UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Revista Científica

http://revistas.udistrital.edu.co/ojs/index.php/revcie/index

CIDC

Artículo de revisión

Faltas en el aprendizaje del modelado de clases y casos de uso: una revisión sistemática

Learning Failures in Class Modeling and Use Cases: A Systematic Review

Falhas de aprendizagem em modelagem de classes e casos de uso: uma revisão sistemática

Juan-Pablo Ucán-Pech[1]
Raúl-Antonio Aguilar-Vera[2]
Julio-César Díaz-Mendoza[3]
Omar-Salvador Gómez-Gómez[4]

Recibido: julio de 2022 **Aceptado:** octubre de 2022

Para citar este artículo: Ucán-Pech, J. P., Aguilar-Vera, R. A., Díaz-Mendoza, J. C. y Gómez-Gómez, O. S. (2023). Faltas en el aprendizaje del modelado de clases y casos de uso: una revisión sistemática. *Revista Científica*, *46*(1), 93-106. https://doi.org/10.14483/23448350.19655

Resumen

En este artículo se presenta una revisión de los estudios primarios que abordan la identificación de faltas durante el aprendizaje de los diagramas de casos de uso (DCU) y de los diagramas de clase (DC) en los últimos 10 años. Este trabajo es el inicio de un proyecto de investigación relacionado con la detección de faltas en los diagramas UML. Este artículo presenta un análisis del estado del arte con respecto a la tipificación de faltas en DCU y DC, con el objetivo de identificar oportunidades y brechas de investigación. Se encontraron 20 documentos de acuerdo con los criterios de inclusión y exclusión establecidos mediante la metodología utilizada para la revisión sistemática de literatura. Considerando la relevancia del tema, se puede observar que es limitada la investigación relacionada con la detección de faltas en los diagramas UML tanto en DCU como DC.

Palabras clave: diagramas de casos de uso; diagramas de clase; faltas en el modelado de software; RSL; UML.

Abstract

This paper presents a review of the main studies that address the identification of faults during the learning of use case diagrams (DCU) and class diagrams (DC) in the last 10 years. This work is the beginning of a research project related to the detection of faults in UML diagrams. This paper presents an analysis of the state of the art regarding the typification of faults in DCU and DC, with the objective of identifying opportunities and research gaps. 20 documents were found according to the inclusion and exclusion

1. Ph. D. Universidad Autónoma de Yucatán (Mérida, México). juan.ucan@correo.uady.mx.
2. Ph. D. Universidad Autónoma de Yucatán (Mérida, México). avera@correo.uady.mx.
3. Universidad Autónoma de Yucatán (Mérida, México). julio.diaz@correo.uady.mx.
4. Escuela Superior Politécnica de Chimborazo (Riobamba, Ecuador). ogomez@espoch.edu.ec.

criteria established through the methodology employed for the systematic review of the literature. Considering the relevance of the topic, it can be observed that there is limited research on fault detection in UML diagrams for both DCU and DC.
Keywords: class diagrams; faults in software modeling; SLR; UML; use case diagrams.

Resumo

Neste artigo, apresenta uma revisão dos estudos primários que aborda a identificação de faltas durante o aprendizado dos diagramas de casos de uso (DCU) e dos diagramas de classe (DC), nos últimos 10 anos. Este trabalho é o início de um projeto de investigação relacionado com a detecção de faltas nos diagramas UML, especificamente neste artigo, apresentando a análise do estado da arte com a tipificação de faltas nos DCU e DC com o objetivo de identificar oportunidades e brechas de investigação. De acuerdo com os critérios de inclusão e exclusão estabelecidos através da metodologia da revisão sistemática da literatura utilizada, se encontrar 20 documentos. Considerando a relevância do tema, pode-se observar que é limitada a investigação relacionada à detecção de faltas nos diagramas UML tanto em DCU como DC.
Palavras-chaves: diagramas de casos de uso; diagramas de classes; falhas na modelagem de software; RSL; UML.

Introducción

De acuerdo con Aguilar *et al*. (2017), se entiende por diseño al proceso de definición de arquitectura, componentes, interfaces y otras características del sistema de software, así como del producto de dicho proceso.

En el contexto del paradigma orientado a objetos surgido en la década de 1980, se desarrollaron métodos, modelos y técnicas que permiten generar representaciones relacionadas con la estructura y el comportamiento del software a construir. Años más tarde, el esfuerzo por la unificación de dichos métodos y modelos dio origen a un estándar de diseño en la ingeniería de software que hoy se conoce como lenguaje modelado unificado (UML por su sigla en inglés) (Rumbaugh, Jacobson y Booch, 2000).

Como parte del proceso de verificación, es necesario realizar una serie de revisiones al diseño, y de acuerdo con Brunet, Guerrero y Figueiredo (2009), este proceso consiste en un mecanismo para encontrar faltas en el diseño y su representación. En este sentido, proponer una técnica para encontrar faltas, específicamente en el modelado del diseño de software, resulta un área de investigación que puede aportar elementos innovadores en la ingeniería de software.

Para efecto de distinguir entre cuatro conceptos muy relacionados que suelen ser utilizados de manera indistinta, en el estudio se consideró la propuesta de Juristo, Moreno y Vegas (2006), la cual considera:

- Error: acción humana que produce una falta.
- Falta: algo que está mal en un producto (modelo, código, documento, etc.).
- Fallo: manifestación de una falta.
- Defecto: error, falta o fallo.

Esta investigación es el inicio de un proyecto relacionado con la detección de faltas en los diagramas UML, en este artículo se presenta específicamente una revisión de los estudios primarios publicados para la identificación de faltas durante el aprendizaje de los diagramas de casos de uso y diagramas de clase en los últimos 10 años.

