo de software se encuentran las pruebas de software [1]. Estas son muy importantes ya que permiten encontrar defectos en un producto de software. Los defectos tienen un gran impacto en el presupuesto del software, por lo que es muy importante tratar de detectarlos y reducirlos en las etapas tempranas del proceso de desarrollo del software, por ejemplo, a nivel de modelo conceptual especialmente cuando se sigue cualquier tipo de proceso de Ingeniería de Software Dirigido por Modelo (MDSE, por sus siglas en inglés de Model-Driven Software Engineering).

Un modelo conceptual (MC) explica cuáles son y cómo se relacionan los conceptos relevantes en la descripción de un problema [2]. Existen muchas variantes para describir un modelo conceptual [1], como por ejemplo: diagramas entidad-relación, diagramas de clases basados en el lenguaje de Modelamiento Unificado (UML), diagramas de procesos basados en Business Process Model and Notation (BPMN) [3], etc. Por el lado del lenguaje, UML es el estándar de facto para el modelado de sistemas de software[4]. UML[4] proporciona varios diagramas para modelar la estructura de un sistema de software, su arquitectura y su comportamiento, siendo el diagrama de clases el más utilizado [5].

Una gran cantidad de herramientas comerciales y de código abierto están disponibles para admitir el modelado UML, p. ej., MagicDraw, Papyrus, ArgoUML, Modelio, StarUML, USE, OCLE entre muchas otras[1]. Si bien las características de cada herramienta difieren, todas ofrecen un editor gráfico para asistir en la definición de modelos UML y facilidades para la verificación del modelo creado. Sin embargo, nosotros creemos que hace falta una comparación teórica y empírica que ayude a seleccionar una herramienta que permita validar un MC basado en UML. Particularmente, este trabajo se centra en herramientas de

[1] https://modeling-languages.com/uml-tools/

modelado con licencia libre que permiten automatizar la ejecución de scripts de prueba.

Según Meyer: "Probar un programa (en nuestro caso un modelo conceptual) es intentar hacerlo fallar, y el objetivo de las pruebas es descubrir fallas inyectando defectos en el artefacto del software "[6], [7].

Un ejemplo claro de estos tipos de pruebas sería instanciar un objeto a nivel de modelo y probar que cumplan las restricciones (i.e. precondiciones, postcondiciones, invariantes) que se esperan a medida que se va cambiando de estado los objetos, durante la ejecución de un caso de prueba. Puede darse el caso de que algún elemento del modelo falte, nunca se utilice o se lo utiliza de forma redundante o incorrecta. En este caso, como hay otros fallos que se pretenden analizar, es necesario el uso de una herramienta que permita ejecutar casos de prueba con el propósito de validar el modelo conceptual. Esta validación es muy útil durante las fases de análisis y diseño del software, donde se requiere determinar en una fase temprana si el MC cumple con los requisitos especificados por el usuario [1]. Sin embargo, hace falta una evaluación empírica y una comparación teórica de herramientas usadas para la validación de requisitos funcionales a nivel de modelos conceptuales basados en diagramas de clases UML.

1.2 Motivación

Las pruebas dentro de la etapa del ciclo de vida del software tienen como fin reforzar el desarrollo de productos de software que cumplan con las expectativas de las partes interesadas. Actualmente, la complejidad de los sistemas de software es cada vez mayor y la capacidad de identificar en su mayoría los defectos desde el principio a nivel de modelo se convierte en un desafío, un correcto análisis podría ayudar mejorar la calidad del software y con ello obtener esquemas conceptuales completos y correctos. Para ello es necesario el uso de herramientas que verifiquen y validen un MC. La validación y verificación están relacionados con los conceptos del aseguramiento y la calidad del software [8]. Varias propuestas para validar un MC usando herramientas se presentan en la

literatura. Estas herramientas se basan en pruebas para esquemas conceptuales desde múltiples vistas.

Por lo mencionado anteriormente este trabajo se enfoca en realizar una evaluación empírica y comparación teórica de herramientas usadas para la validación de requisitos funcionales a nivel de modelos conceptuales basados en UML.

1.3 Planteamiento del Problema

Los errores en el diseño de MC son los más comunes en los proyectos de desarrollo de software, por lo que es necesario realizar una validación a nivel de MC. Dentro de la validación de esquemas UML intervienen las restricciones basadas en las reglas del negocio del sistema modelado. Todos los modelos UML deben ser planteados según los requerimientos pedidos por el cliente, lo cual indica que dichos modelos deben tener ciertas restricciones o reglas que el cliente impone para que el producto sea de su total agrado y satisfacción. Estas condiciones o reglas se las puede plantear en un MC mediante la inserción de restricciones, las cuales deben estar definidas correctamente. En la literatura existen lenguajes que son especializados para incorporar restricciones en un modelo UML, tal es el caso del lenguaje OCL [9], el cual se plantea utilizar en este trabajo.

Nuestro estudio agrega una nueva perspectiva a estos trabajos anteriores al realizar una evaluación empírica y teórica de herramientas que permitan detectar defectos a través de la ejecución de scripts de prueba sobre MC usando UML. Para este propósito, este trabajo se centra en el diagrama de UML más utilizado, el diagrama de clase [5] y en herramientas de modelado que permiten automatizar la ejecución de scripts de prueba.

El punto de partida de este trabajo comienza con la exposición del problema de investigación "Comparar teórica y empíricamente herramientas con licencia libre que validen MCs basados en diagramas de clase".

1.4 Objetivos

1.4.1 Objetivo General

Realizar una evaluación empírica y comparación teórica de herramientas usadas para la validación de requisitos funcionales a nivel de modelos conceptuales basados en UML.

1.4.2 Objetivos específicos

1. Elaborar el marco teórico necesario para la realización del trabajo de titulación.
2. Realizar el estado arte acerca de las herramientas usadas para la validación de requisitos funcionales a nivel de modelos conceptuales basados en UML.
3. Identificar dos herramientas adecuadas para la validación de modelos conceptuales basados en UML, evaluarlas, y compararlas.
4. Evaluar empíricamente las herramientas seleccionadas en cuanto a su eficacia, eficiencia y satisfacción de usuario.

1.5 Estructura del Documento

Con el fin de describir claramente el trabajo que se ha realizado, el documento se ha organizado de la siguiente manera:

Capítulo 2. Metodología

En este capítulo se especifican los métodos y técnicas aplicados sistemáticamente durante todo el proceso de desarrollo del trabajo de titulación.

Capítulo 3. Marco Teórico

Dentro de este capítulo se describen los conceptos relacionados a este trabajo para un mejor entendimiento sobre lo que se realiza.

Capítulo 4. Trabajo Relacionado

Se mencionan los trabajos relacionados a este tema realizando una comparación entre lo que realizan con lo que este trabajo aporta. Luego se plantea un estado del arte sobre las herramientas que existen en la actualidad respecto a la validación de MC.

Capítulo 5. Comparación Teórica de las herramientas

En este capítulo se desarrolla la comparación teórica de dos herramientas seleccionadas luego de realizar un análisis de en base a las características y objetivos que plantea este trabajo.

Capítulo 6. Evaluación Empírica de las herramientas

Este capítulo describe todo el proceso experimental realizado para comparar la eficiencia, eficacia y satisfacción de usuario de las herramientas seleccionadas. También se incluye la discusión en base a los resultados obtenidos; y, se mencionan las posibles amenazas que podrían afectar los resultados del experimento.

Capítulo 7. Conclusiones y Trabajo Futuro

Este capítulo presenta las conclusiones finales del trabajo de titulación, así como los trabajos futuros que se pueden abordar a partir de los resultados obtenidos.

Referencias. Lista de referencias bibliográficas tomadas para el estudio y análisis de este trabajo de titulación.

Apéndices. En este apartado se presentan evidencias experimentales realizadas en este trabajo, capturas, modelos de cuestionarios y evidencias de recolección de datos. Adicionalmente, se anexa el artículo académico “¿Cómo pueden validarse los diagramas de Clases UML? Una Evaluación Empírica y Teórica de dos herramientas UML: USE y OCLE” que actualmente está en revisión para ser publicado en la conferencia CLEI 2020- XLVI Conferencia Latinoamericana de Informática, que tiene previsto ser desarrollada el 19 de octubre de 2020.

CAPÍTULO 2: METODOLOGÍA

En este capítulo se describe la metodología aplicada para este trabajo de titulación. La Fig. 1 resume la metodología aplicada.

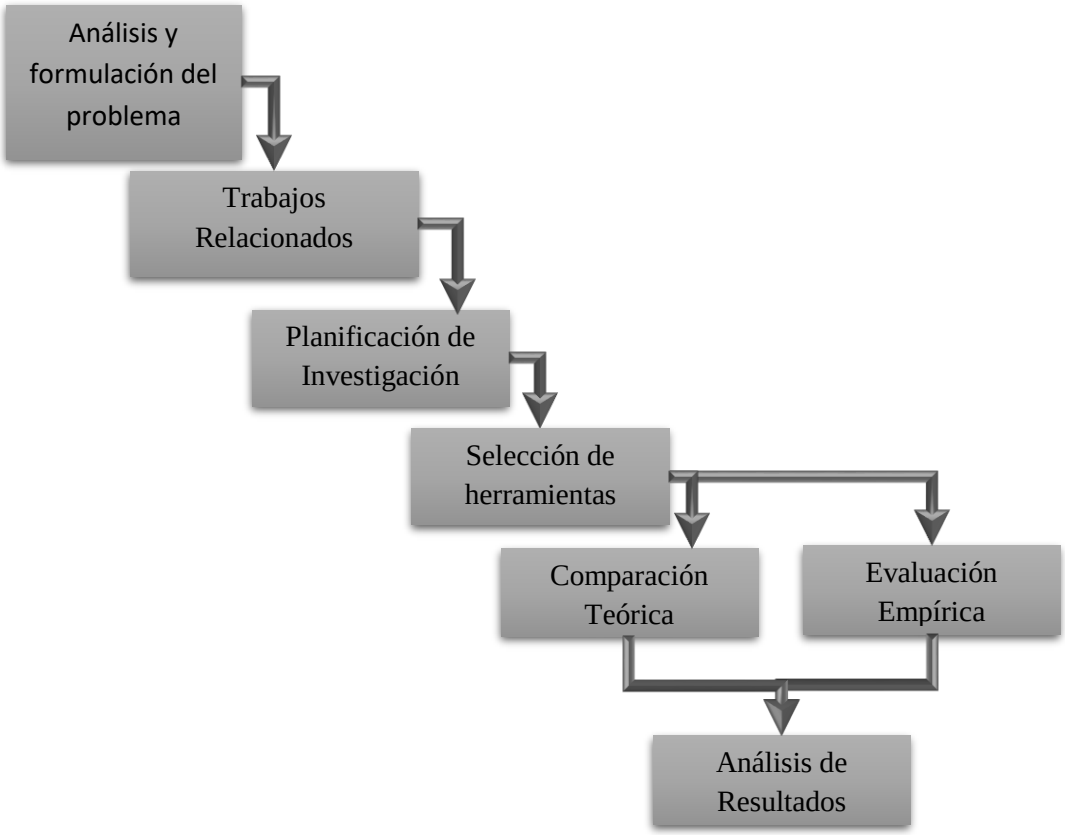


Fig. 1. Esquema general de la metodología aplicada.

Las fases de la metodología se describen a continuación.

1. **Análisis y Formulación del problema.** En esta etapa se estructura formalmente la idea del trabajo. Además, se delimita el campo de investigación estableciendo claramente los límites de los cuales se desarrollará este trabajo.
2. **Revisión de trabajos relacionados.** Describe las tecnologías relacionadas a la validación de un MC, así como los avances de investigación que se han venido desarrollando y que son cercanos a nuestro trabajo.

3. **Planificación de la investigación.** Se plantea una definición de criterios de evaluación para las herramientas como tipo de licencia, facilidades para la edición de los MCs, soporte para la validación y tipos de validaciones que se utilizan. Con los criterios mencionados se delimita la búsqueda de herramientas que validen MC usando diagramas de clases UML.
4. **Selección de Herramientas:** Para la selección de herramientas, se realiza un proceso ordenado como indica la *Fig. 2*. Inicialmente se recolecta información de herramientas relacionadas a evaluar MCs basados en diagramas de clase UML, describiendo el perfil de cada una. Luego, se realiza la preselección de las herramientas que son de libre acceso, esto con el fin de acceder a todo el contenido y poder analizarlas completamente. Una vez obtenidas las herramientas en base a las características mencionadas, se procede a seleccionar y evaluar dos de ellas. Para la selección de las dos herramientas, se consideran las que validen MC utilizando OCL como lenguaje por defecto para plantear restricciones y también las que permitan evaluar el MC mediante la ejecución de casos de prueba.

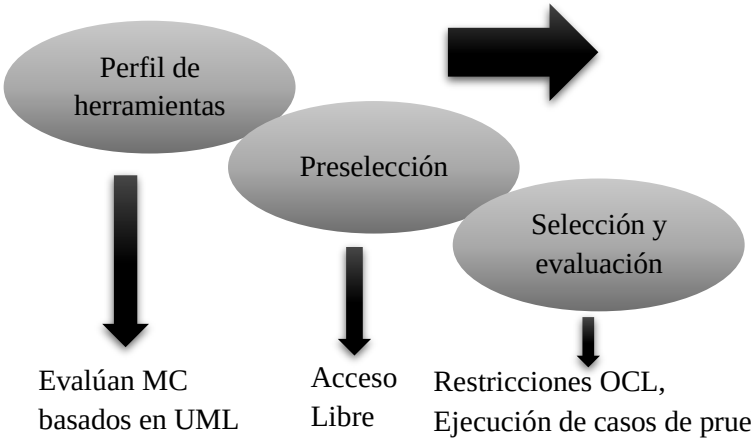


Fig. 2. Proceso de Selección de dos Herramientas.

5. **Comparación teórica.** La investigación teórica a realizarse es de tipo bibliográfica y analítica, la misma que permitirá obtener información relevante respecto a las herramientas de validan MC. Para realizar el siguiente trabajo se han denominado diferentes etapas que se centrarán en la comparación de las herramientas, las cuales son:

- Investigación bibliográfica de trabajos que correspondan a estudios similares desarrollados en el mismo campo (herramientas y trabajos relacionados a la validación de MCs).
- Determinación y procesamiento de datos recopilados.
- Establecimiento de un análisis comparativo de las diferentes herramientas que validan MC y que cumplen con los objetivos planteados en este trabajo. Para este análisis se usará una matriz comparativa aplicando criterios relacionados con el dominio del MC, los tipos de verificación que se ejecutan, la técnica de validación que se aplica, la calidad de las herramientas y por último la forma de ejecución, estos criterios permiten analizar los siguiente:
    - Dominio de MC
        - Tipo de Modelado
        - Lenguaje de la herramienta
        - Formato de MC
        - Forma de creación de Elementos
    - Verificación
        - Tipos de datos
        - Sintaxis del MC
        - Restricciones
    - Validación
        - Técnicas de validación que soporta
        - Nivel de automatización
        - Tipos de casos de prueba que soporta
    - Calidad
        - Tipos de defectos que detecta la herramienta
        - Objetivos de validación
    - Ejecución
        - Ambiente para su ejecución
        - Retroalimentación de la simulación del modelo
        - Tipos de mensajes en la retroalimentación
        - Scripts de prueba por lotes
        - Localización de fallas en el MC
        - Retroalimentación de la ejecución de restricciones

- Análisis de los datos obtenidos y verificación de los mismos.
- Elaboración de los resultados y la conclusión sobre las herramientas analizadas, selección de la mejor alternativa.

