ISSN: 1692-7257 - Volumen 1 – Número 43 - 2024

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v1i43.2798

Visión de las competencias en arquitectura de software integrando las perspectivas de la industria y la academia

Insights into software architecture competencies integrating industry and academic perspectives

PhD. Wilson Libardo Pantoja Yépez[1], PhD. Andrés Fernando Solano A.[2], PhD. Ajay Bandi[3], PhD. Julio Ariel Hurtado Alegría[1]

[1] ***Universidad del Cauca,*** *Facultad de Ingeniería Electrónica y Telecomunicaciones, Grupo I+D IDIS, Popayán, Cauca, Colombia.*
[2] ***Universidad Autónoma de Occidente,*** *Facultad, Programa o Grupo de Investigación, Cali, Valle, Colombia.*
[3] ***Northwest Missouri State University,*** *Maryville, USA*

Correspondencia: *wpantoja@unicauca..edu.co*

Recibido: *8 noviembre 2023.* ***Aceptado:*** *10 enero 2024.* ***Publicado:*** *25 febrero 2024.*

Cómo citar: *W. L. Pantoja Yépez, A. F. Solano Alegría, A. Bandi, y J. A. Hurtado Alegría, «Visión de las competencias en arquitectura de software integrando las perspectivas de la industria y la academia», RCTA, vol. 1, n.º 43, pp. 9–23, feb. 2024. Recuperado de https://ojs.unipamplona.edu.co/index.php/rcta/article/view/9-23*

Resumen: (Propósito) La formación de un arquitecto de software es una labor compleja que requiere de una mezcla de experiencia y conocimiento especializado que es difícil lograr en el contexto universitario. Este artículo busca determinar las competencias mínimas que debe lograr un arquitecto de software cubriendo la expectativa de la industria, así como el contexto formativo de las universidades e instituciones de educación superior. (Métodos) Para identificar y documentar estas competencias, se realizó un ciclo de investigación-acción, en el cual se diseñó un estudio basado en encuestas y talleres en el que participaron ingenieros de software de la industria y profesores universitarios que imparten cursos relacionados con el diseño y evaluación de la arquitectura. Para dar contexto al estudio de caso se sitúa de forma específica el problema y se definen las preguntas de la investigación. En paralelo, se realizó una revisión de la literatura para profundizar en el tema de estudio para diseñar adecuadamente los instrumentos del estudio. De acuerdo al propósito y con el contexto literario establecido, se diseñó, ejecutó y reportó el estudio. Finalmente se realizó una reflexión tanto de los resultados como los aprendizajes cerrando el ciclo de investigación-acción. (Resultados) Como primer hallazgo, el estudio arroja un conjunto de 11 competencias esenciales a nivel de arquitectura de software que la industria espera de los egresados de las cuales todas son competencias técnicas y ninguna competencia blanda. Como segundo hallazgo, el estudio permitió determinar que las universidades en un 16.1% no abordan las competencias obligatorias y en un 11.7% poco se abordan. (Conclusión) La discrepancia entre lo que se enseña en las universidades y lo que la industria de software espera es un problema que se evidencia a través de este estudio. Alinear los cursos de arquitectura de software con los requisitos de la industria es crucial para los planes de estudio de ciencias de la computación, ingeniería de sistemas y programas relacionados. Sin embargo, desarrollar en los estudiantes, las competencias demandadas por la industria, plantea numerosos restos. Conocer las competencias que requiere la industria es el primer paso para crear cursos que ayuden a la empleabilidad de

los recién egresados. La identificación de qué competencias se pueden ir incorporando con menor esfuerzo y mayor eficacia permiten trazar una ruta en la que las universidades puedan iniciar ese camino hacia el cubrimiento de las expectativas de la industria de software.

Palabras clave: Competencias en el aprendizaje, arquitectos de software, investigación-acción; industria de software.

Abstract: (Purpose) Training a software architect is a complex task requiring a mix of experience and specialized knowledge that is difficult to achieve in the university context. This article seeks to determine the minimum competencies a software architect should achieve, covering industry expectations and the training context of universities and higher education institutions. (Methods) We conducted an action research cycle to identify and document these competencies, in which a study was designed based on surveys and workshops involving software engineers from industry and university professors who teach courses related to architecture design and evaluation. We defined the problem and research questions to contextualize the case study. A literature review was conducted to deepen the study topic and adequately design the study instruments. According to the purpose and with the established literature context, the study was designed, executed, and reported. Finally, a reflection on the results and the lessons learned was carried out, closing the action research cycle. (Results) As a first finding, the study shows a set of 11 essential competencies at the software architecture level that the industry expects from graduates, all of which are technical competencies and none of which are soft competencies. As a second finding, the study determined that 16.1% of universities do not address the mandatory competencies, and 11.7% do not address them. (Conclusion) The discrepancy between what is taught in universities and what the software industry expects is a problem evidenced throughout this study. Aligning software architecture courses with industry requirements is crucial for computer science, systems engineering, and related program curricula. However, imparting industry-demanded competencies to undergraduate students poses numerous challenges. Knowing the skills required by industry is the first step in creating courses that will help the employability of recent graduates. Identifying which competencies can be incorporated with less effort and greater efficiency allows us to trace a route in which universities can start this path towards meeting the expectations of the software industry.

Keywords: Software architecture competencies, recent graduates, software engineering, action research, software industry.

1. INTRODUCCIÓN

La Arquitectura de Software (AS) es un área fundamental de la ingeniería de software para asegurar la calidad de los productos software, de ahí que tanto la academia como la industria han centrado sus expectativas en el diseño un buen currículo [1]. Sin embargo, enseñar arquitectura de software sigue siendo una tarea difícil, el docente necesita abordar problemas con una complejidad similar al mundo real, trabajo en equipo, brindar un acompañamiento especial, entre otros desafíos [2].

El rol del arquitecto es muy desafiante en cualquier proyecto de software. Un arquitecto podría ser una persona, equipo u organización que diseñe la arquitectura del sistema (norma IEEE 1471-2000 [3]). La motivación básica de un arquitecto de software es desarrollar una arquitectura para el sistema. Sin embargo, además, un arquitecto de software desempeña cualquier otro papel importante en el ciclo de vida de un proyecto software. Un arquitecto de software comprende el proceso de desarrollo, tiene el conocimiento del dominio de negocio y, además, cuenta con las habilidades de análisis y programación. Un arquitecto es un buen comunicador, conoce las políticas de la organización y juega un papel importante en la toma de decisiones durante el proyecto. Un arquitecto desempeña el papel de un catalizador para mejorar la comunicación y desarrollar el entendimiento entre clientes y desarrolladores [4] [5]. Se considera que un arquitecto es un líder técnico del proyecto de software, ya que en todas las decisiones técnicas es

imprescindible la participación del arquitecto como mediador de intereses [6].
Las competencias que deben lograr los estudiantes en el campo de la AS son muy amplias y complejas. El primer paso para un diseño curricular en AS que disminuya la brecha entre lo que se enseña en las aulas y lo que la industria demanda, es conocer con precisión cuáles son las competencias mínimas que la industria de software espera de los recién egresados. El presente artículo lleva a cabo una metodología compuesta por una serie de pasos para obtener ese listado de competencias. En esta sección se presenta la introducción. La sección 2 presenta el problema. La sección 3 describe los trabajos relacionados, mientras que la sección 4 presenta el diseño de la experiencia. La sección 5 presenta el análisis de resultados. Finalmente, la sección 6 incluye un conjunto de conclusiones y trabajo futuro.