El presente artículo se encuentra organizado de la siguiente manera: la segunda sección presenta conceptos básicos sobre el diseño de software y el lenguaje de modelado unificado; en la tercera sección se presentan los trabajos relacionados; la cuarta sección describe la metodología de investigación utilizada para la revisión sistemática de la literatura (RSL); la quinta sección describe la planificación del estudio basada en la metodología de la sección anterior; en la sexta sección está la ejecución de dicho estudio; y la séptima sección muestra los resultados obtenidos de la RSL.

Finalmente, la octava sección refiere las conclusiones y los trabajos futuros.

Modelado del software

Una de las fases del desarrollo de un producto de software es el diseño, este permite modelar el sistema o software a construir; esta fase comienza una vez que se han analizado y especificado los requisitos, y se realiza antes de la fase de codificación (Figura 1).

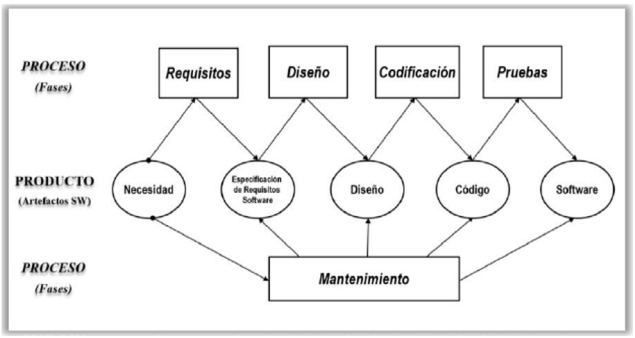

Figura 1. Dualidad Proceso-Producto en la Ingeniería de Software.
Fuente: Aguilar, Oktaba y Juárez (2019).

Los modelos derivados del diseño pueden analizarse desde dos dimensiones distintas: dimensión del proceso y dimensión de la abstracción (Figura 2). La dimensión del proceso indica la evolución del modelo del diseño conforme se ejecutan las tareas de este como parte del proceso del software; por su parte, la dimensión de la abstracción representa el nivel de detalle a medida que cada elemento del modelo de análisis se transforma en un equivalente de diseño y luego se mejora en forma iterativa. La Figura 2 permite observar una línea discontinua que representa la frontera entre los elementos del modelo de análisis y del modelo de diseño. Así mismo, los elementos del modelo de diseño usan muchos de los diagramas UML que se utilizaron en el modelo del análisis (Pressman y Maxim, 2021).

En los últimos años el lenguaje de modelado unificado se ha consolidado como una de las mejores herramientas de modelado con el que se han establecido los procesos del diseño del software. El UML nos permite presentar los aspectos muy variados de un sistema de software, por ejemplo:

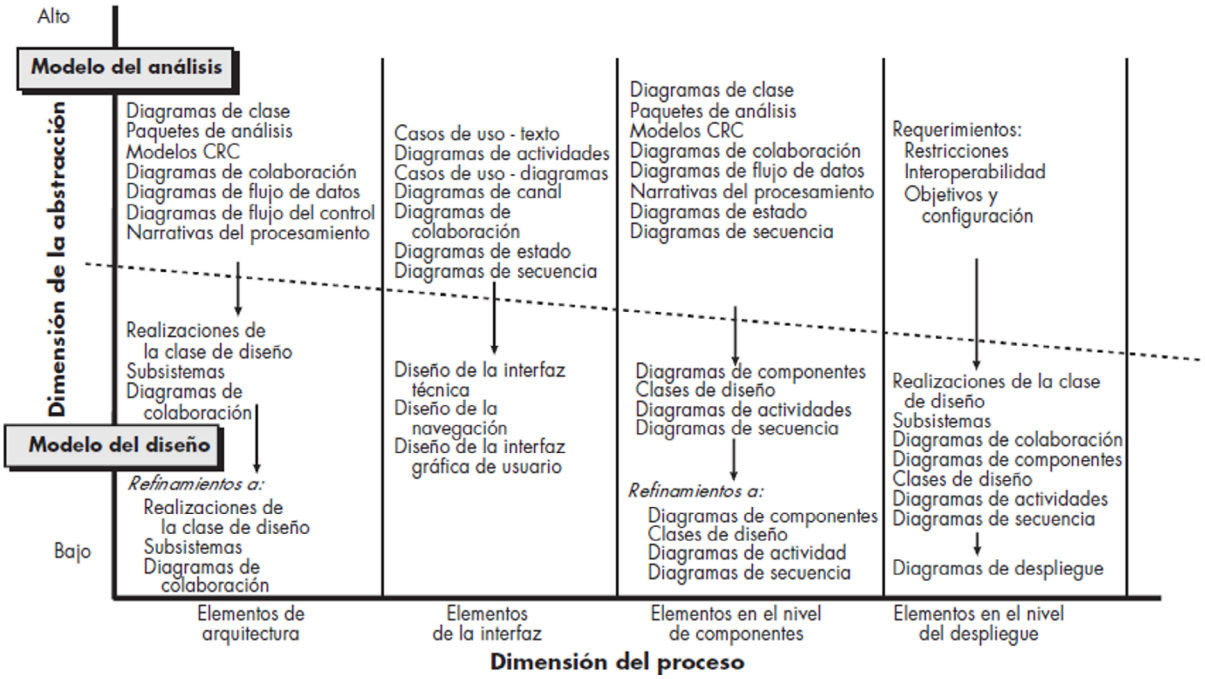

Figura 2. Dimensiones del modelo de diseño.
Fuente: Pressman y Maxim (2021).

requisitos, estructuras de datos, flujos de datos y flujos de información, lo anterior utilizando conceptos orientados a objetos (Seidl, 2015).

Cabe destacar que el UML permite modelar la gran mayoría de los diagramas encontrados tanto de la dimensión del proceso como en la dimensión de abstracción. En la Tabla 1 se presentan los diagramas de este lenguaje de modelado agrupado por áreas y vistas, así como los conceptos principales de cada uno de los diagramas.