6. **Evaluación Empírica.** Esta evaluación se ejecuta luego de haber seleccionado dos herramientas de entre las que se analizan inicialmente. En esta fase se utilizará el modelo GQM (Goal-Question-Metric)[10] el cual es un paradigma para desarrollar y mantener un programa de métricas que ayudan a lo siguiente:
   - Definir el Objetivo. El objetivo por el que se realiza la evaluación empírica de las herramientas.
   - Generación de preguntas de investigación. Son planteadas para resolver los problemas que intentamos resolver al llevar a cabo este estudio empírico.
   - Formulación de hipótesis. Con esto podemos identificar el impacto de las variables independientes sobre las dependientes.
   - Establecer Variables y Métricas. En esta parte se identifican las variables dependientes e independientes necesarias en este estudio.
   - Contexto Experimental. Se identifica a los sujetos involucrados, los materiales digitales para el experimento y también una descripción de lo que se plantea para llevar a cabo el procedimiento experimental.
   - Procedimiento Experimental. Se detallan las actividades realizadas en el experimento.
   - Análisis e interpretación de resultados. Se muestran y se interpretan los resultados con respecto a las variables de estudio.
7. **Discusión Final sobre los Resultados**. Discutiremos los resultados finales obtenidos y responderemos las preguntas de investigación planteadas en este trabajo, además describiremos las amenazas que hemos podido identificar en el transcurso de este trabajo.

CAPÍTULO 3: MARCO TEÓRICO

En este capítulo de describen todos los conceptos que deben tomarse en cuenta para contextualizar el desarrollo de este trabajo de titulación, de esta forma se tendrán claros los procesos que se vayan realizando.

3.1 Conceptos Relacionados con la Validación de MC basados en Diagramas de Clase

3.1.1 Modelo Conceptual

Es una descripción en lenguaje de alto nivel sobre una aplicación. Presenta todos los conceptos relacionados a la aplicación, describe cómo los conceptos se relacionan entre sí cómo estos se ajustan a las tareas que los usuarios realizan con la aplicación [11]. En cuanto calidad de Software se toma en cuenta los factores de un producto de software, que contribuyen a la satisfacción de las necesidades de un usuario u organización [12]. La segunda fase del ciclo de vida del software en cascada es la ingeniería de requisitos y es muy importante en este análisis ya que en esta etapa se obtienen los requisitos para poder definir el modelo estructural de un sistema [13].

3.1.2 Pruebas de Software

En el contexto del Desarrollo Dirigido por Modelos MDD por sus siglas en inglés Model-Driven Development) se trata de ejecutar pruebas en el artefacto a partir de su modelo en lugar de un código fuente [6]. Para realizar las pruebas el MC debe estar en un formato que sea ejecutable, además, las pruebas se basan en ejecutar casos de prueba, los cuales definen valores de entrada para obtener una salida esperada.

3.1.3 Tipos de Pruebas

De acuerdo a Tort [14], los tipos de pruebas que se pueden realizar a nivel de modelo conceptual están relacionados con la verificación de:

- La consistencia de un estado.

- La inconsistencia de un estado.
- La ocurrencia de un evento de dominio.
- La no ocurrencia de un evento de dominio.
- Verificar el contenido de un estado.

3.1.4 Verificación y Validación de Software

Consiste en operar el MC bajo condiciones controladas para: (1) verificar que se comporta como se especifica; (2) detectar defectos y (3) validar los requisitos del usuario. En la literatura relacionada, se centra en una verificación automática de propiedades deseables en el MC a analizar (p. ej., análisis de sintaxis, semántica, dominio) [6].

3.1.5 Tipos de Defectos en MC Basados en UML

Un MC no siempre puede representar la funcionalidad para la que está destinado. Las causas y consecuencias de las desviaciones del funcionamiento esperado en el MC son factores que afectan la calidad del producto de software, a continuación, se definen los términos que definen defectos en un MC:

- Defecto. Es una imperfección o deficiencia en un producto de trabajo en donde no cumple con los requisitos o especificaciones por lo que necesita ser reparado o reemplazado.
- Anomalía. Es la presencia de un defecto en un MC.
- Falla. Es un evento en el cual el MC no realiza una función dentro de los límites especificados.

3.2 Conceptos relacionados con la Evaluación Empírica de Software

Como evaluación empírica de software se refiere proponer y evaluar un software mediante estudios empíricos, p. ej. por medio de casos de estudio o experimentos. Es un conjunto de estudios realizados, en base a percepciones humanas subjetivas lo cual permite la recopilación de datos cuantitativos, cualitativos, subjetivos y objetivos, y con ello emitir conclusiones e

interpretaciones. La estrategia para la recopilación de datos puede ser mediante cuestionarios [15]. El ciclo para una evaluación empírica [16] consiste en las siguientes etapas (ver Fig. 3):

Observación: La observación consiste en recoger y organizar los hechos empíricos para formar hipótesis.

Inducción: La inducción es el proceso de formación de hipótesis.

Deducción: Deducir las consecuencias con los datos empíricos recién adquiridos.

Pruebas: Probar la hipótesis con los datos empíricos.

Evaluación: Realizar la evaluación de los resultados de las pruebas.

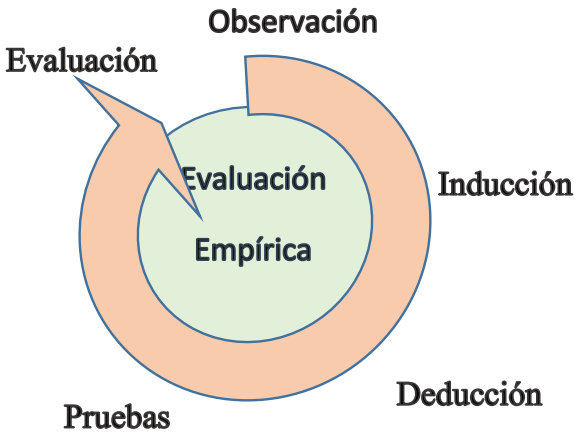


Fig. 3. Ciclo para evaluación Empírica.

3.3 Herramientas para validar MC basados en UML

En este apartado se realiza un análisis de herramientas existentes en la actualidad y posteriormente se selecciona dos herramientas que se utilizaran para realizar la comparación teórica y experimental propuesta en este trabajo. Las herramientas identificadas luego de la revisión en la literatura de trabajos relacionados son:

3.3.1 Papyrus [17]

Es un entorno de modelado de acceso libre basado en la plataforma Eclipse con soporte incorporado para personalizaciones UML y para el Lenguaje de Especificación de Sistemas (SysML)[2]. Disponible para varias plataformas

[2] https://sysml.org/

(Windows, Linux, macOS) con licencia pública de Eclipse (EPL) [3]. Esta herramienta permite técnicas como: simulación basada en modelos, pruebas formales basadas en modelos, análisis de seguridad, exploración de arquitectura, entre otros. Para ejecutar modelos, Papyrus utiliza un marco de animación y simulación rico y extensible llamado Moka[4]. Además, como el modelado gráfico no siempre es la mejor manera de especificar el comportamiento de los modelos ejecutables, Eclipse Papyrus proporciona edición de notación textual con resaltado de sintaxis, finalización y asistencia de contenido basado en el Lenguaje de Acción para UML (Action Language for Foundational UML - ALF) [5]. Sin embargo, no permite la validación con pruebas usando sólo diagramas de clase UML, sino que requiere integrar diagramas de actividad para la especificación del comportamiento, por lo que no ha sido considerada para este estudio.

3.3.2 Fujaba

Es el acrónimo de “From UML to Java and back again” (de UML a Java y viceversa). Es una herramienta de acceso libre, integrada dentro de la plataforma Eclipse y usada en la academia. Disponible para varias plataformas (Windows, Linux, macOS). Ofrece un lenguaje de especificación visual formal que se puede utilizar para especificar la estructura y el comportamiento de un sistema de software en desarrollo. La estructura se especifica mediante diagramas de clase UML, el comportamiento se especifica mediante los denominados diagramas de historia, una combinación de diagramas de actividad UML y patrones de historia. Los patrones de historias son reglas de transformación de gráficos que especifican modificaciones de estructuras de objetos (modelos). Esta herramienta no permite la validación de diagramas de clase mediante la especificación de casos de prueba, por lo que queda fuera de nuestro estudio. Adicionalmente, no dispone de soporte[6].

[3] https://www.eclipse.org/legal/epl-v10.html
[4] https://wiki.eclipse.org/Papyrus/UserGuide/fUML_ALF_Moka
[5] https://www.omg.org/spec/ALF/About-ALF/

[6] https://web.cs.upb.de/archive/fujaba/about-fujaba.html

3.3.3 Umple

La palabra 'Umple' es un juego de palabras, que significa 'Simple', 'Lenguaje de programación UML' y ‘Amplio’. Es un proyecto de acceso libre que funciona en línea, como un complemento de Eclipse y como un Jar de línea de comandos independiente. Disponible para varias plataformas (Windows, Linux, macOS) y con soporte limitado. Usado en la academia e industria. Tiene su propio lenguaje y compilador que facilitan la edición del código proporcionando implementaciones predeterminadas razonables. Permite la creación de diagramas de clase UML y diagramas de estado de manera textual. La validación de MC la realiza mediante OCL básico con pequeñas adaptaciones en cuanto al código. Sin embargo, no soporta restricciones y validaciones complejas[7], por lo que no se consideró como parte de nuestro estudio.

3.3.4 BoUML

Es una herramienta de acceso libre y multiplataforma (Linux, Mac y Windows). Permite la creación de varios diagramas UML como diagramas de clases, objetos, paquetes, despliegue, componentes, casos de uso, estado, actividad, secuencia, iteración, entre otros. Permite la creación y de MC basados en diagramas de clases y también permite la inserción de restricciones, precondiciones y postcondiciones, sin validar su sintaxis. Tiene limitaciones en cuanto al manejo de restricciones, no valida su sintaxis, así como tampoco valida los tipos de datos cuando se asignan valores en los atributos de un objeto[8].

3.3.5 USE [18]

Es el acrónimo de UML-based Specification Environment, es un lenguaje basado en UML y OCL. Una especificación USE contiene una descripción textual de un modelo utilizando características que se encuentran en los diagramas de clases UML (clases, asociaciones, etc.). Las expresiones escritas en el lenguaje de restricción de objetos (OCL) se utilizan para especificar restricciones de integridad adicionales en el modelo. Se puede animar un modelo para validar la especificación según los requisitos. Los estados del sistema (instantáneas de un

[7] https://cruise.eecs.uottawa.ca/umple/SimpleConstraints.html.
[8] www. https://www.bouml.fr/index.html

sistema en ejecución) se pueden crear y manipular durante una animación. Para cada instantánea, las restricciones de OCL se verifican automáticamente. Dispone de un lenguaje de programación completo llamado SOIL, que permite la creación de scripts de prueba para validar posibles escenarios y ejecutarlos por lotes. Es una herramienta multiplataforma (Linux y Windows) y de acceso libre.

3.3.6 OCLE

La palabra OCLE es el acrónimo de “Object Constraint Language Environment” (Entorno de lenguaje de restricción de objetos). Es una herramienta de acceso libre que permite el modelado de diferentes diagramas UML entre ellos diagramas de clase. Adicionalmente permite la inserción de restricciones en un diagrama usando OCL. Facilita la creación de scripts de prueba que se pueden ejecutar por lotes. El usuario tiene la posibilidad de navegar por el modelo y hacer una evaluación detallada de las expresiones de las invariantes para encontrar las causas de los defectos reportados. En esta herramienta se pueden crear varios MC en un solo proyecto, por lo que las restricciones se pueden crear para un MC específico o a nivel general para todos los MC que contenga el proyecto, cabe mencionar que soporta todo tipo de restricciones OCL complejas o simples[9].

En la Tabla 1 se muestra un resumen de las características que se ha analizado de las herramientas que permiten validar MC basados en diagramas de clases.

Otras herramientas como ALTOVA UModel, Visual Paradigm se consideran fuera del alcance de este trabajo ya que son comerciales y requieren el pago de una licencia para su uso. Adicionalmente, hemos encontrado varias herramientas que permiten gestionar y verificar MC basados en diagramas de clase UML como StarUML[10], Umbrello[11], GenMyModel[12], UMLetino[13],

[9] http://lci.cs.ubbcluj.ro/ocle/overview.htm
[10] www. staruml.io/
[11] www. umbrello.kde.org/
[12] www. genmymodel.com/
[13] www.umlet.com/umletino/umletino.html

Diagramo[14], Astah[15], ConceptDraw[16], Dia[17], Sparxsystems[18], Gliffy[19], Lucidchart[20], Magic Draw[21], Visio[22], Nclass[23], Openmodelshere[24], Reactive Blocks[25], Software ideas modeler[26], UML Designer[27], WhiteStarUML[28], Modelio[29] entre otras. Sin embargo, estas herramientas no soportan el proceso de validación basado en pruebas, por lo que no se las consideró para este trabajo.

Tabla 1. Características de las herramientas relacionadas a validación de MC.

Características de las Herramientas	Herramientas					
	Papyrus	Fujaba	Umple	BoUML	USE	OCLE
Licencia libre	●	●	●	●	●	●
Visualización gráfica del MC	●	●	●	●	●	●
Edición gráfica del MC	●	●		●		●
Edición textual del MC			●		●	●
Validación basada sólo en diagramas de clase		●	●	●	●	●
Lenguaje para validar restricciones	●	●	○		●	●
Lenguaje para casos de prueba	●		●		●	●
Ejecución de scripts por lotes					●	●
Dispone de soporte	●			●	●	●
Soporta multiplataforma	●	●	●	●	●	●
Editor de texto para restricciones						●
Validación de MC mediante línea de comandos					●	●
Edición de casos de prueba	●				●	●

● dispone de la característica

○ dispone de la característica parcialmente

[14] www. diagramo.com/
[15] www. astah.net/
[16] www. conceptdraw.com/products/drawing-tool
[17] www. dia-installer.de/
[18] www. sparxsystems.com/
[19] www. gliffy.com/
[20] www. lucidchart.com/pages/
[21] www. nomagic.com/products/magicdraw
[22] www. products.office.com/en-in/visio/flowchart-software
[23] www. nclass.sourceforge.net/
[24] www. modelsphere.com/org/
[25] www. bitreactive.com/reactive-blocks/
[26] www. softwareideas.net/
[27] www. umldesigner.org/
[28] www. whitestaruml.sourceforge.net/
[29] www.modelio.org/

De las herramientas analizadas en esta sección se ha seleccionado solamente dos, aquellas que disponen de un lenguaje para escribir y ejecutar casos de prueba. Adicionalmente son de acceso libre, y aún disponen de soporte como es el caso de USE y OCLE

CAPÍTULO 4: TRABAJO RELACIONADO

En este capítulo se revisan los trabajos relacionados más relevantes sobre comparaciones empíricas y teóricas de herramientas UML, de acuerdo con varias perspectivas.