2. CONCEPTOS FUNDAMENTALES DE LA FORMACIÓN EN ARQUITECTURAS DE SOFTWARE

Esta sección incluye los conceptos fundamentales alrededor de la formación en AS.

2.1. Arquitectura de software

El concepto de arquitectura de software está en continua evolución, y es esencial comprender las diversas definiciones que aparecen en la literatura. Una definición moderna podría implicar un conjunto de decisiones fundamentales relativas a la estructura del sistema de software, que guía el diseño y la construcción del sistema [7] [8] [9]. Esta estructura incluye la organización de los componentes, la forma en que se comunican y la distribución de responsabilidades entre ellos [10]. La arquitectura del software también aborda cuestiones relacionadas con la calidad de los atributos no funcionales, como el rendimiento del sistema, la escalabilidad, la seguridad, la facilidad de uso y la mantenibilidad [4].

Algunas de las características de la AS que se mencionan con frecuencia son [10]: (i) es una abstracción primaria del sistema que los involucrados usan para pensar, diseñar, codificar y comunicarse en términos de grandes bloques conceptuales, (ii) promueve la reutilización de alto nivel y la reutilización de componentes, (iii) influye en la productividad del desarrollo al reutilizar grandes marcos para respaldar la construcción de líneas de productos, (iv) asegura la calidad a lo largo del ciclo de vida del software al tratar explícitamente los atributos de calidad como la modificabilidad, la portabilidad, escalabilidad y seguridad. A medida que crece el tamaño del sistema, los desarrolladores pueden rastrear las soluciones a estos problemas mediante el análisis de la arquitectura.

2.2. Atributos de Calidad

Los atributos de calidad se refieren a los rasgos específicos que satisface un producto de software, y cada atributo se asocia con métricas específicas que definen los niveles de calidad de un producto de software [11]. Los atributos de calidad incluyen la seguridad, la fiabilidad, el rendimiento y la interoperabilidad, que surgen de complejas normas empresariales y preocupaciones de calidad [12]. La gestión adecuada de estos atributos de calidad es crucial, ya que un manejo inapropiado supone un riesgo empresarial significativo. El arquitecto debe tener en cuenta los posibles conflictos entre atributos de calidad y resolverlos mediante compensaciones.

2.3. Patrones de Arquitectura

Los patrones de arquitectura se refieren a estructuras comunes de solución a problemas de diseño similares. Cada patrón describe una estructura general del sistema de software o un comportamiento de alto nivel que debe satisfacer las funcionalidades, cualidades y restricciones de un producto. Estos patrones se eligen en función de las primeras decisiones de diseño, como la satisfacción de los requisitos funcionales, los requisitos no funcionales y las restricciones del sistema [13].

2.4. Competencias, habilidades y conocimiento

Una *competencia* se define como la *habilidad* para hacer algo [14]. El *conocimiento* puede ser entendido como el entendimiento teórico o práctico. Para un individuo, la competencia se compone de conocimientos y habilidades.

Según Bass et al., en su libro Software Architecture in Practice [10], definen deberes, habilidades y conocimientos. Los deberes, las habilidades y el conocimiento forman una tríada sobre la cual se apoyan las competencias arquitectónicas de los ingenieros. Las habilidades y conocimientos soportan la ejecución de las competencias (funciones u obligaciones) tal como lo muestra la Fig. 1. Un ejemplo de estos tres conceptos:

- "Diseñar una arquitectura" es un deber.
- "Pensar de manera abstracta" es una habilidad.
- "Patrones y tácticas" es parte de un cuerpo de conocimiento.

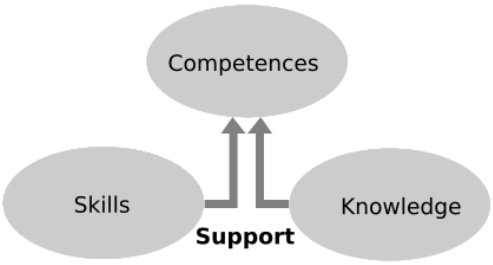

Fig. 1. *Las habilidades y conocimientos soportan la ejecución de las competencias*
Fuente**:** [10].

3. EL PROBLEMA

Tradicionalmente, los programas de pregrado en ingeniería, como Ciencias de la Computación e Ingeniería, Tecnología de la Información, Ingeniería de Software e Ingeniería de Sistemas, incluyen muchos cursos en los cuales se desarrollan habilidades y se imparten conocimientos técnicos relacionados con la construcción de software a través del uso de lenguajes de programación y plataformas de desarrollo [15]. Sin embargo, los estudiantes de estos programas presentan poco conocimiento sobre AS y problemas de diseño [16] a pesar de su creciente importancia para la industria de software. Especialmente los recién graduados carecen de las habilidades suficientes la toma adecuada de decisiones de diseño, aplicando prácticas y conocimientos relacionados al diseño de software en el contexto empresarial [17].

La literatura existente sobre la educación y la práctica de la AS apuntan a diferentes razones para esta falta de habilidades relacionadas con la arquitectura y el diseño de software [18]. En primer lugar, existe una diferencia considerable entre la percepción académica de la AS y su práctica en el sector industrial [19]. A veces, los problemas que la industria consideran más críticos y desafiantes no reciben la debida importancia para la investigación o la formación en las universidades [20]. En segundo lugar, existe una brecha entre las habilidades que la industria espera de los graduados en áreas de Ingeniería de Software y las habilidades que se enseñan en los planes de estudios [21]. En tercer lugar, muchas instituciones no tienen una visión clara sobre los temas en los que se debe capacitar a los arquitectos de software [16].

Para diseñar el currículo de un curso de Arquitectura de Software es importante alinear los objetivos del curso con las expectativas de la Industria de Software. Para ello, es fundamental definir las competencias que se esperan desarrollar.