Trabajos relacionados

Para el desarrollo de esta revisión se identificaron y analizaron tres estudios previos, en particular dos estudios secundarios, a continuación, se listan los documentos encontrados.

Alanazi (2009) propone quince reglas básicas que ayudan a los desarrolladores a construir diagramas UML correctos y consistentes, de esas quince reglas, las primeras dos son para diagramas de casos de uso y alrededor de seis para los diagramas de clase.

Por otra parte, Torre *et al*. (2018) proporcionan un resumen de las técnicas de síntesis de UML descritas en la literatura sin restricción de inicio, pero con límite hasta octubre de 2013, el objetivo fue obtener una visión general, amplia y detallada, de la investigación actual en esta área, como método usaron un estudio de mapeo sistemático de Kitchenham y Charters (2007), y como resultado de dicho estudio proponen un resumen de reglas de consistencia UML.

De forma similar, Shaikh *et al*. (2021) presentan una visión general de aproximaciones de verificación del modelado de clases UML, e identifican problemas abiertos, tendencias de investigación actuales y otras áreas de mejora. Como método

Tabla 1. Vistas y diagramas UML

Área principal	Vista	Diagrama	Conceptos principales
Estructural	Vista estática	Diagrama de clases	Asociación, clase, dependencia, generalización, interfaz, realización
	Vista de diseño	Estructura interna	Conector, interfaz, interfaz obligatoria, interfaz proporcionada, parte, puerto
		Diagrama de colaboración	Colaboración, conector, rol, uso de la colaboración
		Diagrama de componentes	Componente, dependencia, interfaz proporcionada, interfaz obligatoria, puerto, realización, subsistema
	Vista de casos de uso	Diagrama de casos de uso	Actor, asociación de caso de uso, extensión, generalización de casos de uso, inclusión
Dinámica	Vista de máquina de estado	Diagrama de máquina de estados	Actividad hacer, disparador, efecto, estado, evento, región, transición, de finalización
	Vista de actividad	Diagrama de actividad	Acción, actividad, control de flujo, división, excepción, flujo de datos, nodo de control, nodo objeto, pin, región expansión
	Vista de interacción	Diagrama de secuencia	Especificación de la ejecución, especificación del suceso, fragmento de la interacción, línea de vida, mensaje, operando de la interacción, señal
		Diagrama de comunicación	Colaboración, condición de guarda, mensaje, rol, número de secuencia
Física	Vista de despliegue	Diagrama de despliegue	Artefacto, dependencia, manifestación, nodo
Gestión del modelo	Vista de gestión del proyecto	Diagrama de paquetes	Importar, nodo, paquete
	Perfil	Diagrama de paquetes	Estereotipo, perfil, restricción, valor etiquetado

Fuente: Rumbaugh et al. (2000)

usaron una revisión sistemática de la literatura también de Kitchenham y Charters (2007); los documentos considerados en el estudio fueron de los años 1997 a 2020 y el estudio estuvo limitado solo a diagramas de clases.

Metodología

En el contexto de los estudios secundarios, una revisión sistemática de la literatura (RSL) permite identificar, evaluar e interpretar la investigación disponible en la literatura, que resulta relevante para una pregunta de investigación, un área temática o un fenómeno de interés (Genero, Cruz-Lemus y Piattini, 2014).

En el caso del presente estudio, los autores utilizaron la guía propuesta por Kitchenham y Charters (2007), cuyo desarrollo consiste en tres etapas:

A. Planificación: las principales tareas de esta primera etapa consisten en: identificar la necesidad de la revisión, formular las preguntas de investigación, definir el protocolo de la revisión, y validar el protocolo de la revisión.
B. Ejecución: la segunda etapa tiene como principales tareas: identificar la investigación relevante, seleccionar los estudios primarios relevantes para el estudio, evaluar la calidad de los estudios primarios seleccionados, extraer la información vinculada con las preguntas de investigación y sintetizar la información extraída.
C. Difusión: una vez realizada la RSL, las tareas que integran la tercera etapa consisten: redactar el informe de la revisión, validar el informe y, a juicio de los autores, se debería incluir la difusión de los hallazgos.

Planificación del estudio

Con el propósito de explorar la identificación de faltas comunes en los diagramas UML, específicamente en los diagramas de casos de uso y en los diagramas de clase, así como también describir características de las herramientas que se han usado con la verificación de esas faltas en dichos diagramas, se estableció un conjunto de preguntas de investigación que orientan el estudio.

A. Preguntas de investigación

PI1. ¿Cuáles son las características de las tipologías propuestas para faltas comunes en los diagramas de casos de uso?

PI2. ¿Cuáles son las características de las tipologías propuestas para faltas comunes en los diagramas de clases?

PI3. ¿Cuáles son las características de las herramientas para verificación de diagramas de casos de uso con base en las taxonomías propuestas?

PI4. ¿Cuáles son las características de las herramientas para verificación de diagramas de clases con base en las taxonomías propuestas?

B. Selección de fuentes y estrategias de búsqueda

Una vez establecidas las preguntas de investigación, se continuó con la identificación de los estudios primarios seleccionados (EPS) para la revisión sistemática. Para estos estudios se seleccionaron las siguientes bases de datos (BD): IEE Explore, ACM Digital Library y Scopus, debido a que estas bases de datos contienen muchos trabajos de investigación sobre el área de ingeniería de software (Shaikh *et al.*, 2021), área relacionada con el tema de interés el cual es lenguaje de modelado unificado.