4.1 Trabajos que realizan una comparación de Herramientas que soportan la validación de MCs basados en UML

En los últimos años, una cantidad cada vez mayor de investigación en Ingeniería de Software se ha dedicado a analizar y experimentar cómo los estudiantes y profesionales utilizan los métodos y herramientas de Ingeniería de Software.

En [19], se han recolectado varias herramientas que ejecutan MC basados en UML y OML, éstas se han clasificado según las soluciones que ofrecen al ejecutar un MC, entre las clasificadas se encuentran las aplicaciones que validan MC basados en diagramas de secuencia, de colaboración y de estados. Este estudio concluye que la mejor compresión de modelado depende del tipo de diagrama y del dominio de la aplicación, adicional a ello se concluye que los MC basados en diagramas de estado proporcionan una interpretación más segura de la información, mientras que MCs basados en diagramas de secuencia, son los que aportan mayor estabilidad semántica, sin embargo ninguno de los diagramas propuestos en este estudio incrementan la capacidad de comunicación de la especificación dinámica de un diseño UML. En nuestro trabajo respecto al modelado de MCs se hace uso de diagrama de clase UML lo cual en este trabajo mencionado no lo consideran, lo que tomamos en cuenta en este trabajo son las herramientas que se mencionan para posteriormente estudiarlas. En [20] se propone un enfoque para generar automáticamente una secuencia de instantáneas de comportamiento. Las restricciones de estas secuencias de instantáneas se expresan utilizando el OCL, este trabajo analiza el comportamiento de los MCs utilizando USE y OCLE. Los resultados en este trabajo muestran que existen limitaciones en la forma de especificación de una operación en un MC en la fase de diseño, además se menciona que este experimento sólo se puede aplicar para verificar sistemas donde las operaciones

ocurren secuencialmente. Se plantea como trabajo futuro eliminar la mayor cantidad de limitaciones mientras se preserva el nivel de automatización. En [21], se realiza un análisis de características funcionales entre dos herramientas USE y OCLE que permiten crear y validar MC en base a sus características, posteriormente en base a la información obtenida de la herramientas, se determina cuál de las dos es la mejor y como resultado final se obtiene que es OCLE. Nuestro trabajo tiene un enfoque similar, pero la diferencia es que además de comparar las características de las herramientas, se realiza una evaluación empírica para medir su eficacia, eficiencia y satisfacción de usuario.

En [22] se realiza una evaluación empírica de herramientas UML para comparar su productividad en términos de capacidad de aprendizaje, tiempo y número de clics requeridos, y carga de memoria requerida para que el Ingeniero de Software complete una tarea. Se enfoca en tres herramientas (Papyrus, IBM Rational Software Architecture y MagicDraw) de las cuales dos son de licencia comercial (IBM Rational Software Architecture y MagicDraw). Los resultados de este trabajo sugieren que MagicDraw se desempeñó significativamente mejor en términos de capacidad de aprendizaje, carga de memoria e integridad de la máquina de estados y el diagrama de secuencia. MagicDraw superó a IBM RSA y Papyrus en términos de integridad de máquinas de estado y diagrama de secuencia, capacidad de aprendizaje y carga de memoria. En términos de tiempo y número de clics, IBM RSA fue significativamente mejor al modelar diagramas de clase y máquinas de estado en comparación con Papyrus, sin embargo, ninguna herramienta sola superó a otras en todas las tareas de modelado. A diferencia de nuestro trabajo se consideran sólo herramientas con licencia libre que permiten validar los MC basados en diagramas de clase. Otro estudio realizado, descrito en [23], tiene como objetivo analizar cómo los modeladores construyen modelos UML evaluando la usabilidad desde tres perspectivas como son el proceso de modelado, el esfuerzo de modelado y los obstáculos de modelado que se encuentran durante el proceso además se analiza cómo Las herramientas de modelado apoyan esta tarea para mejorar el rendimiento de las herramientas de modelado UML, así como la forma en que se enseña el modelado UML con respecto a la construcción de diagramas de clase UML desde la perspectiva del modelador. Dicho estudio se basa en dos herramientas

Papyrus y MagicDraw(comercial). El resultado del experimento en este trabajo informa que no hay diferencias notables con respecto a la usabilidad de MagicDraw y Papyrus, además se indica que ambos mostraron importantes deficiencias que podrían mejorarse para proporcionar una experiencia de modelado significativamente mejor. El experimento reportado en [24] presenta un enfoque de aplicar técnicas de usabilidad para evaluar herramientas de modelado UML y una comparación de seis herramientas mediante un estudio empírico y método analítico GOMS para modelar los objetivos de los usuario y comparar la cantidad de pasos necesarios para lograr los objetivos con cada herramienta. Este estudio reporta que las herramientas Visual Paradigm, Jude, Poseidon y ArgoUML son las mejores, como siguiente herramienta y con una pequeña diferencia a la anteriores se encuentra Enterprise Architect y como último lugar es la herramienta Metamill, dando como conclusión que los problemas de usabilidad son muy importantes ya que en base a este trabajo se plantean muchos trabajos futuros.

Hasta donde sabemos, nuestro estudio es el primero que analiza la eficiencia, eficacia y satisfacción del usuario en el uso de herramientas de modelado UML, desde el punto de vista de validaciones basadas en pruebas de diagramas de clases. En la Tabla 2 se muestra un resumen comparativo de los estudios realizados según sus objetivos, herramientas bajo estudio, el enfoque de investigación empleado, tipos de participantes y los datos analizados durante el estudio. La última fila se muestra los datos analizados para fines de comparación entre los trabajos.

Tabla 2. Trabajos relacionados sobre Herramientas de Modelado UML.

REF.	OBJETIVO	HERRAMIENTAS RELACIONADAS	TIPO DE ANÁLISIS	PARTICIPANTES	DATOS ANALIZADOS
[19]	Evaluación empírica en medir la eficacia de modelado dinámico UML y OML de los MCs.	▪ Programas escritos en Fortran	▪ Paradigma GQM ▪ Análisis Empírico	▪ Estudiantes ▪ Profesionales informáticos	▪ Técnicas de verificación ▪ Tipo de programa ▪ Nivel de experiencia
[20]	Analizar el comportamiento de los MCs utilizando USE y OCLE mediante secuencia de instantáneas.	▪ USE ▪ OCLE	▪ Análisis Comparativo	▪ N/A	▪ Comportamiento especificado en diagramas de secuencia y modelos de instantáneas.
[21]	Comparar y analizar dos herramientas con validación OCL.	▪ USE ▪ OCLE	▪ Análisis comparativo	▪ N/A	▪ Características funcionales de cada herramienta.
[22]	Comparar la productividad de tres de las herramientas de modelado UML.	▪ IBM Rational Software Architect (RSA) ▪ MagicDraw ▪ Papyrus ▪ Enterprise Architect ▪ Visual Paradigm ▪ Rational Rose ▪ Argo UML	▪ Análisis comparativo	▪ Estudiantes universitarios ▪ Profesionales	▪ El esfuerzo de modelado requerido para completar correctamente una tarea. La capacidad de aprendizaje. ▪ La carga de memoria requerida para que el ingeniero complete una tarea.
[23]	Analizar cómo se crean los diagramas de clase UML utilizando dos herramientas	▪ MagicDraw ▪ Papyrus	▪ Análisis comparativo	▪ Estudiantes de pregrado	▪ La estrategia de modelado.

	de modelado y qué tan bien las herramientas de modelado apoyan esta tarea.		▪ Análisis empírico		▪ El esfuerzo de modelado. ▪ Los obstáculos de modelado.
[24]	Comparación de modelado de seis herramientas.	▪ Visual Paradigm ▪ Jude ▪ Poseidon ▪ ArgoUML ▪ Enterprise Architect ▪ Metamill	▪ Método analítico ▪ Avaluación de herramientas	▪ Estudiantes ▪ Profesionales	▪ Usabilidad

.

CAPÍTULO 5: ANÁLISIS COMPARATIVO TEÓRICO DE LAS HERRAMIENTAS

En este capítulo se aborda uno de los objetivos específicos de este trabajo de titulación que es el de realizar un análisis comparativo de dos herramientas luego de haber pasado por un proceso de selección (ver Capítulo 3.3). Se detallan las características mediante una tabla comparativa, también se describe la funcionalidad de cada una presentando como evidencia capturas de pantalla de las dos herramientas y de esta manera poder reflejar una comparación más visible.

5.1 Criterios de Comparación

Para la comparación teórica se seleccionaron criterios de comparación relevantes para nuestro estudio como: dominio de aplicación, los tipos de verificaciones y validaciones que admite cada herramienta, la calidad en cuanto a los tipos de defectos que detecta cada una y los objetivos de validación a lograr con cada herramienta; y, el ambiente donde se ejecuta la validación de MC. Cada uno de estos criterios se detallan a continuación y se resumen en la Tabla 3.

5.2 Análisis Basado en Criterios de Comparación

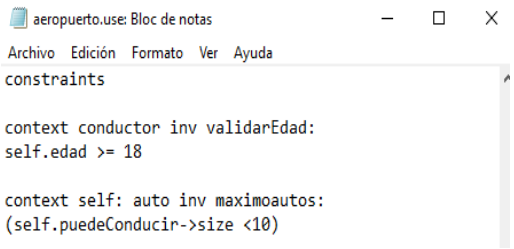
Tanto USE como OCLE soportan la versión UML 2.0 que provee un lenguaje de modelado más extensible, permitiendo la validación y ejecución de MC.

Para USE el MC es definido en código USE y las restricciones en OCL, los dos lenguajes son definidos en un mismo archivo de extensión “.use”.

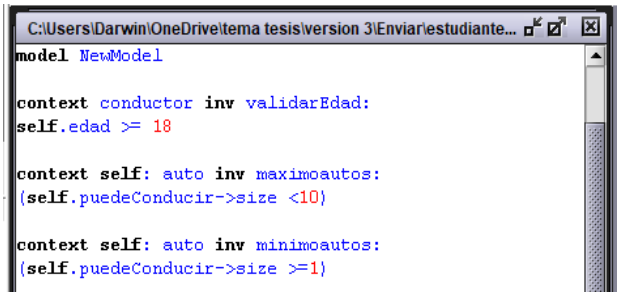
Tabla 3.Comparación Técnica De Herramientas Seleccionadas.

CARACTERÍSTICA ANALIZADA		**USE** [18]	**OCLE** [25]
Dominio	**Modelado**	UML 2.0	
	Lenguaje de herramienta	Notación USE	Notación OCLE
	Formato de MC	(.use)	(.ocl, oepr, .bcr,.zip)
	Creación de Elementos	Notación USE	Manipulando la interfaz gráfica
	Relaciones que soporta	• Asociación, Generalización, Composición, Dependencia.	
	Diagramas que admite	• Diagramas de clases, objetos, secuencia	• Diagrama de clases, casos de uso, objetos.
Verificación	**Tipos de datos**	SI	
	Sintaxis	SI	
	Restricciones (Invariantes, precondiciones, postcondiciones)	Mediante texto con notación OCL	Interfaz gráfica con notación OCL
Validación	**Técnicas de Validación que soporta**	• Testing del MC • Simulación de MC	
	Nivel de automatización	Automático	Semiautomático
	Tipos de Casos de prueba que soporta	• Inconsistencia de estados. • Dominio de un estado determinado. • La no ocurrencia de un evento. • La ocurrencia indebida de un evento.	
Calidad	**Tipos de defectos que detecta**	• Elemento faltante • Elemento incorrecto • Elemento innecesario • Sintaxis al ingresar datos	
	Objetivo de la Validación	• Verificar consistencia • Correctitud • Completitud	
Ejecución	**Ambiente**	USE	OCLE
	Retroalimentación de la simulación del modelo	SI	
	Tipos de mensajes en la retroalimentación	Tablas en la interfaz gráfica y mensajes de texto en consola	Mensajes en la interfaz gráfica
	Scripts de prueba por lotes	SI	
	Localización de fallas en MC	SI	
	Retroalimentación de la ejecución de restricciones	SI	NO

En OCLE, el modelo se crea mediante su editor gráfico, el cual es generado en un archivo de extensión ".xml" dentro de un archivo comprimido, en el cual también se encuentran los demás MC ya sean diagramas de clases, de caso de uso, etc. Las restricciones se encuentran en un archivo de extensión ".ocl", por último se tiene un archivo de extensión ".oepr" el cual une el MC y las OCL en un solo proyecto. Las relaciones entre elementos de un MC que admiten estas herramientas se basan en las relaciones UML (asociaciones, generalizaciones, composiciones y dependencia), el modelado a nivel de metamodelo es soportado únicamente por OCLE, este modelado se refiere a la creación de varios MC dentro de un mismo proyecto. Refiriéndose a la verificación, las dos herramientas permiten verificar el ingreso de tipo de datos a nivel de atributos, parámetros y valores para los métodos, así como de precondiciones, postcondiciones e invariantes. La diferencia está en que las restricciones para USE (Fig. 4.a) son ingresadas en el mismo archivo del proyecto utilizando un editor de texto particular, mientras que en OCLE (Fig. 4.b) se puede ingresar o modificar utilizando su propia interfaz gráfica.



a) Ejemplo de la definición de Restricciones en USE usando editor de texto

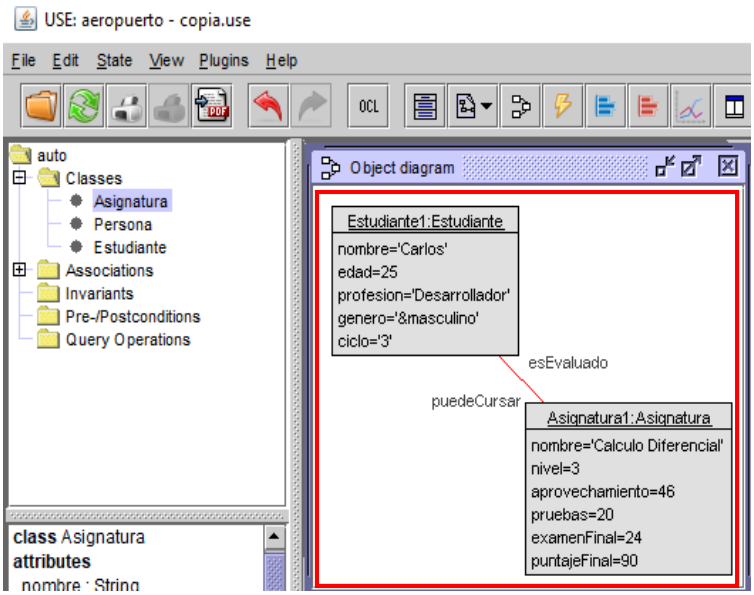


b) Ejemplo de la definición de Restricciones en editor OCLE.