Una competencia se entiende como la sumatoria de conocimientos y habilidades, sin embargo, una competencia es más que esto; implica la habilidad para satisfacer demandas complejas movilizando y recurriendo a destrezas y actitudes en un contexto particular [22]. Si nos enfocamos en las competencias, habilidades y conocimientos que un estudiante debe desarrollar en el área AS, nos encontramos con una amplia variedad. Por ejemplo:

- La principal habilidad del arquitecto es *diseñar, modelar, analizar y evaluar arquitecturas de software* [23] [24]. El arquitecto debe saber cómo aplicar patrones y marcos para crear aplicaciones de calidad [25]. La arquitectura de sistemas de software complejos a gran escala, que tienen muchos requisitos y millones de líneas de código, requiere habilidades muy altas de modelado y abstracción [26].
- El *conocimiento de múltiples tecnologías* le permite al arquitecto de software elegir las tecnologías apropiadas para el proyecto. Aunque los arquitectos de software no necesitan ser expertos en tecnología, el arquitecto debe mantenerse actualizado sobre las tendencias tecnológicas [27] [28].
- *Comprender el dominio en el que vivirá un sistema*, esto implica comprender el entorno empresarial, social y operativo en el que debe operar un sistema [29] .
- Las *habilidades analíticas* son esenciales para que el arquitecto de software comprenda rápidamente el problema, diagnostique las posibles causas raíz y tome decisiones importantes para el proyecto [19] [30]. Además, los arquitectos requieren la capacidad de encontrar las causas raíz de los problemas de alto nivel en los diseños existentes, como por qué un sistema funciona demasiado lento o no es seguro [28].
- Se requiere la *capacidad de investigación* para comprender situaciones complejas y luego resolver problemas [31] [32].
- Aunque los arquitectos no deben ser expertos en programación, deben tener *habilidades mínimas de programación* para comunicarse con los desarrolladores [33] [31].

- La *toma decisiones arquitectónicas* es otra habilidad fundamental. Un arquitecto debe aprender a tomar decisiones de diseño en ambientes donde se desconoce mucho, donde no hay tiempo suficiente para explorar todas las alternativas y donde hay presión para llevar a cabo [19]. Además, el arquitecto debe tomar decisiones de forma colaborativa.
- Se requiere un *pensamiento sistémico y holístico* y consideran los problemas desde diferentes perspectivas [28].
- Además de lo anterior, se requieren *habilidades de comunicación*, tales como hablar, escribir y hacer presentaciones para abordar problemas complejos con un diseño aparentemente simple que es fácil de entender [29] [34]. Los arquitectos de software supervisan y trabajan en estrecha colaboración con otros miembros del equipo de desarrollo (*Trabajo en equipo*), como los programadores [35] [36]. Finalmente, la *negociación y el liderazgo* son esenciales para que un arquitecto dirija, presente, negocie y justifique sus diseños y decisiones arquitectónicas [37] [6].

Desarrollar todas las habilidades anteriores seguramente requiere mucho tiempo y experiencia. Por lo tanto, los objetivos de un curso de AS para estudiantes de pregrado deben reconocer las limitaciones de la audiencia a la que se dirige y trabajar con los recursos que el docente dispone. El primer paso para un diseño curricular en AS que disminuya la brecha entre lo que se enseña en las aulas y lo que la industria demanda, es conocer con precisión cuáles son las competencias mínimas que la industria de software espera de los recién egresados. Por consiguiente, las preguntas de investigación que nos planteamos son:

- ¿Cuáles son las competencias mínimas a nivel de AS que la industria espera de un recién egresado?
- ¿De qué manera la academia está abordando esas competencias definidas por la industria?

4. TRABAJOS RELACIONADOS

Niño & Anaya propone una reforma curricular del área de ingeniería de software en programas de ingeniería de sistemas [38]. El principal producto es la definición de un mapa de competencias profesionales del área de ingeniería de software, estructurado en competencias de primero y segundo nivel. También logra identificar las asignaturas centrales que contribuyen con el desarrollo de las competencias identificadas.

Garousi et al. realizan una revisión de la literatura de estudios que abordan la dificultad que tienen graduados de ingeniería de software al iniciar sus carreras profesionales, debido a la desalineación de las habilidades aprendidas en su formación universitaria con lo que necesita la industria [18]. Este estudio permite a los educadores y gerentes de contratación adaptar sus esfuerzos de educación/contratación para preparar mejor a la fuerza laboral de ingeniería de software.

Rupakheti & Chenoweth describen la historia de diez años de enseñanza de un curso de AS de pregrado, como parte de un programa de licenciatura en ingeniería de software [29]. Se incluyen descripciones de lo que perciben que son los objetivos realistas de la enseñanza de la AS en este nivel. Describen que el objetivo primordial del curso es preparar a los estudiantes para situaciones de diseño de alto nivel en la industria, mediante el empleo de tecnologías y procesos nuevos y generalizados en sus proyectos, y la corrección de problemas arquitectónicos en sistemas existentes.

Kiwelekar & Wankhede presentan un conjunto de objetivos de aprendizaje y su clasificación utilizando la Taxonomía de Bloom Revisada (RBT) [39]. El análisis pone de manifiesto las habilidades cognitivas genéricas requeridas para el modelado de arquitectura. Uno de los beneficios potenciales de la clasificación de los objetivos de aprendizaje es que los diferentes procesos educativos, como la instrucción, el aprendizaje y la evaluación, se pueden alinear de manera efectiva utilizando la clasificación de los objetivos de aprendizaje presentada en este estudio.

Paulisch et al. [38] estudiaron la descripción detallada de las funciones de un arquitecto, incluidas las competencias que debe adquirir.

Teniendo en cuenta los estudios relacionados, logramos identificar que no hay un listado de competencias mínimas en temas de AS alineadas con las necesidades de la industria de software, que le permitan posteriormente a los docentes caracterizar las necesidades y estrategias de formación del rol de arquitecto de software en programas de pregrado.

5. METODOLOGÍA

En el marco de esta investigación utilizamos el método investigación-acción propuesto por Putman & Rock [40], que realiza de manera iterativa las etapas de planificación, actuación y reflexión.

Para conocer las competencias mínimas en el área de AS que la industria espera en los recién egresados de programas de Ingeniería de sistemas y afines, llevamos a cabo un primer ciclo de investigación-acción siguiendo los siguientes pasos:

- Identificar el problema y definir las preguntas de investigación que guiarán el proceso de investigación-acción.
- Analizar la teoría para desarrollar un conocimiento profundo y sintético del tema de investigación.
- Crear un plan de investigación.
- Ejecutar el plan y analizar los datos recolectados.
- Reflexionar sobre los resultados de la investigación-acción.
- Desarrollar un siguiente ciclo basado en los datos de investigación.

A continuación, se explica en detalle cada uno de los pasos anteriores.

5.1. Paso 1 - Identificar el problema y definir las preguntas de investigación que guiarán el proceso de investigación-acción

El problema a resolver y las preguntas de investigación están descritas en la Sección 3 y están enmarcadas en las dificultades en la formación de competencias de AS en pregrado alineadas con las expectativas de la industria de software.

5.2. Paso 2 - Analizar la teoría para desarrollar un conocimiento profundo y sintético del tema de investigación

Realizamos una revisión de la literatura buscando experiencias de cursos de arquitecturas de software y a partir de ahí logramos crear una base conceptual sólida, definiendo conceptos claves como competencia, habilidad y conocimiento. Además, logramos recopilar las competencias que se buscan desarrollar en los cursos.

5.3. Paso 3 - Crear un plan de investigación

Acorde a las preguntas de investigación establecidas en este ciclo, que buscan conocer las competencias en el área de AS que la industria espera de los recién egresados, seleccionamos tres métodos de recopilación de datos adecuados: encuesta, workshop y focus group. La Tabla 1, muestra los instrumentos específicos que utilizamos para recolectar los datos. Cada instrumento permite ir encontrando y refinando la lista de competencias esperada tal como lo muestra la Fig. 2.

