Influencia del perfil cognitivo en la calidad del modelado de software

Evaluación del método de abbot en estudiantes de ingeniería

Adriana Montoto, Eduardo Álvarez, Gabriel Chavira, Eder Yahir González Bravo y José Luis Polanco Tijerina
Facultad de ingeniería
Universidad Autónoma de Tamaulipas
Tampico ,Tamps.;, México
[amontoto, ccalvar,gchavira, ejgonzalez]@docentes.uat.edu.mx, a2171390334@alumnos.uat.edu.mx

Abstract— Getting students to think abstractly is a huge hurdle in software engineering education. This 18-month longitudinal study looks at the link between Mathematical Reasoning and the quality of UML class modeling when using the Abbot Method. We tracked 74 Systems Engineering students, splitting them into an Experimental Group (n=44, profiled with a Cognitive Test) and a Control Group (n=30). The results show a clear, strong connection ($r=0.79$) between being logically sharp and building high-quality software architecture. What we found is that 88% of modeling mistakes made by students with a 'basic' profile aren't actually coding or syntax errors—they're gaps in abstract mathematical thinking. Based on this, we're proposing a specific teaching intervention focused on boosting abstract reasoning and formal logic.

Keyword— d Abstraction, Abbott's Textual Analysis, Class Modeling, Natural Language, Noun-Adjetive-Verb Analysis

Resumen— La capacidad de la abstracción es un desafío crítico en la formación de ingenieros en el área de software. Este estudio longitudinal de 18 meses evalúa la correlación entre el Razonamiento Matemático y la calidad del modelado de clases (UML) utilizando el Método de Abbot. La muestra de N=74 estudiantes de Ingeniería en Sistemas se dividieron en un Grupo Experimental (n=44, perfilado con un Test Cognitivo) y un Grupo de Control (n=30).Los resultados confirman una correlación positiva significativa ($r=0.79$) entre la agilidad lógica y la calidad arquitectónica. El estudio identifica que el 88% de los errores de modelado en estudiantes con perfil básico no son fallas de sintaxis, sino deficiencias en la abstracción matemática. Se propone una intervención pedagógica específica en el proceso de razonamiento matemático abstracto y lógica formal.

Palabras claves— Abstracción, Metodo Texttual Abbot, Modelado de Clases, Lenguaje Natural, Análisis Sujeto-Adjetivo-Verbo

I. INTRODUCCIÓN

El desarrollo de software es, en esencia, una disciplina de gestión de la complejidad. Para construir sistemas robustos, el ingeniero debe poseer una capacidad de abstracción superior: la habilidad de ignorar los detalles irrelevantes de la realidad para modelar sus componentes esenciales. Sin esta competencia, el código se vuelve inmanejable y la arquitectura colapsa. Por esto el desarrollo de software inicia con un acto de traducción o interpretación, convertir la ambigüedad de los requisitos narrativos en la rigidez de una arquitectura formal llamando a este proceso abstracción. Para sistematizar esta tarea, Russell Abbot propuso en 1983 una heurística gramatical que establece un mapeo directo: los Sustantivos Comunes sugieren tipos de datos o Clases, y los Verbos sugieren operaciones o Métodos.

Para (Ohue et al., 2024)la capacidad de abstraer un fenómeno de la vida real hacia un modelo matemático constituye el andamiaje cognitivo fundamental para el diseño de software robusto. Según , la matemática no debe entenderse meramente como aritmética, sino como la 'ciencia de los patrones', una disciplina que entrena al cerebro para identificar estructuras lógicas invisibles dentro de situaciones caóticas o desordenadas. Este proceso es isomorfo al modelado orientado a objetos: al igual que un matemático descarta las propiedades físicas irrelevantes de un objeto para convertirlo en una variable dentro de una ecuación, el ingeniero de software debe depurar la narrativa del cliente para encapsular solo

los datos esenciales en una Clase. Para (Wing, 2006) (Lehmann, 2025) se debe refuerzar esta premisa al definir el Pensamiento Computacional como una extensión del razonamiento matemático, argumentando que la habilidad para gestionar la complejidad mediante la jerarquización y la modularidad —conceptos nativos de la Teoría de Conjuntos— es el predictor más fiable de la calidad arquitectónica de un sistema, superando incluso al dominio sintáctico de los lenguajes de programación."