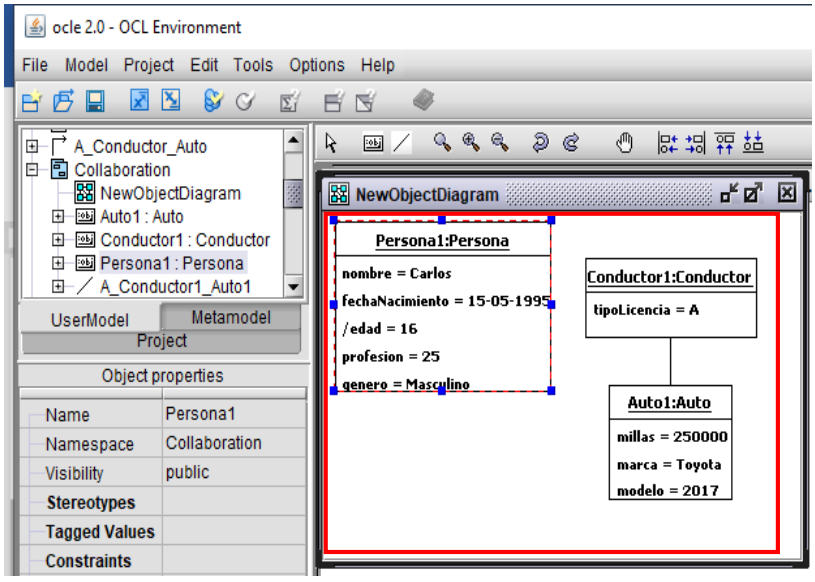
Fig. 4. Definición de Restricciones en a) USE y b) OCLE.

En cuanto a los métodos utilizados en el proceso de validación, USE y OCLE realizan técnicas de validación mediante pruebas del MC y simulación del MC

mediante el uso del diagrama de objetos como se indica en la Fig. 5. Una vez definido el diagrama de objetos se evalúa el MC en base a los valores ingresados en el diagrama, esto se validará considerando las restricciones planteadas en el proyecto.



a) Diagrama de Objetos en USE.



b) Diagrama de Objetos en CLE.

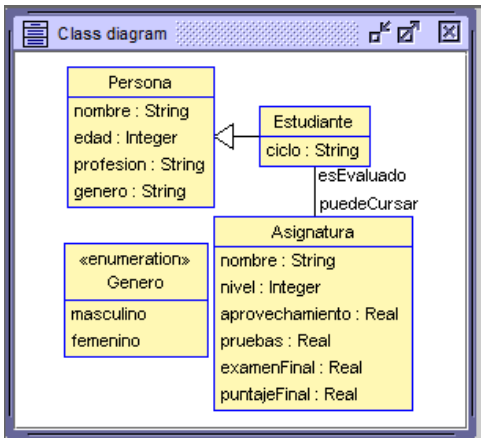
Fig. 5. Captura de pantalla de una instancia de MC usando a) USE y b) OCLE.

El modelamiento en USE se lo realiza mediante código USE y una vez compilado se puede generar el diagrama de clases (ver Fig. 6.a), la modificación del MC se lo

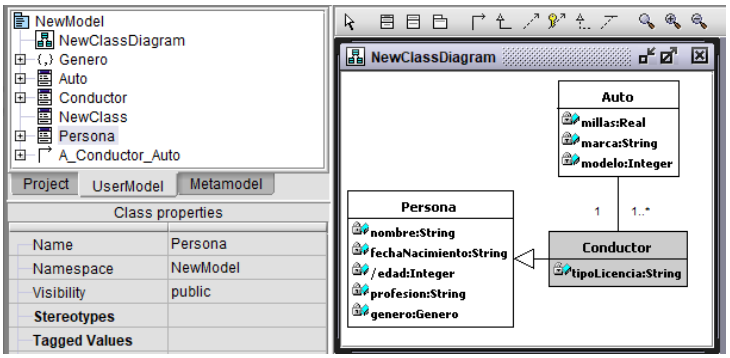
debe hacer de igual manera mediante notación USE, en OCLE el modelamiento es utilizando su interfaz gráfica arrastrando elementos (paquetes, clases, relaciones, etc) al panel gráfico, la modificación del MC se lo realiza gráficamente (ver Fig. 6.b). El análisis del MC en USE es automático, ya que la validación del MC se la realiza al momento de cargar el proyecto, mientras que en OCLE el análisis de MC se realiza una vez cargado el proyecto, se tienen todas las clases, relaciones y restricciones, pero no se visualiza el diagrama de clases, sino que se debe crear el diagrama arrastrando todos los elementos a un panel en donde finalmente se pueda observar cómo está diseñado el diagrama. Para analizar los tipos de casos de pruebas que soportan las herramientas, hemos usado los que plantea Tort [14], los mismos que se pueden ejecutar en ambas herramientas. Con respecto a la calidad de la validación de un MC, se realiza un análisis de los tipos de validaciones que las herramientas soportan. Según [21], USE y OCLE detectan los mismos tipos de defectos: elementos faltantes, elementos incorrectos, elementos no necesarios y sintaxis al ingresar datos. Los objetivos de las dos herramientas son verificar la consistencia, correctitud y completitud del MC.

```
model auto

--classes
enum Genero {masculino,femenino}

class Asignatura
attributes
      nombre: String
      nivel: Integer
      aprovechamiento: Real
      pruebas: Real
      examenFinal: Real
      puntajeFinal: Real
end

class Persona
attributes
      nombre: String
      edad: Integer
      profesion: String
      genero: String
end

class Estudiante < Persona
attributes
      ciclo: String
end
-- associations

association maneja between
  Estudiante[*] role esEvaluado
  Asignatura[1] role puedeCursar
end
-- constraints
```

Notación USE



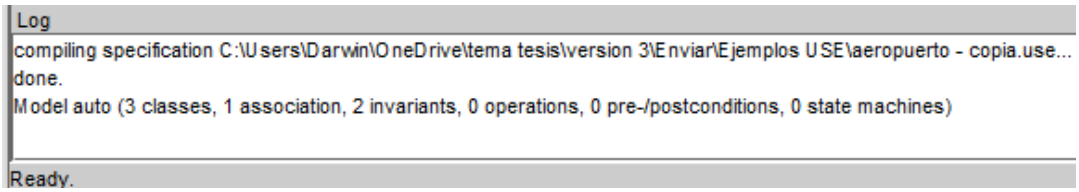
a) Diseño de MC mediante notación USE.



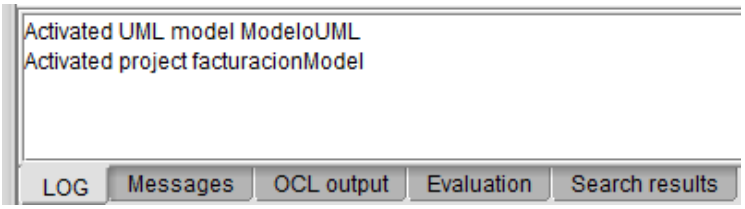
b) Diseño de MC mediante interfaz gráfica OCLE.

Fig. 6. Ejemplo de modelado de un MC usando a) USE y b) OCLE.

En la compilación de MC, USE compila automáticamente al momento de cargar el proyecto (ver Fig. 7.a). En OCLE la compilación del proyecto es de forma manual, se debe cargar el proyecto y posteriormente ejecutar la opción que permite compilar (ver Fig. 7.b).



a) Compilación automática en USE al momento de cargar el proyecto.

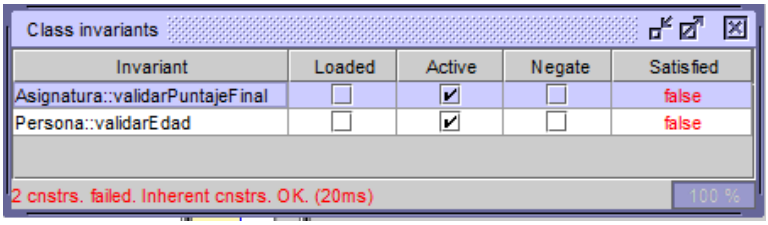


b) Carga del proyecto en OCLE, requiere compilación manual.

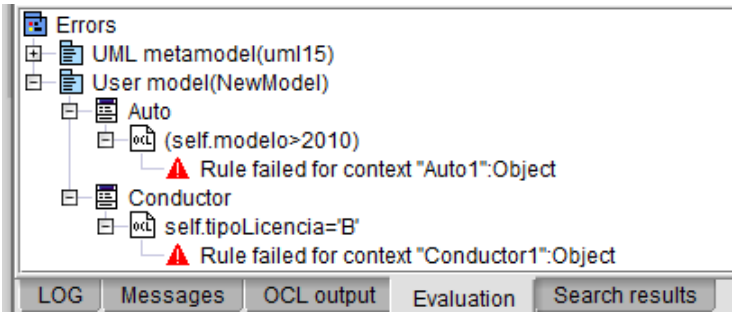
Fig. 7. Ejemplo de la Compilación de un MC en a) USE y b) OCLE.

También permiten la ejecución de prueba por lotes, es decir, un conjunto de validaciones puede ejecutarse en un solo proceso. Las dos herramientas permiten localizar defectos dentro del MC en caso de haberlos, en USE el defecto es localizado directamente en una tabla de restricciones indicando si los valores ingresados son correctos (true en verde) o incorrectos (false en rojo) (ver Fig. 8.a); OCLE indica un defecto en forma jerárquica (nombre del proyecto - objeto - nombre del atributo - valor incorrecto) (ver Fig. 8.b). Una vez localizados los defectos, USE

y OCLE permiten también la corrección de los elementos hasta conseguir un MC completamente validado.



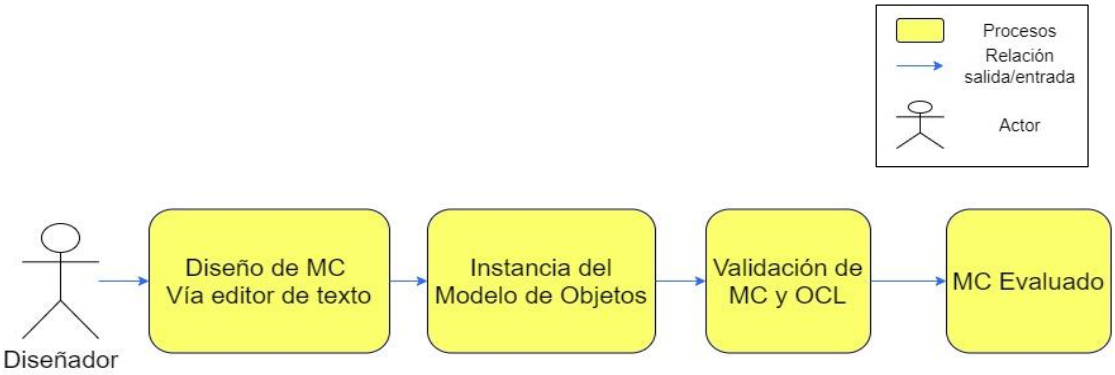
a) Tabla de resultados de un MC evaluado usando USE.



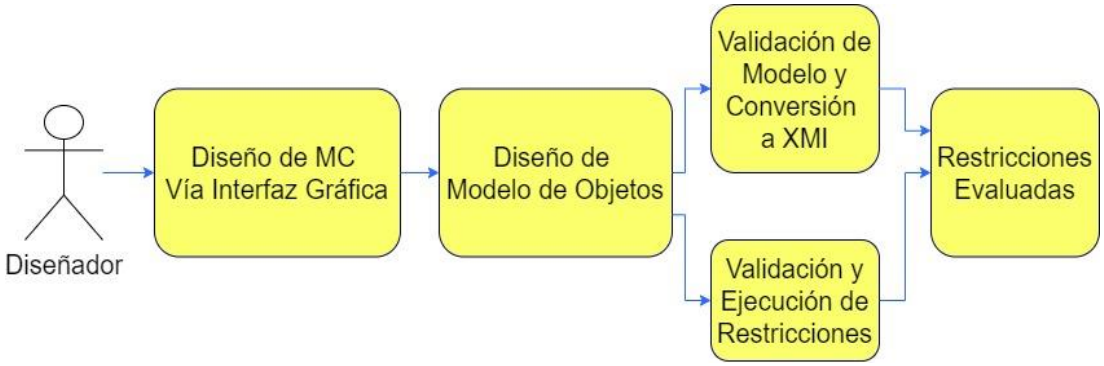
b) Menú desplegable de resultados de un MC evaluado en OCLE.

Fig. 8. Ejemplo de los resultados de validar un MC en a) USE y b) OCLE.

USE y OCLE tienen su propia arquitectura y trabajan de manera similar como se muestra en la Fig. 9. Ambas requieren para su funcionamiento cargar un proyecto, luego se procede a su compilación, la inserción de pruebas y emisión de resultados de evaluación en base a las restricciones.



a) Esquema General del Flujo de Trabajo para validaciones de MC en USE.



b) Esquema General del Flujo de Trabajo para validaciones de MC en OCLE.

Fig. 9. Flujo de Trabajo para validaciones de un MC usando a) USE y b) OCLE.

5.3 Conclusiones del Análisis Comparativo

En base al análisis realizado, se puede decir que USE y OCLE presentan características similares en los criterios de comparación seleccionados, habiendo pocas diferencias en cuanto a su funcionalidad y características que cada uno posee: (a) cada herramienta tiene su propia notación/lenguaje, (b) la estructura de un proyecto USE contiene un solo archivo mientas que OCLE se compone de varios tipos de archivos, (c) los diagramas que admite cada herramienta, (d) la forma de insertar las restricciones en un MC, (e) el nivel de automatización que soporta la validación, (f) el ambiente de ejecución, (g) los tipos de mensajes en la retroalimentación y, (h) la retroalimentación en la ejecución de las restricciones. El resto de las características analizadas del dominio, verificación, validación, calidad y ejecución son similares.

CAPÍTULO 6: EVALUACIÓN EMPÍRICA DE LAS HERRAMIENTAS

En este capítulo se evalúa empíricamente las dos herramientas anteriormente seleccionadas. Para lo cual, se describe el plan experimental, el entorno en que se desarrolló el experimento, el procedimiento que se ejecutó, los resultados obtenidos con su respectivo análisis y un resumen de las amenazas de esta evaluación.