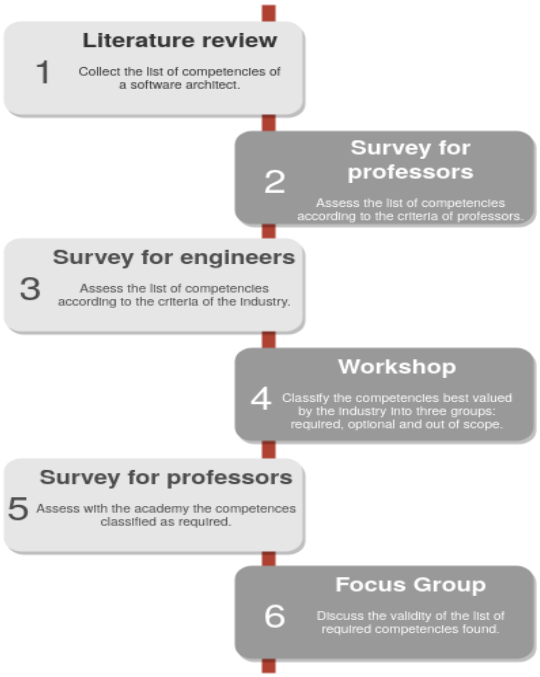

Fig. 2. *Instrumentos de recolección de datos y su propósito.*
Fuente: *elaboración propia.*

Finalmente, elegimos las técnicas y las herramientas de análisis estadístico adecuadas para la investigación:

- Google Forms para capturar datos de las encuestas.
- Jamboard para el workshop con ingenieros de la industria para clasificar las competencias.
- Studio R para procesar los datos y generar gráficos estadísticos.

Tabla 1*: Instrumentos de recolección de datos y su propósito.*

No	Instrumento	Objetivo
1	Revisión de la literatura	Recolectar la lista de competencias de un arquitecto de software utilizando la revisión de la literatura.
2	Encuesta de valoración a docentes	Valorar la lista de competencias según el criterio de los docentes

3	Encuesta de valoración a ingenieros	Valorar la lista de competencias según el criterio de la industria.
4	Workshop con ingenieros de la industria	Clasificar las competencias mejor valoradas por la industria en tres grupos: obligatorias, opcionales y fuera de alcance de un curso de pregrado.
5	Encuesta de competencias obligatorias a docentes	Valorar con la academia las competencias clasificadas como obligatorias.
6	Focus group con docentes	Debatir la validez la lista de competencias obligatorias encontradas

Fuente: *elaboración propia*

5.4. Paso 4 - Ejecutar el plan y analizar los datos recolectados

Después de planificar las actividades a realizar en el ciclo de investigación-acción ejecutamos las acciones. A continuación, explicamos cada uno de los instrumentos ejecutados y los resultados obtenidos.

5.4.1. Revisión de la literatura

Después de hacer una revisión de la literatura obtuvimos una lista amplia de 35 competencias de los arquitectos de software (ver Tabla 6 de los Anexos). Estas competencias están clasificadas en las categorías: Creación de una Arquitectura, Análisis y Evaluación De Una Arquitectura, Documentación de Arquitectura, Trabajando con sistemas existentes, Otras competencias, Gestión de requisitos, Implementación del producto, Testing del producto y Selección de herramientas y tecnología. En los pasos siguientes se busca analizar este listado y filtrar las competencias que la industria espera en recién egresados. En las secciones posteriores seguiremos utilizando los mismos identificadores de las competencias dados en la Tabla 6**.**

5.4.2. Encuesta de valoración a docentes

El listado de competencias de la Tabla 6 fue sometido a valoración por un grupo de nueve docentes del área relacionadas con Ingeniería de Software. Se le pidió al grupo de docentes que evalúen las competencias de Arquitectura de Software que adquieren sus estudiantes durante el proceso de formación en pregrado en cada una de sus universidades. En cada competencia se les pidió valorar el nivel de importancia (o dedicación) que se le da en el Programa, siguiendo la siguiente escala: (1) No es importante, (2) Poco importante, (3) Neutral, (4) Importante, (5) Muy importante. Los resultados de esta experiencia se pueden apreciar en la Fig. 3**.**

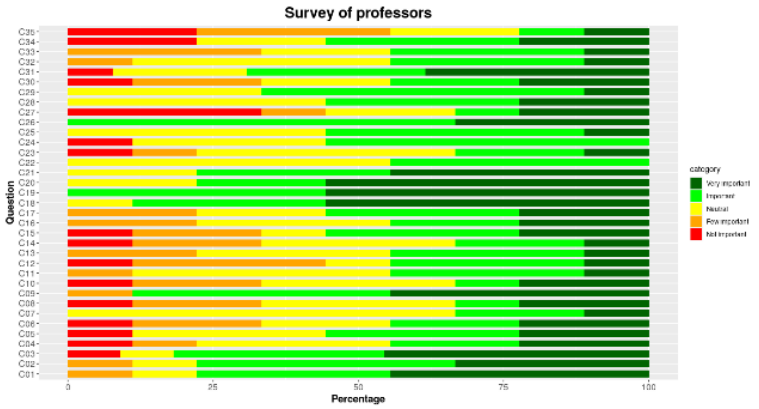

Fig. 3. *Valoración de las 35 competencias de Arquitectura de Software por parte de los docentes universitarios*
Fuente: *elaboración propia.*

Posteriormente, pasamos las valoraciones de las competencias de la Fig. 3 mediante la siguiente fórmula:

Puntaje = VotosNoImportante x 0 + VotosPocoImportante x 1 + VotosNeutral x 2 + VotosImportante x 3 + VotosMuyImportante x 4

De tal manera que las 10 competencias más importantes según la valoración de los docentes se pueden apreciar en la Tabla 2

Tabla 2: *Las 10 competencias mejor valoradas según la encuesta de los docentes*

No	Competencia	Puntaje
1	C03	34
2	C19	32
3	C18	31
4	C20	30
5	C26	30
6	C09	29
7	C21	29
8	C01	28
9	C02	27
10	C28	25

Fuente: *elaboración propia*

5.4.3. Encuesta de valoración a ingenieros

El listado de competencias de la Tabla 6 fue sometido a valoración por un grupo de 21 ingenieros de la industria que se desempeñan como arquitectos de software o tareas afines. Se le pidió al grupo de ingenieros que evalúen las competencias de Arquitectura de Software que esperan de un recién egresado de un programa de ingeniería de sistemas o carreras afines. En cada competencia se les pidió valorar el nivel de importancia para la industria,

acorde a la siguiente escala: (1) No es importante, (2) Poco importante, (3) Neutral, (4) Importante, (5) Muy importante. Los resultados de esta experiencia se pueden apreciar en la Fig. 4**.**