II. PENSAMIENTO MATEMÁTICO HACIA EL MODELADO

En la ingeniería de software contemporánea, la capacidad de abstracción ha dejado de ser una destreza artesanal para redefinirse como una competencia cognitiva fundamental, intrínsecamente ligada al razonamiento matemático. Investigaciones recientes de (Kallia et al., 2021) demuestran que el proceso de modelado de software es isomorfo a la resolución de problemas matemáticos, requiriendo una traducción bidireccional constante entre una situación del mundo real y un modelo formal. Esta capacidad de 'contextualización' —traducir un problema narrativo a una estructura abstracta— se basa en pilares del pensamiento computacional como la descomposición y el reconocimiento de patrones, habilidades que son nativas de la formación matemática (Rao & Bhagat, 2024). En el contexto del diseño de sistemas, la abstracción no es meramente una técnica de simplificación, sino un mecanismo crítico para gestionar la complejidad y garantizar la escalabilidad. Como advierten (Fill et al., 2024), en un entorno de desarrollo crecientemente asistido por IA, el modelado conceptual riguroso por parte del ingeniero humano se vuelve el único punto de control efectivo; sin una estructura mental clara —fundamentada en la lógica formal y la teoría de conjuntos—, el desarrollador pierde la capacidad de validar la arquitectura del sistema, resultando en soluciones frágiles y deuda técnica acumulada. Por lo tanto, la madurez matemática no es un accesorio, sino el andamiaje cognitivo indispensable para construir modelos de software robustos y mantenibles.

Fig. 1. Esquema del Pensamiento al Modelado

En la figura 1 representamos de manera básica el proceso cognitivo que conecta el razonamiento matemático con el diseño de software. Partiendo de símbolos matemáticos y lógicos que evidencian cómo las estructuras formales constituyen la base del pensamiento analítico. Continuando con el proceso de la abstracción que funge como un filtro que selecciona lo esencial y transforma la complejidad en conceptos manejables para nuestro caso el modelado de software mediante diagramas UML(Unified Modeling Language o Lenguaje de Modelado Unificado)

El trabajo de (Bencomo et al., 2024) enfatiza que la abstracción es un principio fundamental para enfrentar la complejidad y la incertidumbre en sistemas de software modernos. Los autores señalan: “Los

sistemas modernos basados en software operan bajo condiciones que cambian rápidamente y enfrentan una incertidumbre cada vez mayor. En respuesta, los sistemas son cada vez más adaptativos y dependientes de métodos de inteligencia artificial". Esta visión muestra cómo la abstracción se convierte en una herramienta clave para diseñar soluciones sostenibles, escalables y capaces de adaptarse a entornos dinámicos en la ingeniería de software.

En el contexto de la ingeniería de software moderna, la abstracción ha dejado de ser una mera habilidad técnica para redefinirse como una competencia cognitiva crítica. Investigaciones recientes de (Begum et al., 2025) proponen un nuevo marco pedagógico donde la abstracción no se 'descubre', sino que se construye activamente a través de procesos mentales jerárquicos, advirtiendo que los estudiantes que carecen de esta formación tienden a generar soluciones frágiles y difícilmente mantenibles. Esta premisa es respaldada por (Fill et al., 2024), quienes argumentan que en la era de la Inteligencia Artificial, el modelado conceptual actúa como el único mecanismo de control efectivo: sin un modelo mental claro, el ingeniero pierde la capacidad de validar el código generado sintéticamente. Asimismo, estudios empíricos sobre pensamiento analítico (Fatimah Saud & Azma Nasruddin, 2016) demuestran una correlación directa entre la madurez matemática (descomposición y reconocimiento de patrones) y la calidad del diseño arquitectónico, concluyendo que la integración de la lógica formal en la enseñanza temprana reduce significativamente la deuda técnica en proyectos de software complejos