6.1 Plan Experimental

Para esta parte del experimento nos hemos basado de acuerdo a Wohlin et al. [26] y hemos reportado de acuerdo a Juristo y Moreno[27].

6.1.1 Objetivo

En línea con el paradigma Objetivo/Pregunta/Métrica [10], el objetivo de nuestro estudio empírico es el siguiente: Analizar las herramientas de validación USE y OCLE con el propósito de realizar una evaluación empírica con respecto a la eficacia y eficiencia en detectar fallos en modelos conceptuales y la satisfacción de usuario de las mismas desde el punto de vista del investigador en el contexto de estudiantes de Informática validando requisitos.

6.1.2 Preguntas de Investigación

Se han planteado las siguientes preguntas de investigación:

PI1: ¿Existe una diferencia significativa entre el grado de eficiencia para detectar defectos en MC usando USE y usando OCLE?

PI2: ¿Cuál de las dos herramientas presenta un mayor grado de eficacia en detectar la mayor cantidad de defectos en un MC?

PI3: ¿Cuando los participantes están validando los MC usando las herramientas seleccionadas, su percepción de satisfacción de usuario es impactada?

6.1.3 Hipótesis

Se ha definido tres hipótesis (ver Tabla 4). Las hipótesis nulas (representadas por el subíndice 0), corresponden a la ausencia de un impacto de las variables independientes sobre las variables dependientes.

Las hipótesis alternativas involucran la existencia de un impacto y son el resultado esperado.

Tabla 4. Especificación de las Hipótesis.

ID de la Hipótesis Nula	Declaración: El tipo de herramienta no influencia en …
H_{10}	… la eficiencia de los participantes en detectar fallos en los modelos conceptuales.
H_{20}	… la eficacia de los participantes en detector fallos en los modelos conceptuales.
H_{30}	… la percepción de satisfacción de usuario de los participantes.

6.1.4 Variables y Métricas

Las variables y métricas consideradas en este experimento se definen a continuación:

6.1.4.1 Variables Independientes.

Consideramos una variable independiente (también conocido como factor [27]) la herramienta de validación. Esta variable puede tener dos valores (también conocido como tratamientos [27]): Usuarios aplican la herramienta USE y usuarios aplican la herramienta OCLE.

6.1.4.2 Variables Dependientes y Métricas.

Nosotros consideramos las siguientes cuatro variables dependientes (también conocidas como variables respuesta), que se esperan sean influenciadas en cierta medida por la variable independiente.

- **Eficacia del sujeto.** Para investigar nuestra P1 necesitamos medir la eficacia de encontrar defectos alcanzada con cada una de las herramientas utilizadas. El resultado será un porcentaje que permite valorar de forma comparativa, es decir, si se sitúa en los dos percentiles más bajos el trabajo será ineficaz, mejorando esta capacidad conforme se ascienda hacia el 100%.[28]

$$eficacia = \frac{defectos\ encontrados}{defectos\ inyectados} * 100 \quad (1)$$

- **La eficiencia del sujeto.** Esta variable nos permite investigar la P2 y su métrica se calcula de acuerdo a la siguiente fórmula:

$$eficiencia = \frac{defectos\ encontrados/tiempo\ invertido}{defectos\ previstos/tiempo\ previsto} \quad (2)$$

  El resultado será un valor que permite valorar de forma comparativa las herramientas, considerando que los valores mayores presentan mayor eficiencia y los valores menor deficiencia.[28]

- **Satisfacción de usuario percibida.** El grado en que un participante considera que usar una herramienta de validación de MC está libre de esfuerzo. Esta variable permite responder a la P3. Para calcular esta métrica se ha considerado el modelo TAM (Technology Acceptance Model)[29], que pretende explicar los factores que llevan al individuo a aceptar o rechazar un determinado sistema de información, explica el proceso de adopción a partir de dos factores fundamentales :
  - La utilidad percibida: Definida como “el grado en el que una persona cree que el uso de un sistema concreto mejoraría su desempeño laboral”.

- La satisfacción de usuario percibida: Que hace referencia a “el grado en que una persona cree que el uso de un determinado sistema será libre de esfuerzo”.

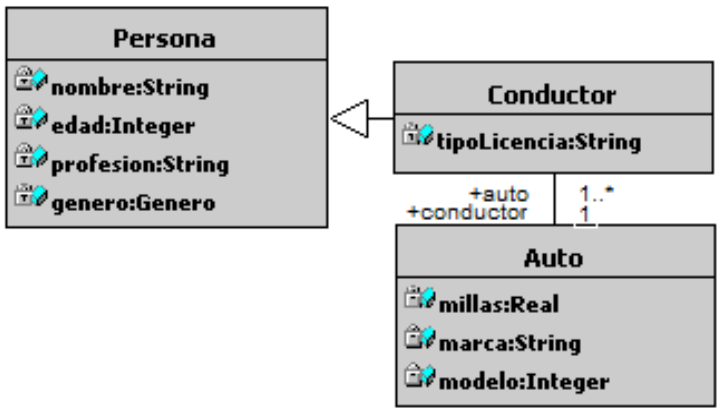
6.2 Contexto Experimental

6.2.1 Sujetos

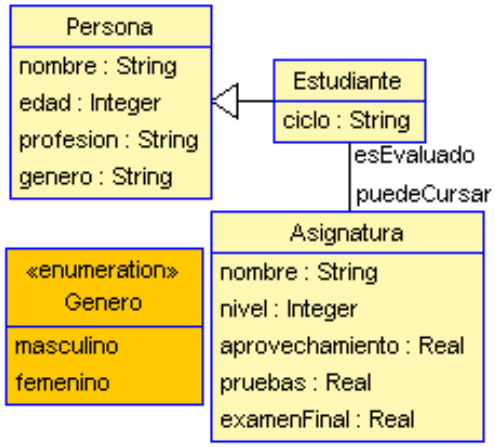
El experimento se llevó a cabo en el 2019 con una muestra a conveniencia, con 18 sujetos, 2 mujeres y 16 hombres, de entre 18 y 22 años, estudiantes de Ingeniería de Sistemas de la Universidad de Cuenca, con experiencia probada en realizar pruebas a nivel de código, así como en el uso de métodos y técnicas de modelado, entre ellas aplicando UML.

6.2.2 Modelos Conceptuales

Se utilizaron 4 modelos conceptuales diferentes en nuestro estudio, uno para la capacitación de USE (MC A), otro para la capacitación en OCLE (MC B) y los otros dos (MC C y MC D) para la sesión experimental con OCLE (ver Fig. 10.a) y USE (ver Fig. 10.b). Estos MC contienen una variedad de características que pueden tener un diagrama de clase UML, incluyendo clases, relaciones, y diferentes tipos de restricciones.



a) Diagrama de Clases para MC C, realizado en OCLE.



b) Diagrama de Clases para MC D, realizado en USE.

Fig. 10. Diagramas de Clases de MC C y MC D.

6.2.3 Fallas Inyectadas en los Modelos Conceptuales

En cada modelo conceptual (ver Tabla 5) usado en la fase experimental de este estudio, se inyectaron 5 defectos los cuales se mencionan a continuación:

6.2.3.1 Defectos en MC C

- **Elemento faltante.** Falta una restricción para validar la edad.
- **Elemento faltante**. Falta la enumeración género (masculino, femenino).
- **Elemento faltante**. Falta una restricción en el nombre para validar el ingreso de solo caracteres alfanuméricos.
- **Elemento incorrecto.** La relación Conductor-Auto no es la correcta, ya que a un conductor se le puede asignar hasta dos autos.
- **Elemento faltante**. Falta una restricción que valide el ingreso de las millas con valores correctos (no deben ser valores negativos).

6.2.3.2 Defectos en MC D

- **Elemento incorrecto.** La restricción que valida el puntaje total es incorrecta, debe validar que el valor de examenFinal sea menor o igual a 100.
- **Elemento Incorrecto.** El atributo “genero” no debe ser de tipo String sino del tipo enumeración “Genero”.
- **Elemento faltante.** Falta una restricción para validar el ciclo que cursa el estudiante en el intervalo de [1-10].
- **Elemento faltante.** Falta una restricción para validar que el valor del “puntajeFinal” de la asignatura corresponda a la suma de “examenFinal”, promedio de “pruebas” y promedio de “aprovechamiento”.
- **Elemento faltante.** Falta una restricción que permita validar que el atributo derivado “edad” de la clase persona sea mayor a cero y que se genere a partir de la fecha de nacimiento.

Tabla 5. Elementos de los Modelos Conceptuales utilizados en la Fase Experimental de este estudio.

Elemento	**MC C**	**MC D**
Descripción	Representa un sistema para cooperativa de taxis en el cual se requiere dar un mantenimiento de la información de los conductores con sus unidades asociadas a cada uno.	Describe básicamente a los estudiantes de una institución educativa, y las entidades que permiten dar seguimiento a las asignaturas que cursan.
Clases	3	3
Atributos	8	11

Operaciones	4	3
Asociaciones	2	2
Restricciones	• Valor numérico de las variables no pueden ser negativos. • Valor numérico de la edad del conductor que debe ser mayor de edad. • Valor en el género del conductor masculino o femenino. • El tipo de licencia debe ser de tipo B. • El nombre de un conductor no debe contener valores numéricos ni ningún otro carácter que no sea letras.	• Valor de las notas debe ser numérico y mayor a cero. • El estudiante puede cursar como máximo 6 asignaturas. • El estudiante tiene como mínimo una asignatura cursando. • El puntaje total debe ser la suma del aprovechamiento con el examen final. • Para que el estudiante apruebe la asignatura, ésta debe tener una puntuación mínima de 60/100.
Generalización	1	1

El tiempo previsto para detectar los 5 defectos es de 3 minutos por cada uno. Este tiempo es el valor promedio obtenido del estudio piloto que se ejecutó, para validar el material del experimento.

6.3 Procedimiento Experimental

Esta sección describe detalladamente los procedimientos que se han realizado para llevar a cabo el experimento sobre las dos herramientas.

Antes del experimento, los sujetos participantes completaron un cuestionario demográfico que reveló que ninguno de ellos había estado en contacto con las herramientas USE y OCLE. Basado en esta información, diseñamos sesiones de entrenamiento intensivo en el uso de las dos herramientas para equilibrar el conocimiento de los temas.

Se preparó el contenido y una planificación sobre cómo llevar las sesiones hacia los usuarios, para incorporar la participación de los mismos en el experimento, manteniendo los objetivos planteados. Los sujetos completaron las tareas experimentales y al final se evaluó cada una de sus percepciones mediante cuestionarios. En Fig. 11 se resume el proceso experimental, el cual se divide en las siguientes sesiones:

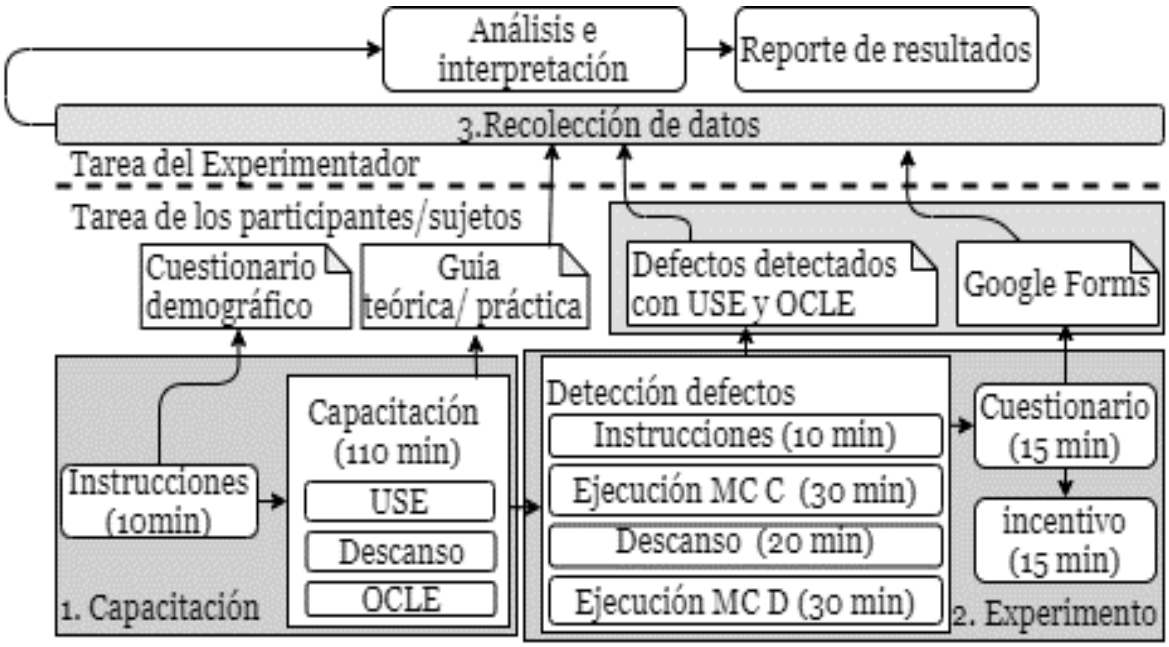


Fig. 11. Esquema General del Experimento.

6.3.1 Sesiones de Capacitación

Se efectuó una presentación general de las tareas que se van a realizar, luego los participantes llenaron el cuestionario demográfico (diez minutos). A continuación, se realizó la capacitación en el uso de la herramienta USE utilizando el MC A. Para la

capacitación en la herramienta OCLE se usó el MC B. Durante las sesiones de capacitación, los sujetos resolvieron algunos ejercicios y recibieron retroalimentación sobre su rendimiento. Esta sesión tuvo una duración de dos horas.

6.3.2 Sesión Experimental

En la sesión experimental, la detección de defectos se realizó usando como entrada el MC C (ver *Tabla 5*). Para este propósito, el grupo 1 procedió a usar la herramienta USE y el grupo 2 la herramienta OCLE. En la segunda parte, se usó como entrada el MC D (ver *Tabla 5*) y la detección de defectos se realizó usando la herramienta OCLE con el grupo 1 y la herramienta USE con el grupo 2.

6.3.3 Recolección y Tabulación de Datos

Al finalizar el uso de cada herramienta, se recolectó manualmente los datos para las tres variables del estudio:

- **Eficiencia y Eficacia.** El número de defectos detectados y el tiempo invertido para hacerlo durante el experimento.
- **Satisfacción de usuario.** los sujetos contestaron el cuestionario del modelo TAM [30] sobre la satisfacción de usuario usando un formulario implementado en Google Forms.