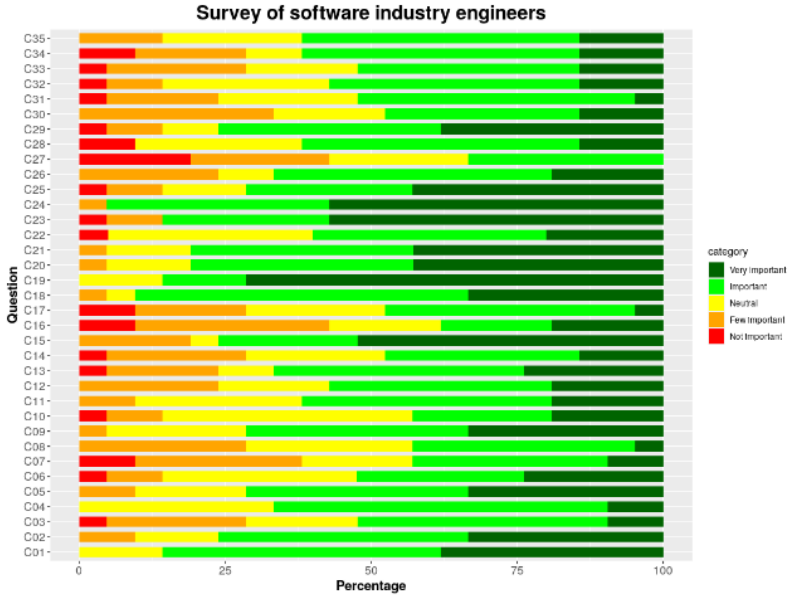

Fig. 4. *Valoración de las 35 competencias de Arquitectura de Software por parte de los ingenieros de la industria de Software.*
Fuente: *elaboración propia.*

Posteriormente, pasamos las valoraciones de las competencias de la Fig. 4, mediante la siguiente fórmula:

$$Puntaje = VotosNoImportante\ x\ 0 + VotosPocoImportante\ x\ 1 + VotosNeutral\ x\ 2 + VotosImportante\ x\ 3 + VotosMuyImportante\ x\ 4$$

De tal manera que las 10 competencias más importantes según la valoración de los ingenieros se pueden apreciar en la Tabla 3**.** Además, colocamos en negrita las competencias que coinciden con la valoración de los docentes: C19, C18, C20, C21 y C02. La Fig. **5** muestra gráficamente los dos conjuntos de competencias y sus coincidencias.

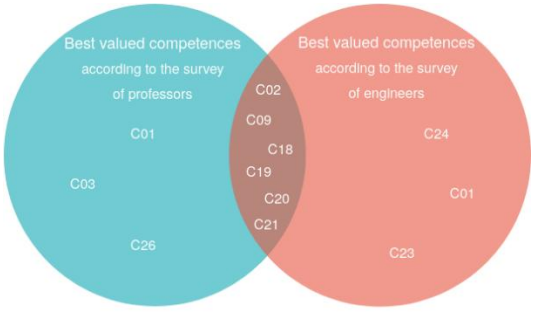

Fig. 5. *Competencias mejor valoradas según la academia y la industria.*
Fuente: *elaboración propia.*

Tabla 3*: Las 10 competencias mejor valoradas según la encuesta de los ingenieros. En negrita las coincidencias entre la industria y la academia.*

No	Competencia	Puntaje
1	**C19**	75
2	C24	73
3	C01	78
4	C23	68
5	**C18**	67
6	**C20**	67
7	**C21**	67
8	C15	65
9	**C02**	63
10	C09	63

Fuente: *elaboración propia*

5.4.4. Workshop con ingenieros de la industria

El siguiente paso fue clasificar las competencias mejor valoradas por la industria en tres grupos: obligatorias, opcionales y fuera de alcance de un curso de pregrado. El listado de competencias de la Tabla 6 es demasiado amplia para ser abordada en una carrera universitaria de pregrado, por lo tanto, esta actividad busca clasificar y reducir ese listado buscando las competencias más relevantes para un curso de formación.

Esta actividad colaborativa la realizamos de forma virtual y síncrona utilizando Google Meet y una Pizarra Jamboard con “adhesivos” por cada competencia. Además, la actividad contó con 4 ingenieros de sistemas con amplia experiencia en

tareas relacionadas con AS. El perfil de los ingenieros se puede apreciar en la

Tabla *7* en los Anexos.

Los pasos que se llevaron a cabo en esta actividad fueron los siguientes:

- Clasificar las competencias de los adhesivos dados en los tres grupos: obligatorias, opcionales, y fuera de alcance para un recién egresado. Cada uno de los cinco integrantes puede tomar la decisión de ubicar el adhesivo en la columna correspondiente. En caso de dudas puede apoyarse en las opiniones de sus colegas. También, se pueden apoyar en los resultados de la encuesta. Tiempo máximo 10 minutos.
- Revisar en grupo la clasificación realizada y hacer ajustes que se consideren necesarios. Es un espacio para refinar el trabajo de grupo. Tiempo máximo 15 minutos.
- Socializar los resultados. Es una breve justificación de la clasificación realizada hacia el investigador. La puede hacer un integrante con apoyo de los demás. Tiempo máximo 5 minutos.

El resultado final de esta actividad se puede apreciar en la Fig. 6**.**

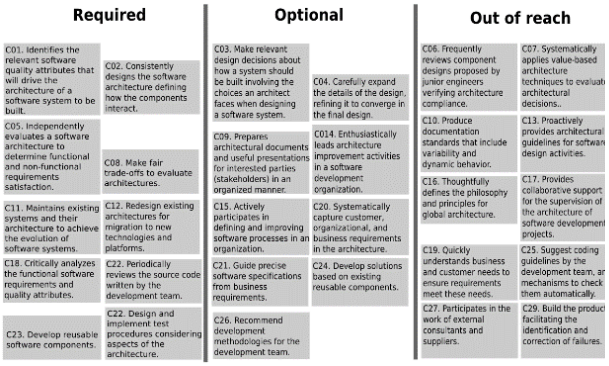

Fig. 6*. Resultado final del workshop para clasificar las competencias*
Fuente: *elaboración propia.*

5.4.5. Encuesta de competencias obligatorias a docentes

La finalidad de esta última encuesta fue conocer de qué manera están siendo abordadas en las Universidades, las competencias clasificadas como obligatorias resultado de la actividad colaborativa. Los pasos ejecutados para esta encuesta fueron los siguientes.

- Buscar en internet universidades regionales, nacionales e internacionales con programas de Ingeniería de Sistemas y afines.
- Por cada Universidad buscar en la web los perfiles de los docentes que tuvieran el perfil de enseñanza en Arquitectura de Software.
- Escribir correos a los docentes seleccionados invitándolos a participar de la encuesta.

La encuesta la llenaron 18 universidades, un docente por cada Institución (ver Tabla 4).