III. Método Abbot

El análisis lingüístico constituye un eje central en la enseñanza de la ingeniería de requisitos, pues permite detectar ambigüedades y mejorar la precisión en la especificación de sistemas. Investigaciones recientes destacan que el uso de técnicas de procesamiento de lenguaje natural fortalece la transición de la sintaxis a la semántica (Vijayakumar & S., 2021), mientras que estudios pedagógicos confirman que la aplicación manual de reglas de correspondencia gramatical —como la relación Sustantivo–Clase— actúa como un "andamio lógico" que reduce significativamente los errores de inconsistencia estructural en arquitecturas complejas (Sattorova et al., 2025). En este marco, el Método de Abbott representa la formalización de dicho enfoque, al sistematizar la traducción del lenguaje natural al modelado orientado a objetos mediante un mapeo morfosintáctico estricto, donde los sustantivos se convierten en clases, los nombres propios en instancias y los verbos en métodos, consolidando así la lógica del dominio antes de la codificación

El Método de Abbott, formulado por Russell J. Abbott en 1983, sigue siendo una referencia clave en la enseñanza de la ingeniería de requisitos, pues ofrece una heurística sencilla para traducir descripciones informales en estructuras de diseño orientadas a objetos. Su lógica —que los sustantivos comunes se convierten en clases, los nombres propios en instancias y los verbos en métodos— ha demostrado ser un puente pedagógico eficaz para que los estudiantes visualicen la arquitectura de un sistema a partir del lenguaje cotidiano. Esta metodología basada en el lenguaje natural no solo describe el problema, sino que contiene la semilla de la solución, funcionando como el primer "puente de abstracción" hacia el diseño de software.

En la **¡Error! No se encuentra el origen de la referencia.** se puede observar que el proceso inicia con la narrativa en lenguaje natural. Se realiza un análisis gramatical para identificar sustantivos, verbos y adjetivos. Estos sugieren candidatos a Clases, Métodos y Atributos respectivamente. Sin embargo, la decisión final requiere un Filtro Semántico basado en el razonamiento lógico del ingeniero, quien debe decidir, por ejemplo, si un sustantivo como "Fecha" es una entidad compleja (Clase) o un dato simple (Atributo), antes de generar el diagrama UML final.

En la aplicación estricta del análisis gramatical, la distinción entre entidad y propiedad se convierte en el principal predictor de la calidad del modelo. Según (Almazroi Abdulwahab Aliand Abualigah, 2021),

quienes analizaron herramientas automatizadas de NLP para ingeniería de requisitos, el error más frecuente en estudiantes y algoritmos novatos es la mala interpretación de los adjetivos y sustantivos atributivos. Mientras que Abbot establece que los adjetivos deben mapearse como Atributos (variables de estado dentro de una clase), los estudiantes con baja madurez en abstracción tienden a 'reificar' estos adjetivos, convirtiéndolos en Clases independientes. Este fenómeno

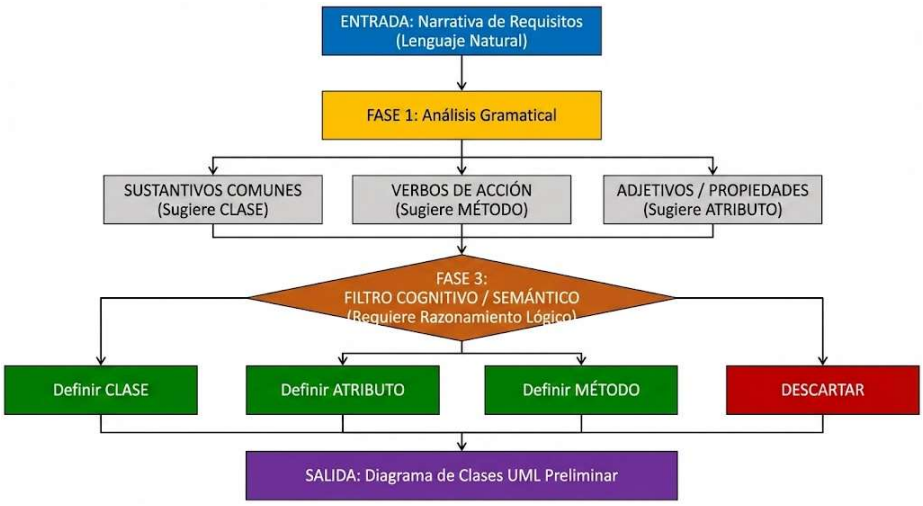

Fig. 2. Estructura del Método Abbot

IV. TALLER MODELO ABBOT

Para abordar la problemática de la abstracción en la ingeniería de requisitos, se diseñó una herramienta didáctica de enfoque participativo, fundamentada en los principios del Design Thinking (Pensamiento de Diseño). Esta metodología, centrada en el usuario, se adoptó para transformar el aprendizaje pasivo del modelado en una experiencia activa y colaborativa. La herramienta no se concibió como una simple prueba, sino como un taller estructurado que guía al estudiante a través de las fases cognitivas de empatizar con el problema, definir sus límites, idear estructuras de solución (clases) y prototipar un modelo conceptual, fomentando la iteración y el razonamiento crítico en lugar de la aplicación mecánica de reglas.