En este experimento los participantes fueron situados en un aula equipada con computadores, cada uno con las herramientas de software debidamente instaladas y probadas. Para plantear los MC C y D se estableció un tiempo de reloj fijo para todos los participantes con el fin de llevar tiempos iniciales y finales lo más preciso posible al momento de evaluar las herramientas y detectar los defectos, para poder estimar la eficacia y eficiencia. A los participantes se les entregó el MC C y D en forma digital en cada computador para su evaluación, adicional a ello, cada participante recibió un documento que incluye la descripción de cada MC propuesto, también se les entrega un documento en el cual se plantea un cuestionario para cuantificar los defectos que cada participante percibe con campos vacíos de descripción del defecto, defecto encontrado, defecto corregido, tiempos inicial y final,

los cuales deben ser llenados por cada uno de los participantes. Cada participante establece un tiempo inicial al momento de analizar el MC correspondiente y un tiempo final al terminar su análisis. Luego analizan los defectos y para esto también registran el tiempo que tardan en encontrar un defecto. Pueden encontrar los defectos que ellos creen conveniente, la decisión de terminar el análisis de defectos corresponde a cada participante. Luego de terminar el análisis, los participantes entregan los documentos facilitados para posteriormente procesar los resultados.

Para evaluar la satisfacción de usuario, el cuestionario fue propuesto vía internet usando un formulario elaborado con Google Forms. Este cuestionario fue llenado por cada uno luego de haber sido capacitado y puesto a prueba cada herramienta. El uso de un formulario basado en Google Forms facilitó la recolección de datos.

6.4 Análisis e Interpretación de Resultados

Esta sección resume los resultados de los datos recopilados de nuestro experimento con respecto a las tres variables de estudio: la eficacia (PI1), la eficiencia (PI2) y la satisfacción de usuario (PI3). El análisis estadístico fue realizado con el Paquete Estadístico para las Ciencias Sociales (SPSS) V23.0.

Debido a que la primera pregunta de investigación (PI1) fue evaluar la eficacia de las herramientas para detectar defectos, se compara el número de defectos encontrados usando cada herramienta (OCLE, USE) en los diferentes MC. La Tabla 6 muestra los defectos detectados (columna 2) por cada sujeto, y el valor de la Eficacia (columna 3). Los datos del tiempo empleado y tiempo previsto (columnas 4 y 5) se usan para calcular la eficiencia en detectar defectos por cada herramienta que se analiza más adelante (columna 6).

Tabla 6. Datos para la Eficacia, Eficiencia y Satisfacción de Usuario por Herramienta.

Herramienta	Número Detectados	Eficacia (%)	Tiempo empleado (min)	Tiempo previsto (min)	Eficiencia	Satisfacción de Usuario
OCLE	2	40	3	6	0,8	4,33
	2	40	5	6	0,5	4,67
	2	40	15	6	0,2	5,00
	4	80	5	12	1,9	2,83
	2	40	6	6	0,4	4,50
	3	60	19	9	0,3	5,50
	2	40	8	6	0,3	1,67
	2	40	17	6	0,1	5,17
	3	60	10	9	0,5	4,67
	3	60	16	9	0,3	5,17
	2	40	16	6	0,2	5,17
	2	40	10	6	0,2	5,00
	2	40	8	6	0,3	4,50
	3	60	9	9	0,6	2,83
	2	40	4	6	0,6	5,17
	2	40	5	6	0,5	4,33
	2	40	6	6	0,4	4,67
	2	40	8	6	0,3	4,67
USE	3	60	10	9	0,5	6,17
	2	40	5	6	0,5	2,00
	3	60	9	9	0,6	2,00
	3	60	6	9	0,9	1,83
	3	60	7	9	0,8	2,50
	4	80	9	12	1,1	3,50
	4	80	17	12	0,6	3,33
	3	60	8	9	0,7	4,83
	3	60	10	9	0,5	1,67
	4	80	14	12	0,7	1,00
	3	60	9	9	0,6	1,17
	2	40	5	6	0,5	2,67
	3	60	6	9	0,9	2,33
	2	40	3	6	0,8	2,17
	3	60	12	9	0,5	4,50
	3	60	9	9	0,6	4,83
	4	80	22	12	0,4	3,33
	3	60	16	9	0,3	4,83

Se realizaron pruebas de Shapiro-Wilk para evaluar la normalidad de las muestras. Utilizamos esta prueba como nuestro medio numérico para evaluar la normalidad porque es más apropiado para muestras pequeñas (<50 muestras).

6.4.1 Eficacia en la Detección de Defectos

Dado que los valores Sig. para las pruebas Shapiro-Wilk fueron 0 para OCLE y 0,001 para USE, estas variables no siguen una distribución normal (<0.05).

Consideramos los datos obtenidos al usar las dos herramientas como grupos independientes. Por lo tanto, la prueba Mann-Whitney U se usó para probar nuestra primera hipótesis nula (H10). En la Fig. 12 muestra el diagrama de caja que contiene datos sobre la eficacia en detectar defectos por herramienta y la Tabla 7 muestra los resultados de la prueba U de Mann-Whitney.

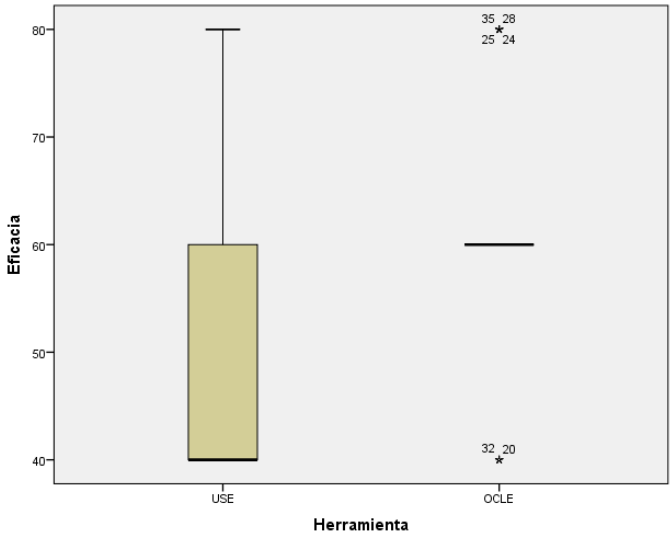


Fig. 12. Diagrama de caja para la Eficacia por herramienta.

Tabla 7. Valores de la prueba Mann-Whitney U.

Test Statistics

	Eficacia
Mann-Whitney U	69,500
Wilcoxon W	240,500
Z	-3,197
Asymp. Sig. (2-tailed)	0,001
Exact Sig. [2*(1-tailed Sig.)]	0,003b

a. Grouping Variable: Tratamiento

b. Not corrected for ties.

A partir de esos resultados, se puede ver que la eficacia de los participantes para detectar los defectos es afectada por el tipo de herramienta. Por lo tanto, rechazamos la hipótesis nula H10. En otras palabras, la eficacia es diferente para cada herramienta, U=69,5, p=0.001<0.05.

6.4.2 Eficiencia en la Detección de Defectos

Como en el análisis anterior, todos los valores Sig. para las pruebas de Shapiro-Wilk fueron 0.401 para OCLE y 0.000 para USE, lo que significa que estas variables no tenían una distribución normal (es decir, <0.05). Considerando ambos tipos datos como grupos independientes, seleccionamos la prueba U de Mann-Whitney (prueba no paramétrica) para evaluar la segunda hipótesis nula (H20). Dado que la eficiencia en la detección de defectos es también afectada por el tipo de herramienta entre OCLE y USE (U = 69.000, p = 0.003 <0.05) (ver Fig. 13), rechazamos la hipótesis H20 (ver Tabla 8).

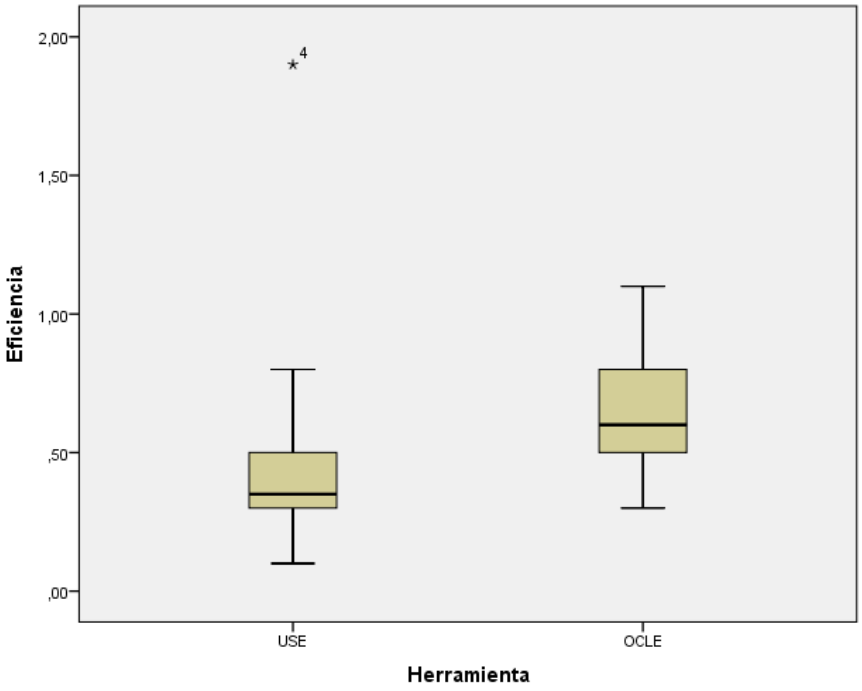


Fig. 13. Diagrama de caja para la Eficiencia por herramienta.

Tabla 8. Valores de la Prueba Mann-Whitney U.

Estadísticos de contraste[a]

	Eficiencia
U de Mann-Whitney	69,000
W de Wilcoxon	240,000
Z	-2,975
Sig. asintót. (bilateral)	,003
Sig. exacta [2*(Sig. unilateral)]	,003[b]

a. Variable de agrupación: Herramienta
b. No corregidos para los empates.

6.4.3 Satisfacción de Usuario de las herramientas

En la PI_3, nuestro objetivo es evaluar si la puntuación de satisfacción de usuario es diferente para las herramientas bajo estudio (USE y OCLE). Para hacer esto, se compara el puntaje general del cuestionario basado en el modelo TAM, obtenido de los dieciocho participantes. La Tabla 6 (columna 7) muestra el puntaje de la satisfacción de usuario.

En Fig. 14 se muestra el diagrama de caja de los datos recopilados para la puntuación de satisfacción de usuario por cada herramienta. Como muestran los resultados, los valores del puntaje de satisfacción de usuario dieron un mejor resultado para la herramienta OCLE que para USE. Como en el análisis anterior (PI_1 y PI_2), se realizó la prueba de Shapiro-Wilk para los valores recopilados de satisfacción de usuario. El valor Sig. fue 0.218 para USE y 0.001 para OCLE, lo que significa que esta variable no tiene una distribución normal (<0.05). Considerando ambos grupos de sujetos independientes, seleccionamos la prueba U de Mann-Whitney (prueba no paramétrica) para evaluar la hipótesis H_{30}.

A partir de estos datos (ver Tabla 9), se puede concluir que el puntaje de satisfacción de usuario de OCLE fue estadísticamente más significativo que el de la herramienta USE, lo que significa que rechazamos la hipótesis nula H_{30} y concluimos que la

satisfacción de usuario percibida por los participantes es diferente para cada herramienta; (U =73.500, p = 0.05).

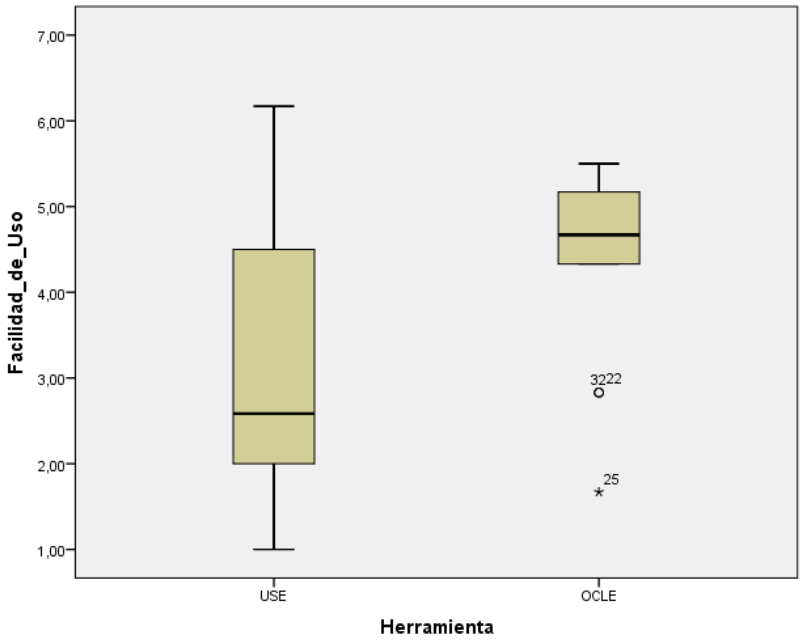


Fig. 14.Diagrama de Caja para la Satisfacción de Usuario por Herramienta.

Tabla 9. Valores de la prueba mann-whitney u para la Satisfacción de Usuario.

Estadísticos de contraste[a]

	Satisfacción de Usuario
U de Mann-Whitney	73,500
W de Wilcoxon	244,500
Z	-2,806
Sig. asintót. (bilateral)	,005
Sig. exacta [2*(Sig. unilateral)]	,004[b]

a. Variable de agrupación: Herramienta

b. No corregidos para los empates.

6.4.4 Discusión Final

A continuación, vamos a discutir e interpretar los hallazgos del experimento empírico de acuerdo a las preguntas de investigación planteadas (sección 6.1.2).

6.4.5 PI_1: ¿Existe una diferencia significativa entre el grado de eficacia para detectar defectos en MC usando USE y usando OCLE?

Desde el experimento empírico, se observó que la eficacia (medida como la razón del número de defectos detectados con respecto a los inyectados) presenta una diferencia significativa cuando se usa OCLE y USE. Esta diferencia está relacionada con el mecanismo y los obstáculos que los participantes encontraron usando ambas herramientas (un poco más en USE). La diferencia se debe a que el mecanismo que tiene cada herramienta para reportar los defectos. En USE los defectos se pueden detectar mediante los mensajes que se presentan en su interfaz gráfica y en la interfaz textual, lo que toma un poco más de tiempo. Mientras que en OCLE los defectos se detectan únicamente utilizando su interfaz gráfica.

6.4.6 PI_2: ¿Cuál de las dos herramientas presenta un mayor grado de eficiencia en detectar la mayor cantidad de defectos en un MC?

A partir del experimento empírico se encontró que, si bien con OCLE se obtuvo una eficiencia ligeramente mayor que con USE, esto no se puede considerar como una diferencia significativa. Lo que nos lleva a inferir que el tiempo requerido para detectar un defecto es similar entre las dos herramientas. Esto puede ser consecuencia del tipo de defecto, ya que se notó claramente que los participantes emplearon más tiempo cuando los defectos implicaban la revisión de restricciones.

6.4.7 PI_3: ¿Cuando los participantes están validando los MC usando las herramientas seleccionadas, su percepción de satisfacción de usuario es impactada?