C. Creación de la cadena de búsqueda

La cadena de búsqueda se construyó con palabras clave que relacionen errores con el lenguaje de modelado unificado; se usó el operador OR para unir palabras en un mismo bloque y el AND para separar los bloques por temas. Finalmente, la cadena de búsqueda conceptual creada es la que se presenta a continuación:
("error" or "fault" or "defect" or "verification") and ("use case" or "class diagram")

D. Criterios de inclusión y de exclusión

En este apartado se presentan los criterios aplicados para la selección de estudios primarios. Estos criterios son:

Criterios de inclusión

1. Que el artículo tenga relación con la detección de errores o faltas, así como también herramientas para la verificación de dichos errores o faltas en los diagramas UML mencionados.
2. Que los estudios primarios hayan sido reportados entre 2012 y 2022.
3. Qué el artículo corresponda a una investigación empírica.

Criterios de exclusión

1. En los artículos de un mismo estudio que reporten avances parciales, se considera el más reciente.
2. Los estudios cuyo texto completo no es accesible, es decir, no se cuenta con los permisos para acceder al artículo completo en la base de datos.

Ejecución del estudio

En esta fase de estudio la actividad principal consistió en realizar el proceso de indagación de acuerdo con la cadena de búsqueda en las fuentes descritas en la sección anterior y como resultado se obtuvo un conjunto de estudios primarios para la investigación. En la Tabla 2 se puede observar la cadena utilizada en cada BD.

Tabla 2. Cadenas configuradas por cada base de datos

Base de datos	Cadena configurada
IEEE Xplore	(“Document Title”:error OR “Document Title”:fault OR “Document Title”:defect OR “Document Title”:verification) AND (“Document Title”:use case OR “Document Title”:class diagram)
ACM	[[Title: error] OR [Title: fault] OR [Title: defect] OR [Title: verification]] AND [[Title: use case] OR [Title: class diagram]]
Springer	(error OR fault OR defect OR verification) AND (use case OR class diagram)

Los documentos encontrados se muestran en la Tabla 3. Primero se buscaron en las tres bases de datos previamente seleccionadas. Se obtuvieron 1.884 artículos, y 1.856 artículos fueron descartados después de aplicar un primer filtrado con los criterios de inclusión y otros ocho en un segundo filtrado con los criterios de exclusión. Con el segundo filtrado, tres documentos de la IEEE y dos de Springer no se obtuvieron completos; también se encontraron tres documentos duplicados en ACM. Luego de ese proceso, quedaron 20 documentos distribuidos de la siguiente manera: IEEE (8 documentos, 40 %), ACM (7 documentos, 35 %) y Springer (5 documentos, 25 %).

Tabla 3. Resultados de la búsqueda

Base de datos	Resultados	Primer filtrado	Segundo filtrado
IEEE Xplore	165	11	8
ACM	1420	10	7
Springer	299	7	5
Total			20

Como se puede observar en la Figura 3, durante los últimos 10 años no ha existido una tendencia de publicación en ningún sentido, ciertamente se observan picos que fluctúan a lo largo de la década; se puede decir que el número de publicaciones por año es de dos en promedio.

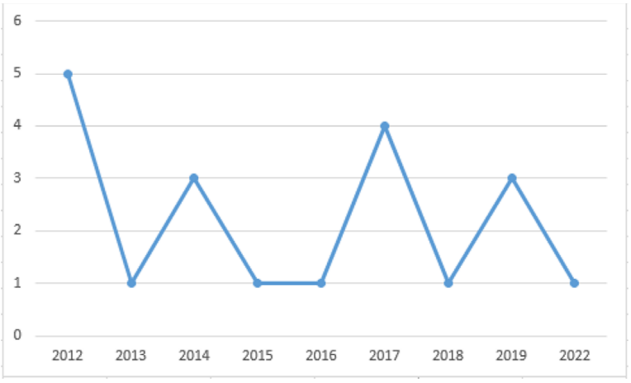

Figura 3. Número de EPS por año.

En la Tabla 4 se presenta el conjunto de artículos seleccionados por base de datos, así como el tipo de publicación; se puede observar que en un 70 % las publicaciones se han presentado en memorias de congresos.

Tabla 4. Estudios Primarios seleccionados por BD y Tipo de Publicación

# EPS	BD	Cita	Tipo de publicación	
			Memoria	Revista
Art_01	IEEE	Kayama *et al.* (2014)	x	
Art_02	IEEE	Halim (2013)	x	
Art_03	IEEE	Hakim, Sellami y Ben-Abdallah (2017)	x	
Art_04	IEEE	Shaikh y Wiil (2018)		x
Art_05	IEEE	Clarisó, González y Cabot (2017)		x
Art_06	IEEE	Claret y Régis-Gianas (2015)	x	
Art_07	IEEE	Chebanyuk y Markov (2016)	x	
Art_08	IEEE	Simko *et al.* (2012)	x	
Art_09	ACM	Shuang *et al.* (2014)	x	
Art_10	ACM	Tianual y Pohthong (2019)	x	
Art_11	ACM	Chourio *et al.* (2019)	x	
Art_12	ACM	Motta (2012)	x	
Art_13	ACM	Baresi *et al.* (2012)		x
Art_14	ACM	Parachuri, Sajeev y Shukla (2014)	x	
Art_15	ACM	Shaikh y Wiil (2012)	x	
Art_16	Springer	Agner, Lethbridge y Soares (2019)		x
Art_17	Springer	Chaudron, Heijstek y Nugroho (2012)		x
Art_18	Springer	Vanwormhoudt, Caron y Carré (2017)		x
Art_19	Springer	Wu (2022)	x	
Art_20	Springer	Sadowska y Huzar (2017)	x	

Como mecanismo para la evaluación de la calidad de los estudios primarios seleccionados, se consideró como criterio la completitud que puede tener cada estudio seleccionado para dar respuesta a las preguntas de investigación; así, un estudio podrá tener una valoración entre 0 y 1, según dé respuesta a las preguntas de investigación; por ejemplo, el Art_02 al dar respuestas a las preguntas 2 y 4 tuvo una valoración de 0,50 (Tabla 5).