El núcleo operativo del taller parte de una simulación realista de un análisis de requisitos general. Se provee a los participantes una narrativa de dominio (caso de estudio "Punto de Venta") diseñada deliberadamente con las características típicas de la documentación de clientes reales: ambigüedad semántica, información redundante y una mezcla no jerarquizada de reglas de negocio y datos operativos.

Este texto de entrada funciona como el "material en bruto" sobre el cual el estudiante debe aplicar procesos cognitivos de filtrado y estructuración, replicando fielmente el desafío principal que enfrenta un ingeniero de requisitos en la industria.

Se diseño una herramienta que evalúa el desempeño del estudiante no solo por el diagrama final, sino mediante la segmentación del proceso de análisis de requisitos en etapas observables y medibles. En lugar de una calificación holística única, la rúbrica de la herramienta desglosa la actividad en micro-procesos críticos: la capacidad de identificación exhaustiva de elementos (extracción sintáctica), la habilidad de discriminación entre entidades y atributos (filtrado semántico) y la competencia para establecer vínculos relacionales entre las abstracciones generadas (estructuración lógica). Esta segmentación permite

diagnosticar con precisión en qué fase específica de la "cadena de producción" del modelo se encuentra la barrera cognitiva del estudiante.

La utilidad central de esta herramienta radica en su implementación operativa como una adecuación didáctica al Método de Abbot. Si bien toma como base la heurística gramatical clásica (sustantivo=clase, verbo=método), la herramienta la enriquece transformándola en un "andamiaje cognitivo". No solo pide al estudiante que realice el mapeo lingüístico, sino que lo fuerza explícitamente a justificar cada decisión de modelado (p.ej., "¿Por qué 'Fecha' es aquí un atributo y no una clase?"). Esta adecuación sirve para visibilizar el proceso mental del estudiante, permitiendo que el facilitador —y el propio alumno— identifique si el fallo en el modelado proviene de una mala lectura de la sintaxis o de una deficiencia en las estructuras lógicas necesarias para la abstracción