Algunos participantes destacaron que “los lenguajes usados por las herramientas son intuitivos y fácil de usar”. Así mismo, algunos participantes indicaron que “la información proporcionada por las herramientas fue útil para distinguir el tipo de

defecto y localizarlo en el MC". Mediante el análisis del modelo TAM se determinó que OCLE presenta mejores resultados en cuanto a la satisfacción de usuario de la herramienta, cuando se usa para validar MCs. Esto puede ser porque, para la creación y validación del MC, OCLE usa en mayor grado la interfaz gráfica, en cambio USE requiere escribir código para estas mismas tareas.

6.5 Amenazas a la Validez

En esta sección se describen las principales amenazas a la validez que podrían afectar los resultados de nuestro experimento.

Nuestro trabajo está sujeto a una serie de amenazas: (1) validez interna, que está relacionada con las inferencias que hicimos; (2) validez externa, que analiza la generalización de nuestros hallazgos; y (3) validez de la conclusión se refiere a los problemas que afectan la capacidad de sacar una conclusión correcta.

Respecto a la validez interna, nuestras amenazas se asocian principalmente con los participantes y las mediciones. Primero, los participantes de nuestro experimento podrían tener un conocimiento previo diferente de las herramientas antes de la ejecución del experimento (aunque en el cuestionario demográfico solicitamos explícitamente esta información y todos afirmaron que desconocían las herramientas), por lo que tratamos de minimizar esta amenaza con la capacitación para homogeneizar el conocimiento y experiencia entre ellos). En segundo lugar, con el propósito de garantizar condiciones idénticas para el experimento, se utilizaron ordenadores con las mismas condiciones (hardware y software), material, MC con igual complejidad.

En cuanto a la validez externa, nuestras amenazas están relacionadas con la selección de herramientas de modelado (OCLE y USE), ya que pueden tener características particulares que influyan en el tiempo requerido para la detección de defectos. Esta amenaza ha sido mitigada al seleccionar herramientas con características y funcionalidad similares. Sin embargo, los resultados de este experimento no deben generalizarse a la población de herramientas de modelado,

ya que solo se consideró herramientas de licencia libre. Se propone como trabajo futuro replicar este estudio incorporando herramientas con licencia comercial.

Las amenazas de validez de conclusión fueron mitigadas por el diseño del experimento. Con respecto a los sujetos reclutados, tomamos como muestra un grupo de 18 estudiantes de la carrera de Ingeniería de Sistemas. Además, se realizaron pruebas adecuadas para rechazar estadísticamente la hipótesis nula. Las métricas utilizadas nos permitieron evaluar de manera objetiva la eficacia, eficiencia y la satisfacción de usuario de los sujetos. Como la validez de la conclusión podría verse afectada por el número de observaciones, se requieren réplicas adicionales con un conjunto de datos más grande para confirmar o contradecir los resultados logrados. Respecto al cuestionario para medir la satisfacción de usuario percibida, fue diseñado utilizando preguntas y escalas estándares que tiene un alto grado de fiabilidad demostrada [31].

CAPÍTULO 7: CONCLUSIONES Y TRABAJO FUTURO

En este trabajo de titulación se realizó una evaluación empírica y comparación teórica de herramientas usadas para la validación de requisitos funcionales a nivel de modelos conceptuales basados en UML. Para este objetivo, se emplearon dos enfoques de investigación para comparar herramientas que permiten validar requisitos en MCs basados en diagramas de clases UML. Primero se realizó un análisis teórico de las características de algunas herramientas que permiten validar un MC, para este análisis se tomaron en cuenta varios criterios entre ellos que sean herramientas de licencia libre y que soporten la creación y ejecución de casos de pruebas para validar MCs. El segundo enfoque de investigación que se aplicó en este estudio fue realizar una evaluación empírica para comparar la eficacia, eficiencia y la satisfacción de usuario que perciben los usuarios de USE y OCLE.

7.1 Conclusiones

- De la comparación teórica se concluye que actualmente existen varias herramientas que permiten editar MCs, cada una con diferentes funcionalidades y formas de acceso a la herramienta. Las mencionadas en este trabajo verifican la sintaxis de los MCs y permiten realizar la validación de requisitos a nivel de modelos. En cada una se analizó la forma de modelar y editar MCs, el tipo de validaciones que soportan, los lenguajes de restricciones compatibles, soporte de plataformas y la forma de ejecución de los casos de pruebas.

  Las herramientas USE y OCLE son las que obtuvieron un mejor puntaje de la comparación teórica realizada en este trabajo. Sin embargo, ambas herramientas presentan características similares para los criterios que se usaron en esta comparación (análisis de dominio, verificación, validación, calidad y ejecución).

- A partir de la evaluación experimental, se concluye que el experimento se lo realizó sin mayores inconvenientes. La selección de participantes fue apropiada para la ejecución del experimento empírico (estudiantes de la Universidad de Cuenca), quienes están lo suficientemente preparados para el uso de este tipo de herramientas de modelado y validación UML. El uso de los laboratorios de cómputo de la Universidad de Cuenca fue de gran ayuda ya que en los equipos se pudo instalar las herramientas y alojar el contenido preparado para las sesiones (presentación, ejemplos prácticos, cuestionarios).
- Las métricas planteadas en el experimento empírica fueron evaluadas usando el método experimental GQM. En cuanto a la eficacia se determina que OCLE presenta mejor puntuación por aprovechar el uso de su interfaz gráfica en la detección de defectos. En cuanto a la eficiencia de las herramientas los resultados fueron similares y en base a esto podemos decir que el tiempo en detectar defectos en cada herramienta se asemeja. Por último, para la satisfacción de usuario con los resultados se determinó que OCLE presentan mayor puntaje por lo que aprovecha en mayor gado su interfaz gráfica ya sea para el modelado como para la validación, mientras que USE se basa en escribir código tanto para el modelado como para la validación de MCs.

Las siguientes conclusiones se basan en el logro de los objetivos planteados en este trabajo de titulación, para lo cual se justifica mediante información que ha sido evidenciada en este trabajo, como se describen a continuación:

1. Elaborar el marco teórico necesario para la realización del trabajo de titulación.

Este objetivo se cumplió totalmente, ya que se resumió los fundamentos de nuestro marco teórico (ver Capítulo 3), que son muy importantes porque establecen los criterios utilizados para la comparación teórica y empírica abordada en este trabajo.

2. Realizar el estado arte acerca de las herramientas usadas para la validación de requisitos funcionales a nivel de modelos conceptuales basados en UML.

El objetivo fue cumplido y para conseguirlo se resumió el estado de arte acerca de las herramientas disponibles para la validación de requisitos funcionales a nivel de modelos conceptuales basados en UML. Adicionalmente, se consideraron trabajos relacionados de evaluaciones de herramientas usadas para la validación de MC basados en UML. Esta revisión de la literatura es también una contribución de conocimiento existente para justificar la oportunidad de investigación del estudio.

3. Identificar dos herramientas adecuadas para la validación de modelos conceptuales basados en UML, evaluarlas, y compararlas.

Para cumplir con este objetivo inicialmente se recolectaron varias herramientas relacionadas a la validación de los MCs. En base a éstas, se preseleccionaron las herramientas que cumplían con el objetivo de investigación propuesto en este trabajo y como proceso final luego de haber analizado las herramientas se procedió a seleccionar dos de ellas las cuales cumplen completamente nuestro objetivo para posteriormente evaluarlas y compararlas teóricamente.

4. Evaluar empíricamente las herramientas seleccionadas en cuanto a su eficacia, eficiencia y satisfacción de usuario

Este objetivo se cumplió realizando la evaluación experimental, la cual reportó que hay diferencias notables con respecto a la eficacia y satisfacción de usuario entre USE y OCLE, alcanzando un mejor puntaje la herramienta OCLE. Sin embargo, con respecto a la eficiencia no se encontró diferencias significativas, esto debido a sus características muy similares que tienen estas herramientas para el soporte a la detección de defectos en MCs.

7.2 Trabajo Futuro

A continuación, se plantean actividades que se podrían realizar para ampliar el estudio que se ha realizado en este trabajo de titulación.

- Extender el estudio considerando otras herramientas de modelado UML (no solo herramientas de libre licencia sino también herramientas comerciales como MagicDraw, Visual Paradigm, ALTOVA UModel) y además que integren

otros diagramas UML para representar un modelo conceptual (diagramas de actividad, de secuencia), para ver cómo los resultados de este estudio pueden ser generalizados.

- Proponer preguntas de investigación enfocadas a una sola herramienta de validación para centrarse en detalles más concretos (usabilidad, rendimiento, etc.) y conocer las carencias o ventajas que se podrían extender en la herramienta.
- Luego del análisis de herramientas de licencia libre y comercial, implementar una nueva herramienta de validación con licencia libre que integre las funciones más destacadas de las diferentes herramientas analizadas, y solvente las carencias que se han identificado en esas herramientas.

REFERENCIAS

[1] L. P. Ayabaca and S. Moscoso Bernal, “Verificación y Validación de Software Software Verification and Validation,” *Rev. Kill. Técnica*, vol. 1, no. 3, pp. 25–32, Dec. 2017.

[2] A. Kraus, “Model Driven Software Engineering for Web Applications,” Ludwig-Maximilians-Universität München, 2007.

[3] O. M. Group, “Object Management Group Business Process Model and Notation BPMN,” 2018. [Online]. Available: http://www.bpmn.org/. [Accessed: 11-Mar-2018].

[4] OMG Object Management Group, “An OMG ® Unified Modeling Language ® Publication OMG ® Unified Modeling Language ® (OMG UML ®) OMG Document Number: Date,” no. December, p. 796, 2017.

[5] B. Dobing and J. Parsons, “How UML is used,” *Commun. ACM*, vol. 49, no. 5, pp. 109–113, 2006.

[6] M. F. Granda, “Testing-Based Conceptual Schema Validation in a Model- Driven Environment,” in *CAiSE 2013 Doctoral Consortium*, 2013.

[7] A. Olivé and A. Tort, “Testing Conceptual Schema Satisfiability,” *Intentional Perspect. Inf. Syst. Eng.*, pp. 277–288, 2010.

[8] M. F. Granda, “Testing-based conceptual schema validation in a model- Driven environment,” *CEUR Workshop Proc.*, vol. 1, 2013.

[9] OMG-OCL, “Object Constraint Language,” *Int. Bus.*, no. May, pp. 58–90, 2014.

[10] R. Van Solingen and E. Berghout, *The Goal / Question / Metric Method - A Practical Guide for Quality Improvement of Software Development*, no. October 2015. McGraw-Hill Publishing Company, 1999.

[11] A. Tort and A. Olivé, “An approach to testing conceptual schemas,” *Data Knowl. Eng.*, vol. 69, no. 6, pp. 598–618, 2010.

[12] R. Prieto, L. Susana, and A. García, “Mejores Prácticas Para El Establecimiento Y Aseguramiento De La Calidad Del Software,” *Unidad Multidiscip. CIET*, p. 130, 2008.

[13] M. Cristiá, “Introducción a la Ingeniería de Requerimientos,” Santa Fe, pp. 1–58, 2011.

[14] A. Tort, A. Olivé, and M. R. Sancho, “An approach to test-driven development of conceptual schemas,” *Data Knowl. Eng.*, vol. 70, no. 12, pp. 1088–1111, 2011.

[15] L. C. Briand, “Empirical Evaluation in Software Engineering: Role, Strategy, and Limitations,” in *Empirical Software Engineering Issues. Critical Assessment and Future Directions: International Workshop, Dagstuhl Castle, Germany, June 26-30, 2006. Revised Papers*, V. R. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, and R. W. Selby, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, p. 21.

[16] J. Goodwin, *Research in Psychology Methods and Design*, 6th ed. Estados Unidos: R.R. Donnelley & Sons, Inc., 2010.

[17] K. Hussey, R. Faudou, D. Sciamma, and P. Hofer, "MDT / Papyrus-Proposal - Eclipsepedia," *2008-10-08*, 2008. [Online]. Available: https://wiki.eclipse.org/MDT/Papyrus-Proposal. [Accessed: 24-Oct-2018].

[18] Database Systems Group, "USE A UML based Specification Environment." Bremen, pp. 1–93, 2007.

[19] M. C. Otero, "Evaluación empírica de la comprensión del modelado dinámico en los lenguajes UML y OML de aplicaciones software," *Lenguajes y Sist. Informáticos*, p. 310, 2003.

[20] L. Yu, R. B. France, I. Ray, and K. Lano, "A light-weight static approach to analyzing UML behavioral properties," *Proc. IEEE Int. Conf. Eng. Complex Comput. Syst. ICECCS*, no. Iceccs, pp. 56–63, 2007.

[21] L. Aladib, "Case Study Object Constraints Language (OCL) Tools," Kuala Lumpur, 2014.

[22] S. Safdar Aqeel, I. Muhammad Zohaib, and K. Muhammad Uzair, "Empirical evaluation of uml modeling tools–a controlled experiment," in *Modelling Foundations and Applications. ECMFA 2015. Lecture Notes in Computer Science*, 2015, vol. 9153, pp. 33–44.

[23] E. Planas and J. Cabot, "How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus," *Comput. Stand. Interfaces*, vol. 67, no. October 2018, p. 103363, 2020.

[24] A. Bobkowska and K. Reszke, "Usability of UML Modeling Tools," *Proc. 2005 Conf. Softw. Eng. Evol. Emerg. Technol.*, vol. 130, p. 75, 2006.

[25] "OCLE 2.0 - Object Constraint Language Environment," 2017. [Online]. Available: http://lci.cs.ubbcluj.ro/ocle/overview.htm. [Accessed: 15-Mar-2020].

[26] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*, vol. 9783642290442. Springer-Verlag Berlin Heidelberg, 2012.

[27] N. Juristo and A. M. Moreno, "Introduction," in *Basics of Software Engineering Experimentation*, Boston, MA: Springer US, 2001, pp. 3–22.

[28] S. Eldh, H. Hansson, S. Punnekkat, A. Pettersson, and D. Sundmark, "A framework for comparing efficiency, effectiveness and applicability of software testing techniques," *Proc. - Test. Acad. Ind. Conf. - Pract. Res. Tech. TAIC PART 2006*, pp. 159–170, 2006.

[29] J. C. Sánchez Prieto, S. Olmos Migueláñez, and F. J. García Peñalvo, "EVALUACIÓN DE LA ACEPTACIÓN DE LAS TECNOLOGÍAS MÓVILES EN LOS ESTUDIANTES DEL GRADO DE MAESTRO," pp. 1649–1659.

[30] L. A. Yong Varela, L. A. Rivas Tovar, and J. Chaparro, "Modelo de aceptación tecnológica (TAM): Un estudio de la influencia de la cultura nacional y del perfil del usuario en el uso de las TIC," *Innovar*, vol. 20, p. 36, Jan. 2010.