Tabla 4: *Docentes que participaron en la encuesta de competencias obligatorias*

No	Universidad	Ciudad/País
1	Universidad Autónoma de Occidente	Cali-Colombia
2	Benemérita Universidad Autónoma de Puebla	México
3	Universidad Cooperativa de Colombia	Popayán-Colombia
4	Corporación Universitaria del Caribe	Sincelejo-Colombia
5	Corporación Universitaria Comfacauca	Popayán-Colombia
6	Institución Universitaria Colegio Mayor del Cauca	Popayán-Cauca
7	Universidad del Valle	Cali-Colombia
8	Pontificia Universidad Javeriana	Bogotá-Colombia
9	Universidad San Buenaventura	Cali-Colombia
10	Universidad de Antioquia	Medellín-Colombia
11	Universidad Nacional de Colombia	Bogotá-Colombia
12	Universidad del Cauca	Popayán-Colombia
13	Universidad de Extremadura	España
14	Universidad de Boyacá	Tunja-Colombia
15	Universidad Santo Tomás Seccional Tunja	Tunja-Colombia
16	Universidad de Los Andes	Bogotá-Colombia
17	Universidad Pedagógica y Tecnológica de Colombia	Tunja-Colombia
18	Universidad Nacional de La Plata	Argentina

Fuente: *elaboración propia*

La pregunta que se formuló en la encuesta fue: "*Respecto a las competencias obligatorias, queremos conocer de qué manera están siendo abordadas en su Institución respondiendo según la escala de Likert: (1) No se aborda, (2) Poco se aborda, (3) Neutral, (4) Muy abordada, (5) Totalmente abordada*". Las respuestas a esta encuesta se pueden ver en la Fig. **7** donde se puede

apreciar que todas las competencias tienen algún grado de "No se aborda" y "Poco se aborda". Calculando totales, la Tabla 5 muestra los porcentajes de las competencias en sus cinco categorías. Se puede apreciar que las universidades en un 16.1% no abordan las competencias obligatorias y en un 11.7% poco se abordan.

Tabla 5: *Docentes que participaron en la encuesta de competencias obligatorias*

Categoría	Porcentaje
No se aborda	16.1%
Poco se aborda	11.7%
Neutral	22.8%
Muy abordada	22.8%
Totalmente abordada	26.7%

Fuente: *elaboración propia*

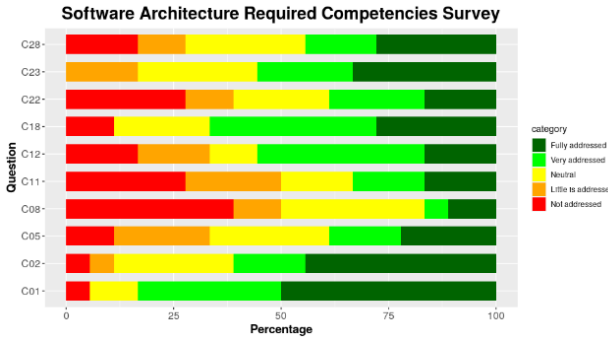

Fig. 7. *Valoración de las 10 competencias de Arquitectura de Software por parte de varias universidades.*
Fuente: *elaboración propia.*

5.4.6. Focus group con docentes

El objetivo de este focus group fue compartir las experiencias en la manera cómo se enseña AS en pregrado entre un grupo de docentes de varias universidades. Se organizó un listado de docentes a nivel regional, nacional e internacional y se hizo la invitación del evento a través de correo electrónico con un documento adjunto que contenía detalles del encuentro (introducción, roles, agenda y preguntas a trabajar). A pesar que cinco docentes en un primer momento aceptaron la invitación, al momento del encuentro sincrónico únicamente se presentaron docentes de tres Universidades: Universidad del Cauca, Institución Universitaria Colegio Mayor del Cauca y la Universidad Nacional de la Plata (Argentina).

Durante el focus group se debatió las siguientes preguntas:

1. *Pregunta 1:* ¿De qué manera están desarrollando las competencias de AS en sus Universidades? (En qué cursos, qué semestres, qué temas se abordan, qué estrategias usan para recrear un ambiente real, entre otros).
2. *Pregunta 2:* ¿Qué opinas de estos resultados, estás de acuerdo, y qué cambiarías de esta clasificación? Indagamos a un grupo de profesionales de la industria de software sobre cuáles son las competencias en temas de AS que esperan de un recién egresado de pregrado. Las competencias fueron clasificadas en tres categorías: a) obligatorias, b) opcionales y c) fuera del alcance para un recién egresado. Los resultados de este experimento se pueden apreciar en la Fig. **6**. ¿Qué opinan de estos resultados? ¿Están de acuerdo? ¿Qué cambiarían de esta clasificación?

A continuación, un resumen de los puntos más importantes de cada una de las dos preguntas.

Pregunta 1:

- "En nuestra institución las carreras tienen un fuerte énfasis en desarrollo de software, pero su orientación está hacia el aprendizaje de la codificación en varios lenguajes de programación. El tema de AS no tiene el suficiente énfasis".
- "En nuestra Universidad existen tres asignaturas de Ingeniería de Software además de los cursos llamados Proyecto 1 y 2. El tema de la AS se lo estudia con mayor énfasis en el curso de Ingeniería de Software II. Sin embargo, el tema es tan amplio que no se alcanza a cubrir en su totalidad".
- "En nuestra Institución tenemos dos carreras relacionadas con Ingeniería de Software y existen cursos que están organizados acorde al ciclo de vida del software, es decir, tenemos las asignaturas de análisis, diseño, implementación, verificación y mantenimiento. Sin embargo, el tema de Arquitectura de Software se lo trata en un capítulo en un mes dentro la asignatura de Diseño de Software".

Pregunta 2:

- "Estoy de acuerdo con todas las tarjetas que se han definido como obligatorias, opcionales y fuera de alcance. Además, yo diría que las competencias se pueden clasificar en dos tipos. La primera las relacionadas con el desarrollo de un sistema nuevo, y la segunda, con el mantenimiento de un sistema software junto con su arquitectura".
- "La competencia relacionada con hacer revisiones del código fuente, en principio me

causa algo de extrañeza, pero podría estar relacionada con revisar que la codificación esté correcta acorde a la arquitectura propuesta".

- "Yo diría que desarrollar todas las habilidades obligatorias, resulta algo costoso para estudiantes de pregrado. Por ejemplo, solamente para entender el valor de hacer una buena especificación de requisitos tenemos que exponer a nuestros chicos a dominios complejos donde aparece términos no conocidos para ellos. Si fuera un dominio simple los estudiantes no ven la necesidad de especificar, sino hacer la codificación directamente. De igual forma ocurre con la AS, se requiere involucrar proyectos grandes para que se vea la importancia de los distintos estilos arquitectónicos".
- "En alguna ocasión intentamos exponer a los estudiantes a modificar un sistema existente (desarrollado por los docentes) para que ellos estudien y modifiquen la arquitectura. Esta experiencia fue muy costosa de desarrollar en cada semestre para evitar que los estudiantes se pasen los trabajos. Esto hizo que en la actualidad trabajemos con proyectos nuevos, pero con requisitos cambiantes durante el transcurso del semestre".
- "En el ámbito empresarial es cada vez más frecuente que los desarrolladores migren hacia otra empresa que les ofrece mejor salario. Esto provoca que tenga que buscarse nuevos desarrolladores y enfrentarlos a modificar sistemas existentes".
- "Opino que la competencia 19: *Comprende rápidamente las necesidades del negocio y del cliente para asegurar que los requisitos satisfagan estas necesidades* no debería estar en la columna Fuera de Alcance sino en Obligatorias. Esta competencia es importante para apropiar rápidamente el dominio de negocio".
- "Al comparar las competencias clasificadas como obligatorias por el sector de la industria, siento que muchas de ellas nuestros estudiantes no las alcanzan a desarrollar en las distintas asignaturas y proyectos que se desarrollan en clase. Nos queda la preocupación sobre cómo llegar que los estudiantes lleguen hasta allá".