De acuerdo con la Tabla 5 se puede observar que el conjunto de artículos seleccionados resultó tener una calidad apropiada para el estudio, ya que cada uno dio respuesta a al menos a una pregunta de investigación e incluso en el 33 % de los casos dio respuesta a dos.

Resultados

En esta sección se presentan los resultados obtenidos del estudio, los cuales permiten conocer propuestas para faltas en el lenguaje de modelado unificado, específicamente en los diagramas de casos de uso y en los diagramas de clases, lo anterior tanto en el ámbito educativo como en la práctica profesional. A continuación, se presentan las respuestas a las preguntas de investigación que fueron planteadas en la presente revisión sistemática de la literatura.

A. PI1. ¿Cuáles son las características de las tipologías propuestas para faltas comunes en los diagramas de casos de uso?

De los estudios primarios elegidos, en Hakim, Sellami y Ben-Abdallah (2017) se identificó que los autores proponen una aproximación para detectar y localizar errores de inter-consistencia en los diagramas de casos de uso; dicha aproximación está basada en dos fases: fase de medición y fase de control.

- La fase de medición consiste en medir el tamaño funcional y el tamaño estructural del diagrama de caso de uso, para esto los autores usan un mapeo de los elementos del diagrama de casos de uso.

Tabla 5. Evaluación de la calidad de los estudios primarios seleccionados

# EPS	PI-1	PI-2	PI-3	PI-4	Completitud
Art_01		x			0,25
Art_02		x		x	0,50
Art_03	x				0,25
Art_04		x	x	x	0,75
Art_05		x			0,25
Art_06	x				0,25
Art_07		x			0,25
Art_08	x				0,25
Art_09	x		x		0,50
Art_10	x				0,25
Art_11	x	x			0,50
Art_12		x		x	0,50
Art_13		x			0,25
Art_14	x	x			0,50
Art_15		x			0,25
Art_16			x		0,25
Art_17	x				0,25
Art_18				x	0,25
Art_19		x		x	0,50
Art_20		x			0,25

- La fase de control consiste en aplicar una serie de reglas para identificar y localizar los errores de inter-consistencia; es decir, proponen reglas de consistencia que se relacionan con la consistencia vertical entre la especificación y los modelos de diseño.

Similarmente, Shuang *et al.* (2014) proponen la detección automática de defectos en los documentos de casos de uso aprovechando técnicas avanzadas de análisis (*parsing*). La propuesta también consiste en dos fases. En la primera fase, a partir de los documentos de caso de uso, se examina cada sentencia dentro un análisis de árbol (dependencia y estructura), posteriormente, en la segunda fase, a partir del análisis de árbol, se genera un diagrama de actividades para cada caso de uso, como salida además del diagrama de actividades se obtiene el reporte de defectos.

Tianual y Pohthong (2019) proponen una técnica para la detección de defectos en las vistas de casos de uso durante la fase de análisis o ingeniería de requisitos; dicha propuesta usa los requisitos de los usuarios escritos en una especificación basada en un formulario para comparar. En este caso los defectos en los casos de uso los categorizan en el flujo incorrecto entre los casos de uso y la meta no alcanzada.

Por otra parte, Chourio *et al.* (2019) confeccionaron una metodología obteniendo como resultado una propuesta de los errores más comunes al usar UML. La metodología consta de cuatro etapas: levantamiento teórico de errores, seguimiento de las actividades del curso, análisis de los registros de etapa anterior y elaboración del catálogo de errores. En la primera etapa realizan una revisión de la literatura para encontrar los errores comunes en el modelado; en la segunda etapa verifican si los errores descritos en la literatura suceden en la práctica; en la tercera etapa realizan un análisis de la etapa anterior y en la última etapa proponen un catálogo de errores; para efectos del estudio, se adoptara el concepto de faltas para la tipología de errores propuesta. En la Tabla 6 se presentan las faltas identificadas en los diagramas de casos de uso.

Tabla 6. Faltas comunes en los diagramas de casos de uso.

Número de error	Nombre
3	Diferentes elementos para representar acciones similares.
4	Elementos similares para representar diferentes acciones.
5	Inadecuación en la denominación de los elementos.
12	Uso incorrecto de generalizaciones/especializaciones.
14	Reconocer acciones recurrentes en casos de uso: include.
15	Reconocer acciones de suscripción entre casos de uso: extend.
19	Reconocer cuándo reutilizar información de otros diagramas.

Fuente: Chourio *et al.* (2019)

En los estudios se encontró que otra forma de detectar faltas en los casos de uso es implementando verificación formal; por ejemplo, Claret y Régis-Gianas (2015) proponen una técnica para representar programas interactivos y casos de uso formalmente verificados; sin embargo, los autores no mencionan las faltas a detectar en los casos de uso. Algo similar ocurre en Parachuri *et al.* (2014) y Simko *et al.* (2012), es decir, proponen la verificación formal para la revisión de los casos de uso; no obstante, el estudio es en los documentos no en los diagramas casos de uso.

Finalmente, Chaudron *et al.* (2014) presentan una selección de evidencia empírica relacionada con la efectividad del UML; para esto analizan el costo y el beneficio del modelado expresando que un parámetro que explica el costo del modelado es el tamaño del modelo, pero también la falta de soporte de herramientas es importante. En este mismo estudio, interesados por la falta de preocupación de los grados de incompletitud e inconsistencias en los modelos UML, para ello mencionan que también realizaron un experimento donde utilizaron modelos UML, con y sin defectos a sujetos que consistían en estudiantes y profesionales; sin embargo, no describen ni mencionan cuáles son esos defectos.

B. PI2. ¿Cuáles son las características de las tipologías propuestas para faltas comunes en los diagramas de clases?

Continuando con los estudios primarios elegidos, tal como se comentó en la respuesta de la pregunta de investigación anterior, Chourio *et al.* (2019) confeccionaron una metodología y obtuvieron como resultado una propuesta de los errores más comunes al usar UML; en la Tabla 7 se presentan los errores comunes propuestos en su estudio para los diagramas de clases.