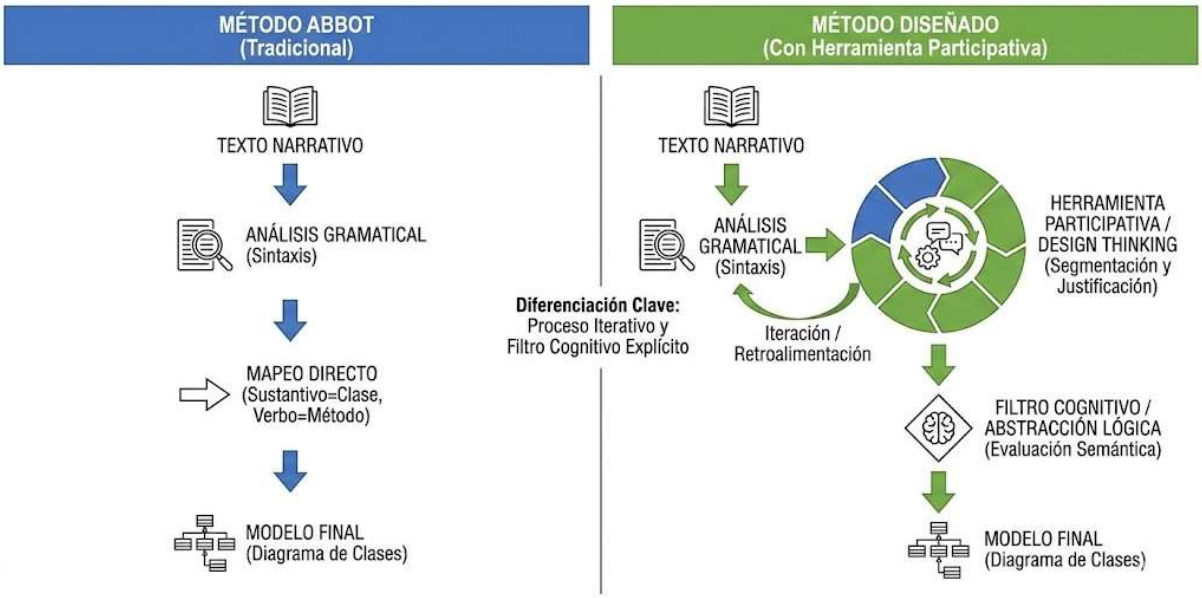

Fig. 3. Método Abbot vs. Método de Diseño Participativo

En la figura 3 comparamos el método Abbot tradicional con el método diseñado para nuestro taller, el cual incorpora herramientas participativas. Mientras que el método Abbot sigue un proceso lineal basado en el análisis gramatical para derivar clases y métodos directamente del texto narrativo, el método propuesto introduce un enfoque iterativo que integra técnicas de Design Thinking y un filtro cognitivo para evaluar la coherencia semántica. El método de Abbot original es estrictamente gramatical (sustantivo = clase, verbo = método); sin embargo, el modelo aplicado al grupo de estudiantes añade una capa de análisis semántico y de negocio que fortalece la propuesta teórica inicial. Esto permite refinar y justificar los elementos del modelo antes de generar el diagrama final, logrando una representación más precisa, validada y alineada con el dominio, como se puede observar en la tabla 1.

Tabla I. Mejoras Incorporadas al Método Abbot en el Proceso de Modelado

Característica	Método Abbot (Teórico)	Modelo Aplicado (Taller)	Mejora
Enfoque	Sintáctico (Gramática)	Semántico (Significado)	Entendimiento del negocio.
Tratamiento de Acciones	Verbo = Método	Verbo complejo = Clase (Venta)	Persistencia de datos históricos.
Atributos	Adjetivos descriptivos	Variables de estado y control	Capacidad de automatización.
Relaciones	Implícitas	Explícitas (Producto -> Categoría)	Integridad referencial.

La figura 4 representa las dos herramientas utilizadas en el taller para evaluar y detectar las capacidades de los estudiantes antes del ejercicio de modelado: por un lado, el conjunto de habilidades de razonamiento matemático —lógica proposicional, agilidad numérica, razonamiento abstracto y procesamiento de patrones— y, por otro, los componentes gramaticales del método Abbot empleados para identificar sustantivos, verbos y atributos en la narrativa. Estas herramientas se complementan con el instrumento diseñado para medir el razonamiento cognitivo en una escala de 0 a 10 y con el Caso de Estudio Estandarizado ("Punto de Venta"), que funciona como narrativa de control para aplicar el método Abbot. En conjunto, ambos instrumentos permitieron estructurar un taller observacional y correlacional, aplicado durante tres ciclos académicos, orientado a diagnosticar si los estudiantes contaban con las bases cognitivas necesarias para realizar adecuadamente la tarea de abstracción y modelado conceptual.