5.5. Paso 5 - Reflexionar sobre los resultados de la investigación-acción

Finalizado el ciclo de investigación-acción hemos encontrado respuesta las dos preguntas de investigación. Como primer hallazgo, tenemos un conjunto de competencias mínimas (de la categoría obligatorias) a nivel de AS que la industria espera de un recién egresado:

1. C01: Identifica claramente los atributos de calidad relevantes del software que conducirán la arquitectura de un sistema de software a construir.
2. C02. Diseña consistentemente la AS definiendo cómo los componentes interactúan entre sí.
3. C05: Evalúa independientemente una AS para determinar la satisfacción de los requisitos funcionales y no funcionales.
4. C08. Realiza imparcialmente un análisis de compensación (trade-off) para evaluar arquitecturas.
5. C11. Mantiene los sistemas existentes y su arquitectura para lograr la evolución de los sistemas software.
6. C12. Rediseña las arquitecturas existentes para la migración a nuevas tecnologías y plataformas.
7. C18. Analiza críticamente los requisitos de software funcionales y de atributos de calidad.
8. C19. Comprende rápidamente las necesidades del negocio y del cliente para asegurar que los requisitos satisfagan estas necesidades.
9. C22. Realiza periódicamente revisiones del código fuente escrito por el equipo de desarrollo.
10. C23. Desarrolla componentes de software reutilizables.
11. C28. Diseña e implementa procedimientos de prueba considerando aspectos de la arquitectura (tipos de componentes/servicios, integración).

Al conjunto de competencias le hemos agregado la competencia C28 por sugerencia del focus group de los docentes.

Como segundo hallazgo, y dando respuesta a la Pregunta 2 de investigación, tenemos que en general, desde la academia es difícil abarcar la enseñanza de este conjunto de competencias. Esto se evidencia tanto con la encuesta de los docentes como con las respuestas del focus group. Se pudo evidenciar con la encuesta que las universidades en un 16.1% no abordan las competencias obligatorias y en un 11.7% poco se abordan.

5.6. Paso 6 - Desarrollar un siguiente ciclo basado en los datos de investigación

Este ciclo de investigación-acción será el insumo principal para desarrollar el segundo ciclo de investigación-acción, el cual está relacionado con la creación de unos patrones de formación que guiarán

al docente en el diseño de cursos de AS para programas de pregrado.

6. CONCLUSIONES Y TRABAJO FUTURO

Un curso de AS acorde a las necesidades de la industria es algo esencial en los planes de estudio de programas de ingeniería de sistemas y afines. Sin embargo, formar estudiantes de pregrado con las competencias que demanda la industria tiene muchos desafíos. La discrepancia entre lo que se enseña en las universidades y lo que la industria de software espera es un problema que se evidencia a través de este estudio. Alinear los cursos de arquitectura de software con los requisitos de la industria es crucial para los planes de estudio de ciencias de la computación, ingeniería de sistemas y programas relacionados. Sin embargo, impartir las competencias demandas por la industria a los estudiantes universitarios plantea numerosos retos. Conocer las competencias que requiere la industria es el primer paso para crear cursos que ayuden a la empleabilidad de los recién egresados. La identificación de qué competencias se pueden ir incorporando con menor esfuerzo y mayor eficacia permiten trazar una ruta en la que las universidades puedan iniciar ese camino hacia el cubrimiento de las expectativas de la industria de software.

Teniendo en cuenta los estudios relacionados, logramos identificar que no hay un listado de competencias mínimas en temas de AS alineadas con las necesidades de la industria de software, que le permitan posteriormente a los docentes caracterizar las necesidades y estrategias de formación del rol de arquitecto de software en programas de pregrado.

El primer paso para encontrar una solución efectiva sobre cómo enseñar AS, es obtener el listado de competencias mínimas a desarrollar desde el pregrado. Para obtener ese listado hemos seguido una serie de pasos de manera secuencial y sistemática involucrando a los docentes e ingenieros de la industria. Hemos logrado clasificar las competencias de AS en las categorías: obligatorias, opcionales y fuera de alcance para un curso de pregrado.

Además, hemos identificado dos grupos claramente diferenciados de competencias. El primero, está relacionado con el desarrollo de un sistema nuevo, y el segundo, con el mantenimiento de un sistema software. Claramente las competencias del segundo grupo son mucho más complejas de desarrollar desde la academia. A los estudiantes les dificulta más entender y modificar la arquitectura de un sistema existente que proponer uno nuevo desde cero. Sin embargo, este segundo grupo de competencias, es el más demandado por la industria.

Las competencias obligatorias encontradas en esta investigación, según la opinión de algunos docentes, son difíciles de tratar en su totalidad en programas de pregrado. Esto indica que en la academia no se está cubriendo las competencias mínimas que la industria demanda de un recién egresado.

Estas competencias encontradas serán la base de futuros proyectos que permitan encontrar estrategias de formación que permitan diseñar cursos de AS acorde a las necesidades de la industria de software.

RECONOCIMIENTO

Damos un reconocimiento especial a todas los docentes e ingenieros que participaron de las encuestas, focus group y talleres durante esta investigación.

REFERENCIAS

[1] S. Angelov and P. de Beer, “Designing and Applying an Approach to Software Architecting in Agile Projects in Education,” *Journal of Systems and Software*, vol. 127, no. C, pp. 78–90, 2017, doi: 10.1016/j.jss.2017.01.029.

[2] M. Galster and S. Angelov, “What makes teaching software architecture difficult?,” in *Proceedings - International Conference on Software Engineering*, Austin Texas: Association for Computing MachineryNew YorkNYUnited States, 2016, pp. 356–359. doi: 10.1145/2889160.2889187.

[3] IEEE, “IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems,” 2000. doi: 10.1109/IEEESTD.2000.91944.

[4] M. Richards and N. Ford, *Fundamentals of Software Architecture: An Engineering Approach 1st Edicion*. Canada: O’Reilly Media, Inc., 2020. [Online]. Available: https://www.amazon.com/Fundamentals-Software-Architecture-Comprehensive-Characteristics/dp/1492043451

[5] M. A. Shah, I. Ahmed, and M. Shafi, “Role of Software Architect: A Pakistani Software

Industry Perspective," *Res J Recent Sci*, vol. 3, pp. 48–52, 2014.

[6] O. E. Lieh and Y. Irawan, "Teaching adult learners on software architecture design skills," in *Proceedings - Frontiers in Education Conference, FIE*, Uppsala, Sweden: IEEE, 2019, pp. 1–9. doi: 10.1109/FIE.2018.8658714.

[7] A. Jansen and J. Bosch, "Software Architecture as a Set of Architectural Design Decisions," in *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, Pittsburgh, PA, USA: IEEE, 2005, pp. 109–120. doi: 10.1109/WICSA.2005.61.