Tabla 7. Faltas comunes en los diagramas de clases.

Número de error	Nombre
3	Diferentes elementos para representar acciones similares.
4	Elementos similares para representar diferentes acciones.
7	Cardinalidad incorrecta.
12	Uso incorrecto de generalizaciones/ especializaciones.
17	Inadecuada descomposición de las funcionalidades del sistema

Fuente: Chourio *et al.* (2019)

Kayama *et al.* (2014) presentan los resultados de un análisis de errores en los diagramas de clase; estos resultados se obtienen de un estudio donde los autores exploran métodos educativos para el modelado conceptual de alumnos principiantes o novatos. Cabe mencionar que previo al estudio que presentan, los autores recolectaron y analizaron de 2010 a 2013 los errores en todos los diagramas finales de los estudiantes novicios, y a partir de eso proponen cuatro categorías de errores: sintácticos, relacionados con los atributos, relacionados con la asociación y relacionados con la clase. Estas categorías son las que organizaron para explorar sus métodos educativos; el resultado de los errores encontrados en los diagramas de clase en los métodos de enseñanza se presenta en la Tabla 8.

Por otro lado, Halim (2013) propone un modelo para medir la complejidad del software orientado a objetos en la fase de diseño para predecir las clases propensas a fallas. Así mimo, el autor menciona que el modelo de predicción se construye

Tabla 8. Faltas a considerar en los métodos de enseñanza.

Grupo de error	Nombre
11T, 12T, 13T	Definición de modelado conceptual.
11T, 12T, 13T	Notación de diagramas de clase.
11T, 12T, 13T	Creación de modelo.
11T, 12T, 13T	Prueba y creación de modelo con las mismas preguntas.
11T	Falta de descripción de los elementos requeridos (basados en sintácticos, relacionados con atributos, relacionados con asociaciones y clases).
11T	Dificultad para comprender la multiplicidad (errores relacionados con la asociación).
11T	Uso del mismo atributo en varias clases (errores relacionados con el atributo).
11T	Descripción del método o comportamiento de la clase objetivo como atributo (errores relacionados con atributos).
11T	Descripción del valor específico de un atributo (errores relacionados con el atributo).
11T	Bajo nivel de conocimiento de los atributos y relaciones adecuados que capturan los requisitos y coinciden con el dominio de destino (errores relacionados con los atributos).
12T	Diferencias entre los cuatro tipos de multiplicidad [1, 0…1, 1…*,*].
12T	Revisión incorrecta del modelo en pruebas de creación del modelo.
13T	Notación del diagrama de objetos (error en la distinción entre valor y atributo).
13T	Revisión de errores comunes en la creación del modelo.

Fuente: Kayama et al. (2014).

utilizando dos algoritmos de clasificación: Naive Bayesian y k-vecinos más cercanos.

Para el experimento presentado en este estudio utiliza ingeniería inversa, es decir a partir del código fuente genera las clases y sus relaciones; sin embargo, en ninguna sección del estudio comenta sobre las faltas que detectan en los diagramas de clases.

De manera similar a los diagramas de casos de uso, se encontró que otra forma de detectar fallas en los diagramas de clase es implementado verificación formal; por ejemplo, en Shaik y Wiil (2018), la propuesta consiste en transformar los diagramas de clases UML en código junto con sus restricciones del lenguaje de especificación de objetos (OCL, por su sigla en inglés), para esto, los autores con el objetivo de obtener una verificación eficiente, proponen el uso de verificación formal con técnicas de corte y retroalimentación. Antes de la transformación, proponen encontrar defectos en el modelo, para esto se verifica que las clases tengan su estructura típica junto con sus asociaciones correspondientes, estas de listan en la Tabla 9.

Tabla 9. Estructura y relaciones de clases

Estructura	Nombre de la clase, atributos y relaciones entre clases
Relaciones de clases	Asociación, agregación y composición
Otras relaciones	Generalización y realización, dependencia y multiplicidad

También Clarisó *et al.* (2019) proponen verificación formal, pero a diferencia de Shaik y Wiil (2018), ellos presentan técnicas de verificación limitada, por ejemplo, solucionadores de programación de especificación (CP).

Chebanyuk y Markov (2016) proponen una verificación en los diagramas de clases utilizando principios de diseño SOLID, donde SOLID representa cinco principios básicos de la programación orientada a objetos y a diseño, estos son: principio de responsabilidad única, principio de abierto cerrado, principio de sustitución de Liskov, principio de segregación de la interfaz y principio de inversión de la dependencia.

Parachuri *et al.* (2014), Motta (2012), Baresi *et al.* (2012), Shaikh y Wiil (2012), Wu (2022) y Sadowska y Huzar (2017) también plantean

verificación formal en la revisión de los diagramas de clase, por ejemplo, Motta (2012), Baresi *et al.* (2012) y Sadowska y Huzar (2017) proponen una semántica formal; sin embargo, en los documentos no mencionan los errores buscados.

C. PI3. ¿Cuáles son las características de las herramientas para verificación de diagramas de casos de uso con base en las taxonomías propuestas?

De los estudios primarios elegidos y relacionados con esta pregunta de investigación, en Shuang *et al.* (2014) se encontró que los autores proponen algoritmos para verificar automáticamente defectos relacionados con la coherencia y la integridad en los casos de uso, pero esta verificación está en los documentos de caos de uso no en los diagramas de casos uso.

Por otro lado, Agner *et al.* (2019) presentan una revisión de la literatura relacionada con las herramientas UML más importante que permita al estudiante determinar sus fortalezas y debilidades en el modelado; lo interesante de este estudio es la presentación de las herramientas UML que se pueden utilizar en el diseño de software, pero no se hace mención en el estudio de alguna herramienta que detecte fallas en los diagramas de casos de casos de uso.