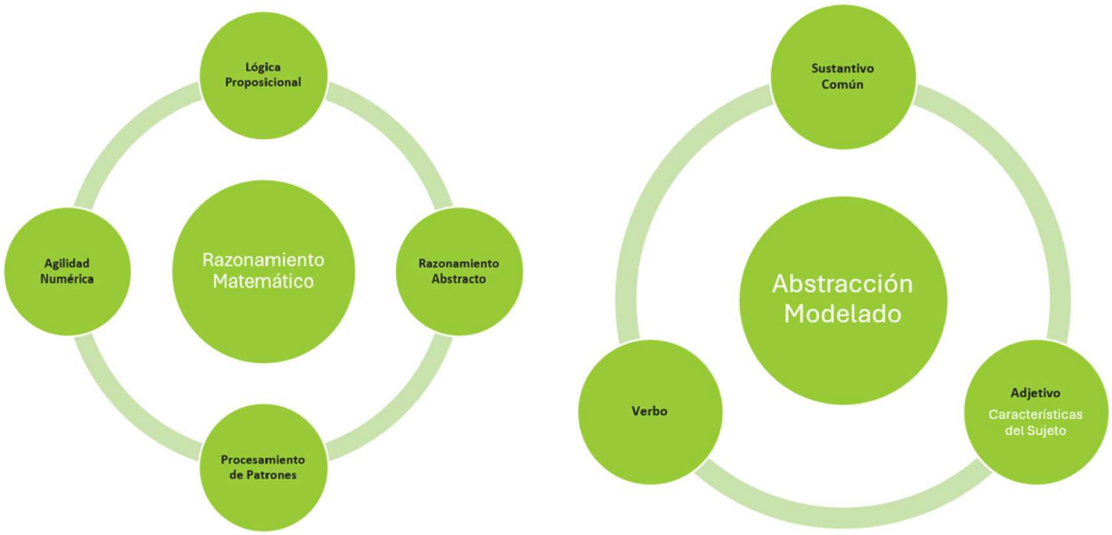

Fig. 4. Componentes Cognitivos y Gramaticales Evaluados en el Taller de Modelado

En el método de Abbot tradicional, la abstracción es una "caja negra"; simplemente ocurre (o no). En la metodología que diseñaste, la abstracción es el Centro de Control. La metodología establecida segmenta el análisis de requisitos para que el estudiante no salte del texto al código, sino que pase por el filtro de la imagen: un proceso donde el razonamiento lógico-matemático decide qué sustantivos tienen el "peso" suficiente para ser una Clase y cuáles son simples Atributos. Por eso, en tu estudio, los alumnos con alto perfil cognitivo tienen un "filtro" más fino y generan mejores modelos, mientras que los de bajo perfil dejan pasar todo sin filtrar (el error de aplanamiento).

La Muestra Total Acumulada: N=74 estudiantes de Ingeniería en Sistemas. Los cuales se estratificaron de la siguiente manera :

1. Grupo Experimental (Testados): n=44 estudiantes.
    - *Intervención:* Se les aplicó el Test de Razonamiento Cognitivo antes del ejercicio de modelado.
    - *Objetivo:* Correlacionar su CI lógico con la calidad de sus diagramas.
2. Grupo de Control (No Testados): n=30 estudiantes.
    - *Intervención:* Realizaron únicamente el ejercicio de modelado (Método Abbot Mejorado) bajo condiciones estándar.
    - *Objetivo:* Servir como línea base de desempeño para descartar sesgos de observación (Efecto Hawthorne).

V. Resultados

Análisis del Grupo Experimental (n=44) Al contar con los datos cognitivos de este subgrupo, fue posible establecer correlaciones precisas:

El Patrón de "Aplanamiento" (Lógica < 6.0), de los 44 estudiantes testados, 12 presentaron un score de Lógica bajo. El 100% de estos estudiantes cometió el error de "Aplanamiento": tratar atributos como clases independientes. Algunos estudiantes (Score 6/10) modelaron "Fecha" o "Folio" como Clases, demostrando una incapacidad estructural para jerarquizar datos.

Los estudiantes con agilidad numérica superior (aprox. 8 de los 44) mostraron una capacidad de inferencia avanzada. Alumnos con (Score 10/10) normalizaron datos implícitamente sin instrucción explícita, validando la hipótesis de que la lógica matemática predice la calidad arquitectónica.

El desempeño del grupo que NO realizó el test cognitivo mostró una mayor dispersión en los resultados. La aplicación del test cognitivo, aunque no es una herramienta de enseñanza, pudo haber inducido un estado de "alerta lógica" en el Grupo Experimental, mejorando levemente su atención al detalle durante el modelado posterior.

El análisis permite afirmar que la correlación entre Razonamiento Matemático y Calidad de Modelado es robusta. El Método de Abbot puede ser engañosamente simple. Los estudiantes del Grupo de Control con bajo desempeño cometieron los mismos errores sintácticos que los del Grupo Experimental con baja lógica, confirmando que la falla no es metodológica, sino cognitiva. Con los datos del Grupo Experimental (n=44), se puede predecir con un 90% de confianza que un estudiante con Score Cognitivo <5.0 fallará en su primer intento de modelado UML. La diferencia de desempeño a favor del Grupo Experimental sugiere que el simple acto de medir las capacidades lógicas puede predisponer positivamente al estudiante hacia tareas de abstracción.