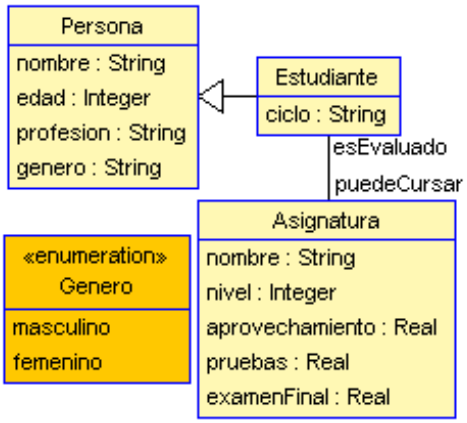
[31] J. Sauro and J. Lewis R., *Quantifying the User Experience*. Waltham: Elsevier Inc., 2012.

APENDICES

Apéndice A

Cuestionario para evaluar la eficacia y eficiencia en USE.

EJEMPLO PRÁCTICO



TAREAS

Por favor realice lo siguiente:

1. Revisar si las especificaciones de los requisitos se cumplen en el código del modelo. Anotar

   Hora y minuto inicial: _____Hora y minuto final: _____.

2. Ejecutar los casos de prueba e indicar si los casos de prueba permiten encontrar algún defecto.

Inicio HH:MM:SS	**Fin HH:MM:SS**	**Defecto encontrado S/N**	**Defecto Corregido S/N**	**Descripción del defecto**	**¿La herramienta proporcionó ayuda? S/N**

3. Cambie el código del modelo para que verifique como una restricción que el puntaje total sea la suma del aprovechamiento más el examen final
   - Marque hora y minuto inicial: HH:MM:SS
   - Marque hora y minuto final: HH:MM:SS
   - Marque del 1 al 5 si fue fácil realizar el cambio en el código (1=fácil, 5=complicado) ____.

- Explique su respuesta (eje: el lenguaje no es intuitivo)

  _____.
- La herramienta le ayudó a encontrar defectos (Si/NO) _____.
- Si su respuesta fue Si, ¿Cómo le ayudó (ejemplo el parser, o mostrando el error de sintaxis)?

  _____.

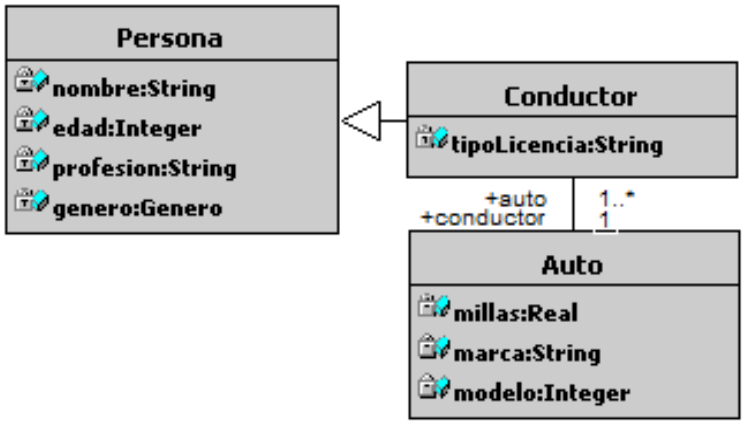
4. Trate de crear un caso de prueba para verificar si se puede validar la nota final
   - Marque hora y minuto inicial: _____Marque hora y minuto final: _____.
   - Marque del 1 al 5 si fue fácil realizar el caso de prueba (1=fácil, 5=complicado) 1
   - ¿Logró detectar el defecto? (Si/No) ____.
   - ¿La herramienta le proporcionó información que le ayuda a localizar los defectos?_(Si/NO) ¿Qué información le ayudó?

     _____.

Apéndice B

Cuestionario para evaluar la eficacia y eficiencia en OCLE.

EJEMPLO PRÁCTICO



TAREAS

Por favor realice lo siguiente:

1. Revisar si las especificaciones de los requisitos se cumplen en el código del modelo. Anotar

   Hora y minuto inicial: _____Hora y minuto final: _____.

2. Ejecutar los casos de prueba e indicar si los casos de prueba permiten encontrar algún defecto.

Inicio HH:MM:SS	Fin HH:MM:SS	Defecto encontrado S/N	Defecto Corregido S/N	Descripción del defecto	¿La herramienta proporcionó ayuda? S/N

3. Cambie el código del modelo para que verifique como una restricción que el puntaje total sea la suma del aprovechamiento más el examen final
   - Marque hora y minuto inicial: HH:MM:SS
   - Marque hora y minuto final: HH:MM:SS
   - Marque del 1 al 5 si fue fácil realizar el cambio en el código (1=fácil, 5=complicado) ____.
   - Explique su respuesta (eje: el lenguaje no es intuitivo)

     _____.
   - La herramienta le ayudó a encontrar defectos (Si/NO) _____.
   - Si su respuesta fue Si, ¿Cómo le ayudó (ejemplo el parser, o mostrando el error de sintaxis)?

     _____.

4. Trate de crear un caso de prueba para verificar si se puede validar la nota final

- Marque hora y minuto inicial: _____Marque hora y minuto final: _____.
- Marque del 1 al 5 si fue fácil realizar el caso de prueba (1=fácil, 5=complicado) 1
- ¿Logró detectar el defecto? (Si/No) ____.
- ¿La herramienta le proporcionó información que le ayuda a localizar los defectos?_(Si/NO)
  ¿Qué información le ayudó?
  _____.

Apéndice C

Cuestionario para evaluar la Satisfacción de Usuario en USE.

Preguntas	Puntuación						
	1	2	3	4	5	6	7
¿Aprender a utilizar USE sería fácil para mí?							
¿Mi interacción con USE sería clara y entendible?							
¿Encuentro a USE flexible para interactuar con él?							
¿Sería fácil para mi llegar a ser un experto en el uso de USE?							
¿Sería fácil para mi conseguir con USE lo que quiero hacer?							
¿Siento que mi capacidad de determinar la facilidad de uso del USE está limitada por mi falta de experiencia?							

Apéndice D

Cuestionario para evaluar la Satisfacción de Usuario en OCLE.

Preguntas	Puntuación						
	1	2	3	4	5	6	7
¿Aprender a utilizar OCLE sería fácil para mí?							
¿Mi interacción con OCLE sería clara y entendible?							
¿Encuentro a OCLE flexible para interactuar con él?							
¿Sería fácil para mi llegar a ser un experto en el uso de OCLE?							
¿Sería fácil para mi conseguir con OCLE lo que quiero hacer?							
¿Siento que mi capacidad de determinar la facilidad de uso del OCLE está limitada por mi falta de experiencia?							

Apéndice E

Captura de parte del artículo académico realizado en base al trabajo de titulación.

¿Cómo pueden validarse los diagramas de Clases UML? Una Evaluación Empírica y Teórica de dos herramientas UML: USE y OCLE

Carlos Vera-Mejia
Departamento de Ciencias de la Computación
Universidad de Cuenca
Cuenca, Ecuador
carlos.veram90@ucuenca.edu.ec

Maria Fernanda Granda
Departamento de Ciencias de la Computación
Universidad de Cuenca
Cuenca, Ecuador
fernanda.granda@ucuenca.edu.ec

Otto Parra
Departamento de Ciencias de la Computación
Universidad de Cuenca
Cuenca, Ecuador
otto.parra@ucuenca.edu.ec

***Resumen*— La validación de modelos conceptuales es una actividad clave para asegurar la calidad de software y ahorrar costos, especialmente cuando se adopta cualquier tipo de metodología de Ingeniería de Software Dirigido por Modelos. En este contexto, los lenguajes de modelado estándar, como el Lenguaje de Modelado Unificado -UML, y las herramientas para apoyar las actividades de validación se vuelven esenciales. El objetivo de este estudio es analizar qué tan buenas son las herramientas de modelado para apoyar la tarea de validación basada en pruebas. Nuestro objetivo es extraer las principales características de las herramientas que soportan este tipo de validación, de manera que en este estudio se comparen herramientas que ayuden a mejorar el proceso de detección temprana de defectos en el modelado conceptual. Nuestro estudio emplea dos enfoques de investigación: (1) el principal es una evaluación empírica que analiza eficacia, eficiencia y usabilidad de dos herramientas seleccionadas que permiten validar requisitos en modelos conceptuales UML y (2) un análisis teórico que complementa el experimento anterior. El estudio se enfoca en el tipo más frecuente de diagrama UML, el diagrama de clase y en dos herramientas ampliamente utilizadas por la comunidad de modelado para realizar una validación basada en pruebas: USE y OCLE.**

Palabras Claves*—*Validación, UML, diagramas de clases, USE, OCLE, experimento controlado.

I. INTRODUCCIÓN

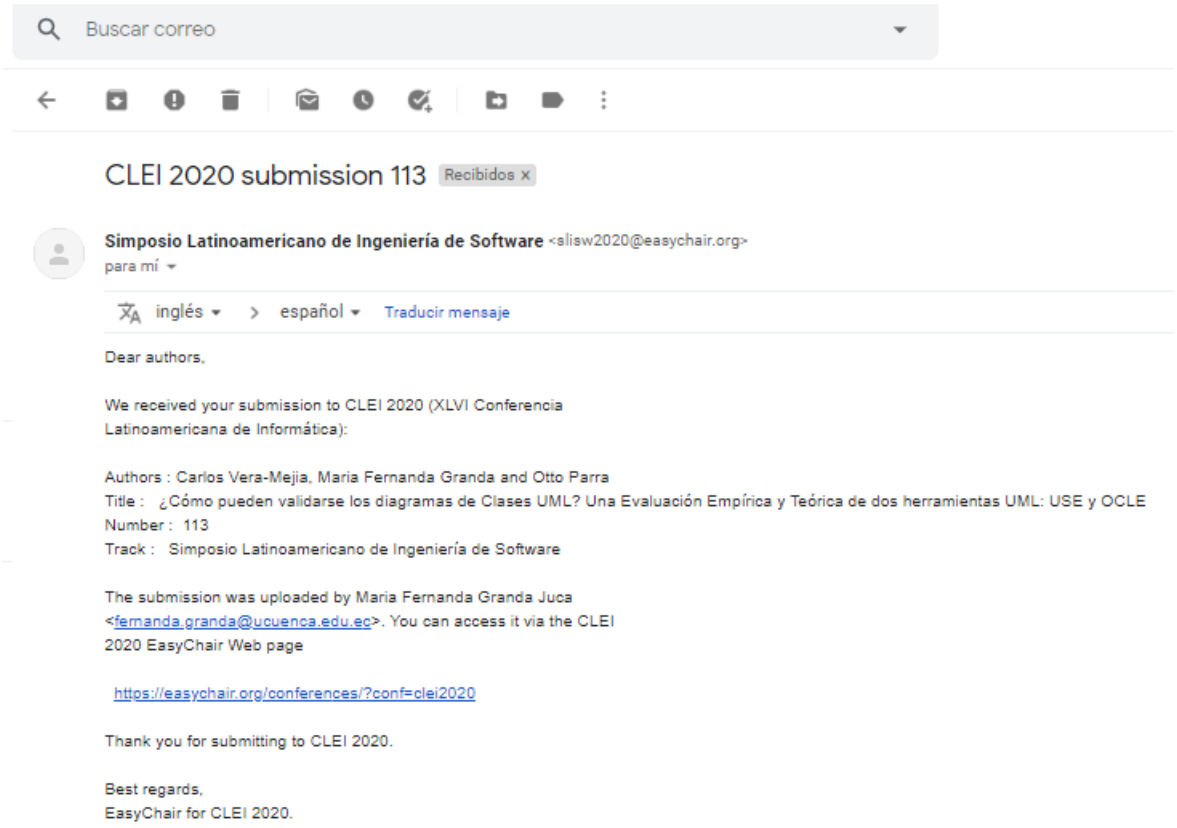
Dentro de las etapas del ciclo de vida de desarrollo de software se encuentran las pruebas de software [1]. Estas son muy importantes ya que permiten encontrar defectos en un producto de software. Los defectos tienen un gran impacto en el presupuesto del software, por lo que es muy importante tratar de detectarlas y reducirlas en las etapas tempranas del proceso de desarrollo del software, por ejemplo, a nivel de modelo basados en Business Process Model and Notation (BPMN) [3], etc. Por el lado del lenguaje, UML es el estándar de facto para el modelado de sistemas de software. UML[4] proporciona varios diagramas para modelar la estructura de un sistema de software, su arquitectura y su comportamiento.

Una gran cantidad de herramientas comerciales y de código abierto están disponibles para admitir el modelado UML, p. ej., MagicDraw, Papyrus, ArgoUML, Modelio, StarUML, USE, OCLE entre muchas otras[1]. Si bien las características de cada herramienta difieren, todas ofrecen un editor gráfico para asistir en la definición de modelos UML y facilidades para la verificación del modelo creado. Sin embargo, nosotros creemos que hace falta una comparación teórica y empírica que ayude a seleccionar una herramienta que permita validar un MC basado en UML. En la literatura relacionada, se han encontrado muchos estudios que reportan comparaciones para determinar algunos aspectos como, por ejemplo: (a) qué tan útiles son las herramientas de modelado UML [11-15] en base a las facilidades que presta para modelar, (b) la usabilidad percibida [5], entre otras características. Nuestro estudio agrega una nueva perspectiva a estos trabajos anteriores, al realizar una evaluación empírica y teórica de herramientas, que permitan detectar defectos a través de la ejecución de scripts de prueba sobre MC usando UML. Para este propósito, este trabajo se centra en el diagrama de UML más utilizado, el diagrama de clase [6] y en herramientas de modelado con licencia libre que permiten automatizar la ejecución de scripts de prueba.

Según Meyer: "Probar un programa (en nuestro caso un modelo conceptual) es intentar hacerlo fallar, y el objetivo de las pruebas es descubrir fallas inyectando defectos en el artefacto del software "[7], [8].

Apéndice F

Carga de Articulo en evento CLEI 2020



Buscar correo

CLEI 2020 submission 113 Recibidos ×

Simposio Latinoamericano de Ingeniería de Software <slisw2020@easychair.org>
para mí

inglés > español Traducir mensaje

Dear authors,

We received your submission to CLEI 2020 (XLVI Conferencia Latinoamericana de Informática):

Authors : Carlos Vera-Mejia, Maria Fernanda Granda and Otto Parra
Title : ¿Cómo pueden validarse los diagramas de Clases UML? Una Evaluación Empírica y Teórica de dos herramientas UML: USE y OCLE
Number : 113
Track : Simposio Latinoamericano de Ingeniería de Software

The submission was uploaded by Maria Fernanda Granda Juca <fernanda.granda@ucuenca.edu.ec>. You can access it via the CLEI 2020 EasyChair Web page

https://easychair.org/conferences/?conf=clei2020

Thank you for submitting to CLEI 2020.

Best regards,
EasyChair for CLEI 2020.