Shaikh y Wiil (2018), para la verificación formal de los diagramas de clases, utilizan una herramienta llamada UMLtoCSP. La herramienta recibe como entrada un diagrama de clases con restricciones OCL y verifica automáticamente la precisión de las propiedades.

En general, dentro de los resultados de los estudios primarios elegidos se observó que no hay una herramienta en específico para detectar faltas en los diagramas de casos de uso, en la mayoría de los estudios revisados se realiza manualmente.

D. PI4. ¿Cuáles son las características de las herramientas para verificación de diagramas de clases con base en las taxonomías propuestas?

En cuanto a las herramientas para la detección de faltas en los diagramas de clases, Halim (2013) comenta que para la recopilación de datos utilizó cuatro versiones de JEdit, esta herramienta es un proyecto de código abierto basado en JAVA; sin embargo, tal como se comentó en la respuesta a la pregunta de investigación anterior, no se mencionan las faltas que se detectan en los diagramas de clases.

Vanwormhoudt *et al.* (2017) presentan el uso plantillas aspectuales, el cual requiere que los parámetros formen un modelo de sistemas que inyecten nuevas funcionalidades. En el UML estándar, una plantilla es un modelo que expone algunos de sus elementos como parámetros; por ejemplo, las clases o los paquetes son denominados plantillas de clase o plantillas de paquete, respectivamente. Lo interesante de este estudio es la presentación de una herramienta para asistir en la construcción de sistemas UML con plantillas aspectuales; por ejemplo, *plugins* para eclipse, pero no se menciona en el estudio de alguna herramienta que detecte fallas en los diagramas de clases.

Finalmente, como se comentó en la respuesta a la pregunta de investigación PI2, algunos autores proponen verificación formal para el estudio de los diagramas de clases (Motta, 2012; Shaikh y Wiil, 2018; Wu, 2022), así como también proponen una herramienta de desarrollo propio.

Conclusiones

En este estudio se presentó una revisión sistemática de la literatura relacionada con la detección de faltas comunes durante el aprendizaje del modelado con diagramas de UML, específicamente en los diagramas de casos de uso y diagramas de clase. La investigación se basó en los estudios de los últimos 10 años publicados en las bases de datos IEEE, ACM y Springer.

Siguiendo el protocolo de investigación planteado se seleccionaron y analizaron 20 estudios primarios reportados; con lo anterior, se observa

en primera instancia que el número de estudios recuperados fue escaso y, por otro lado, el tipo de publicaciones en su gran mayoría (70 %) proviene de memorias de congresos, por tal motivo se puede concluir que la investigación en torno a las tipologías de faltas en los diagramas UML es incipiente, en particular, diagramas de casos de uso y diagramas de clases.

En relación con el uso de alguna herramienta de software, se encontró que se utilizó alguna herramienta de desarrollo propio; sin embargo, para los diagramas de casos de uso se enfocaron más en los documentos de casos de uso que en los diagramas UML. Algo similar ocurre con los diagramas de clases, en estos estudios utilizan métodos formales para la verificación de los diagramas; sin embargo, no presentan cuáles son las faltas comunes que verifican.

Como trabajos futuros se considera repetir la revisión sistemática en otras bases de datos para los dos diagramas presentados en este artículo; hacer la revisión sistemática considerando los otros tipos de diagramas UML; así como proponer una taxonomía de faltas y realizar un estudio primario para su validación.

Contribución de autoría

Juan-Pablo Ucán-Pech: Conceptualización, Curación de datos, Análisis formal, Investigación, Metodología, Supervisión, Administración de proyecto, Escritura – borrador original, Visualización.
Raúl-Antonio Aguilar-Vera; Análisis formal, Adquisición de fondos, Metodología, Validación, Escritura-revisión y edición.
Julio-César Díaz-Mendoza: Recursos, Escritura - revisión y edición.
Omar-Salvador Gómez-Gómez; Investigación, Metodología, Visualización.

Agradecimientos

A la Secretaría de Educación Pública (México) a través del proyecto P/PROFEXE-2020-31MSU0098J-13.

Referencias

Agner, L. T., Lethbridge, T. C., Soares, I. W. (2019). Student experience with software modeling tools. *Softw. Syst. Model.*, *18*, 3025-3047. https://doi.org/10.1007/s10270-018-00709-6

Aguilar, R., Oktaba, H., Ramírez, R., Aguilar, J., Fernández, C., Rodríguez, O., Ucán, J. (2017). Ingeniería de Software. En L. A. Pineda Cortés (coord.), *Computación en México por especialidades académicas* (pp. 167-194). Amexcomp

Aguilar, R., Oktaba, H., Juárez, J. (2019). Introducción a la ingeniería de software. En R. A. Aguilar (ed.), *Ingeniería de software en México: educación, industria e investigación* (pp. 3-17). Amexcomp

Alanazi, M. N. (2009). Basic rules to build correct UML diagrams. En *International Conference on New Trends in Information and Service Science*, 72-76. https://doi.org/10.1109/NISS.2009.252

Baresi, L., Morzenti, A., Motta, A., Rossi, M. (2012). A logic-based semantics for the verification of multi-diagram UML models. *ACM SIGSOFT Software Engineering Notes, 37*(4), 1-8. https://doi.org/10.1145/2237796.2237811

Brunet, J., Guerrero, D., Figueiredo, J. (2009). Design tests: An approach to programmatically check your code against design rules. En *31st International Conference on Software Engineering*, 255-258. https://doi.org/10.1109/icse-companion.2009.5070995

Chaudron, M. R., Heijstek, W., Nugroho, A. (2014). How effective is UML modeling?. *Software & Systems Modeling*, *11*, 571-580. https://doi.org/10.1007/s10270-012-0278-4