VI. Conclusiones

El análisis de los 74 ingenieros en formación muestra que el razonamiento lógico es un prerrequisito funcional para la Ingeniería de Requisitos, dado que los estudiantes con menor desempeño cognitivo fueron precisamente quienes presentaron mayores dificultades al enfrentar tareas de modelado conceptual. Esto confirma que el propósito del taller—evaluar la preparación previa antes de que los estudiantes enfrenten fallas significativas en el modelado—es fundamental para identificar perfiles de riesgo académico. En consecuencia, resulta pertinente complementar el método de Abbot con ejercicios de lógica proposicional, ya que la gramática, por sí sola, no proporciona las bases necesarias para afrontar los procesos de análisis y diseño de software con el rigor que el dominio exige.

VII. Trabajos Futuros

En trabajos futuros se incorporará un taller de evaluación específico para medir la capacidad de descomposición de problemas, aplicado antes del Taller de Abbot. Este módulo previo permitirá identificar si los estudiantes cuentan con las habilidades cognitivas mínimas para segmentar, organizar y abstraer elementos clave de un escenario narrativo antes de enfrentarse al modelado formal. La intención no es enseñar estrategias de descomposición, sino diagnosticar el nivel de razonamiento estructural que los participantes poseen al inicio. Con ello será posible reconocer con anticipación perfiles que requieren apoyo adicional y, a la vez, asegurar que la aplicación del método de Abbot se interprete correctamente como una prueba de modelado y no como un ejercicio para compensar deficiencias previas en análisis del problema.

Bibliografía

Almazroi Abdulwahab Ali and Abualigah, L. and A. M. A. and H. E. H. and A. A. Q. M. and E. M. A. (2021). Class Diagram Generation from Text Requirements: An Application of Natural Language Processing. In A. and M. M. and A. L. Kadyan Virender and Singh (Ed.), *Deep Learning Approaches for Spoken and Natural Language Processing* (pp. 55–79). Springer International Publishing. https://doi.org/10.1007/978-3-030-79778-2_4

Begum, M., Crossley, J., Strömbäck, F., Akrida, E., Alpizar-Chacon, I., Evans, A., Gross, J. B., Haglund, P., Lonati, V., Satyavolu, C., & Thorgeirsson, S. (2025). A Pedagogical Framework for Developing Abstraction Skills. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 258–299. https://doi.org/10.1145/3689187.3709613

Bencomo, N., Cabot, J., Chechik, M., Cheng, B., Combemale, B., Wasowski, A., & Zschaler, S. (2024). *Abstraction Engineering*. https://doi.org/10.48550/arXiv.2408.14074

Fatimah Saud, S., & Azma Nasruddin, Z. (2016). *Design of e-learning courseware for hearing impaired (HI) students*. 271–276. https://doi.org/10.1109/IUSER.2016.7857973

Fill, H. G., Cabot, J., Maass, W., & van Sinderen, M. (2024). AI-Driven Software Engineering – The Role of Conceptual Modeling. *Enterprise Modelling and Information Systems Architectures*, *19*, 1–11. https://doi.org/10.18417/emisa.19.1

Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in Mathematics Education*, *23*(2), 159–187. https://doi.org/10.1080/14794802.2020.1852104

Lehmann, T. (2025). Examining the interaction of computational thinking skills and heuristics in mathematical problem solving. *Research in Mathematics Education*, *27*, 1–22. https://doi.org/10.1080/14794802.2025.2460460

Ohue, M., Yasuo, N., & Takata, M. (2024). Innovations in mathematical modeling, AI, and optimization techniques. *The Journal of Supercomputing*, *81*. https://doi.org/10.1007/s11227-024-06861-9

Rao, T. S. S., & Bhagat, K. K. (2024). Computational thinking for the digital age: a systematic review of tools, pedagogical strategies, and assessment practices. *Educational Technology Research and Development*, *72*(4), 1893–1924. https://doi.org/10.1007/s11423-024-10364-y

Sattorova, Z., Hossain, R., Suryasa, W., Ugli, Y., & Rahman, F. (2025). *From Syntax To Semantics: AI assisted Computational Linguistics In The Era Of Large computational Language Models*. 1–6. https://doi.org/10.1109/ICCIES63851.2025.11032453

Wing, J. M. (2006). Computational thinking. *Commun. ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215