Chebanyuk, E., Markov, K. (2016). An approach to class diagrams verification according to SOLID design principles. En *4th International Conference on Model-Driven Engineering and Software Development*, 435-441

Chourio, P., Azevedo, R., Castro, A., Gadelha, B. (2019). Most common errors in software modeling using UML. En *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, 244-253. https://doi.org/10.1145/3350768.3353820

Claret, G., Régis-Gianas, Y. (2015). Mechanical verification of interactive programs specified by use cases. En *IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering*, 61-67. https://doi.org/10.1109/FormaliSE.2015.17

Clarisó, R., González C. A., Cabot J. (2019). Smart bound selection for the verification of UML/OCL class diagrams. *IEEE Transactions on Software Engineering*, *45*(4), 412-426. https://doi.org/10.1109/TSE.2017.2777830

Genero, M., Cruz-Lemus, J. A., Piattini, M. G. (2014). *Métodos de investigación en ingeniería del software*. Ra-Ma

Hakim, H., Sellami, A., Ben-Abdallah, H. (2017). Identifying and localizing the inter-consistency errors among UML use cases and activity diagrams: An approach based on functional and structural size measurements. En *IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, 289-296. https://doi.org/10.1109/sera.2017.7965740

Halim, A. (2013). Predict fault-prone classes using the complexity of UML class diagram. En *International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, 289-294. https://doi.org/10.1109/IC3INA.2013.6819188

Juristo, N., Moreno, A. M., Vegas, S. (2006). *Técnicas de evaluación de software*. Universidad Politécnica de Madrid

Kayama, M., Ogata, S. Masymoto, K., Hashimoto M., Otani M. (2014). A practical conceptual modeling teaching method based on quantitative error analyses for novices learning to create error-free simple class diagrams. En *IIAI 3rd International Conference on Advanced Applied Informatics*, 616-622. https://doi.org/10.1109/IIAI-AAI.2014.131

Kitchenham, B., Charters, S. (2007). *Guidelines for Performing Systematic Literature Reviews in Software Engineering* (Technical Report). Keele University and University of Durham

Motta, A. (2012).Towards the verification of multi-diagram UML models. En *34th International Conference on Software Engineering (ICSE)*, 1531-1534. https://doi.org/10.1109/ICSE.2012.6227044

Parachuri, D., Sajeev, A. S. M., Shukla, R. (2014). An empirical study of structural defects in industrial use-cases. En *ICSE Companion 2014: Companion Proceedings of the 36th International Conference on Software Engineering*, 14-23. https://doi.org/10.1145/2591062.2591174

Pressman, R., Maxim, B. (2021). *Ingeniería del software: un enfoque práctico*. McGraw-Hill

Rumbaugh, J., Jacobson, I., Booch, G. (2000). *El lenguaje unificado de modelado. Manual de referencia*. Addison Wesley

Sadowska, M., Huzar, Z. (2017). Semantic validation of UML class diagrams with the use of domain ontologies expressed in OWL 2. In: L. Madeyski, M. Śmiałek, B. Hnatkowska, Z. Huzar (eds.), *Software Engineering: Challenges and Solutions*, pp. 47-59. Springer. https://doi.org/10.1007/978-3-319-43606-7_4

Seidl, M., Scholz, M., Huemer, C., Kappel, G. (2015). *UML @ Classroom: An introduction to Object-Oriented Modeling*. Springer

Shaikh, A., Hafeez, A., Wagan, A. A., Alrizq, M., Alghamdi, A., Al Reshan, M. S. (2021). More than two decades of research on verification of UML class models: A systematic literature review. *IEEE Access, 9*, 142461-142474. https://doi.org/10.1109/ACCESS.2021.3121222

Shaikh, A., Wiil, U. K. (2012). UMLtoCSP (UOST): A tool for efficient verification of UML/OCL class diagrams through model slicing. En *FSE '12: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software*, 1-4. https://doi.org/10.1145/2393596.2393639

Shaikh, A., Wiil, U. K. (2018). Overview of slicing and feedback techniques for efficient verification of UML/OCL class diagrams. *IEEE Access,* 6, 23864-23882. https://doi.org/10.1109/ACCESS.2018.2797695

Shuang, L., Jun, S., Yang, L., Yue, Z., Bimlesh, W., Jin, S. D., Xinyu, W. (2014). Automatic early defects detection in use case documents. En *ASE '14: Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, 785-790. https://doi.org/10.1145/2642937.2642969

Simko, V., Hnetynka, P., Bures, T., Plasil, F. (2012). FOAM: A lightweight method for verification of use-cases. En *38th Euromicro Conference on Software Engineering and Advanced Applications*, 228-232. https://doi.org/10.1109/SEAA.2012.15

Tianual, P., Pohthong, A. (2019). Defects detection technique of use case views during requirements engineering. En *ICSCA '19: Proceedings of the 2019 8th International Conference on Software and Computer Applications*, 277-281. https://doi.org/10.1145/3316615.3316631

Torre, D., Labiche, Y., Genero, M., Baldassarre, M. T., Elaasar, M. (2018). UML diagram synthesis techniques: A systematic mapping study. En *MiSE '18: Proceedings of the 10th International Workshop on Modelling in Software Engineering*, 33-40. https://doi.org/10.1145/3193954.3193957

Vanwormhoudt, G., Caron, O., Carré, B. (2017). Aspectual templates in UML. *Software & Systems Modeling*, *16*, 469-497. https://doi.org/10.1007/s10270-015-0463-3

Wu, H. (2022). QMaxUSE: A query-based verification tool for UML class diagrams with OCL invariants. In: E. B. Johnsen, M. Wimmer (eds.), *Fundamental Approaches to Software Engineering*, pp. 310-317. *Springer*. https://doi.org/10.1007/978-3-030-99429-7_17