[8] M. Fowler, "Who Needs an Architect?," *IEEE Softw*, vol. 20, no. 5, pp. 11–13, 2003, doi: 10.1109/MS.2003.1231144.

[9] R. C. De Boer and H. Van Vliet, "On the similarity between requirements and architecture," *Journal of Systems and Software*, vol. 82, no. 3, pp. 544–550, 2009, doi: https://doi.org/10.1016/j.jss.2008.11.185.

[10] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice, third Edition*. Massachusetts, USA: Pearson Education, 2012. [Online]. Available: https://www.amazon.com/Software-Architecture-Practice-3rd-Engineering/dp/0321815734

[11] A. E. Sabry, "Decision model for software architectural tactics selection based on quality attributes requirements," *Procedia Comput Sci*, vol. 65, pp. 422–431, 2015.

[12] L. Dobrica and E. Niemela, "A survey on software architecture analysis methods," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 638–653, 2002, doi: 10.1109/TSE.2002.1019479.

[13] N. B. Harrison and P. Avgeriou, "How do architecture patterns and tactics interact? A model and annotation," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1735–1758, 2010.

[14] R. S. Pillutla and A. Alladi, "Methodology to bridge the gaps between engineering education and the industry requirements," in *Eurocon 2013*, Zagreb, Croatia: IEEE, 2013, pp. 926–932. doi: 10.1109/EUROCON.2013.6625093.

[15] P. M. Leidig and L. Cassel, "ACM Taskforce efforts on computing competencies for undergraduate data science curricula," in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 2020, pp. 519–520.

[16] E. Moreno Vélez, "Arquisoft90 Formación profesional y capacitación," 2020. Accessed: May 31, 2020. [Online]. Available: https://www.linkedin.com/showcase/arquisoft90-entrenamiento-den-arquitectura-de-software

[17] A. Van Deursen *et al.*, "A Collaborative approach to teaching software architecture," in *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*, in SIGCSE '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 591–596. doi: 10.1145/3017680.3017737.

[18] V. Garousi, G. Giray, E. Tüzün, C. Catal, and M. Felderer, "Closing the gap between software engineering education and industrial needs," *IEEE Softw*, vol. 37, pp. 68–77, 2020, doi: 10.1109/MS.2018.2880823.

[19] E. Lieh Ouh, B. Kok Siew Gan, and Y. Irawan, "Did our Course Design on Software Architecture meet our Student's Learning Expectations?," in *2020 IEEE Frontiers in Education Conference (FIE)*, Uppsala, Sweden: IEEE, 2020, pp. 1–9. doi: 10.1109/FIE44824.2020.9274014.

[20] L. M. Barbosa Guerrero, "Arquitectura De Software Como Eje Temático De Investigación," *Ingeniería*, pp. 78–85, 2006, doi: 10.1109/MC.2015.268.

[21] T. Akhriza, Y. ma, and J. Li, "Revealing the Gap Between Skills of Students and the Evolving Skills Required by the Industry of Information and Communication Technology," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, pp. 675–698, 2017, doi: 10.1142/S0218194017500255.

[22] M. A. Unigarro Gutiérrez, "Un modelo educativo crítico con enfoque de competencias," 2017. [Online]. Available: https://revistas.ucc.edu.co/index.php/dotr/article/view/1833/1921

[23] A. Van Deursen *et al.*, "A Collaborative approach to teaching software architecture," in *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*, Seattle Washington USA: ACM, 2017, pp. 591–596. doi: 10.1145/3017680.3017737.

[24] A. Lopes, I. Steinmacher, and T. Conte, "UML Acceptance: Analyzing the

Students' Perception of UML Diagrams," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, in SBES 2019. New York, NY, USA: Association for Computing Machinery, 2019, pp. 264–272. doi: 10.1145/3350768.3352575.

[25] D. C. Schmidt and Z. McCormick, "Producing and delivering a MOOC on pattern-oriented software architecture for concurrent and networked software," in *SPLASH 2013 - Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, and Applications: Software for Humanity*, Indianapolis Indiana USA: ACM, 2013, pp. 167–176. doi: 10.1145/2508075.2508465.

[26] P. Ciancarini, S. Russo, and V. Sabbatino, "A Course on Software Architecture for Defense Applications," in *Proceedings of 4th International Conference in Software Engineering for Defence Applications*, P. Ciancarini, A. Sillitti, G. Succi, and A. Messina, Eds., Cham: Springer International Publishing, 2016, pp. 321–330.

[27] M. Palacin-Silva, J. Khakurel, A. Happonen, T. Hynninen, and J. Porras, "Infusing Design Thinking into a Software Engineering Capstone Course," in *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T)*, Savannah, Georgia, USA: IEEE, 2017, pp. 212–221. doi: 10.1109/CSEET.2017.41.

[28] B. Wei, Y. Li, L. Deng, and N. Visalli, "Teaching Distributed Software Architecture by Building an Industrial Level E-Commerce Application," in *Studies in Computational Intelligence*, vol. 845, Cham: Springer International Publishing, 2020, pp. 43–54. doi: 10.1007/978-3-030-24344-9_3.

[29] C. R. Rupakheti and S. V Chenoweth, "Teaching Software Architecture to Undergraduate Students: An Experience Report," in *Proceedings - International Conference on Software Engineering*, Florence Italy: IEEE Press, 2015, pp. 445–454. doi: 10.1109/ICSE.2015.177.

[30] J. Joy and V. G. Renumol, "Activity oriented teaching strategy for software engineering course: An experience report," *Journal of Information Technology Education: Innovations in Practice*, vol. 17, pp. 181–200, 2018, doi: 10.28945/4116.

[31] Z. Li, "Using Public and Free Platform-as-a-Service (PaaS) based Lightweight Projects for Software Architecture Education," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, Seoul South Korea: Association for Computing Machinery, 2020, pp. 1–11. doi: 10.1145/3377814.3381704.

[32] L. Zhang, Y. Li, and N. Ge, "Exploration on theoretical and practical projects of software architecture course," in *15th International Conference on Computer Science and Education, ICCSE 2020*, Delft, Netherlands: IEEE, 2020, pp. 391–395. doi: 10.1109/ICCSE49874.2020.9201748.

[33] O. E. Lieh and Y. Irawan, "Exploring Experiential Learning Model and Risk Management Process for an Undergraduate Software Architecture Course," in *2018 IEEE Frontiers in Education Conference (FIE)*, San Jose, CA, USA: IEEE, 2018, pp. 1–9. doi: 10.1109/FIE.2018.8659200.

[34] F. G. Silva, P. E. D. Dos Santos, and C. von Flach G. Chavez, "FLOSS in Software Engineering Education: Supporting the Instructor in the Quest for Providing Real Experience for Students," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, Salvador Brazil: ACM, 2019, pp. 234–243. doi: 10.1145/3422392.3422493.

[35] K. May, B. Yang, J. Zhou, Y. Lin, K. Zhang, and Z. Yu, "Outcome-based school-enterprise cooperative software engineering training," in *ACM International Conference Proceeding Series*, Shanghai China: Association for Computing MachineryNew YorkNYUnited States, 2018, pp. 15–20. doi: 10.1145/3210713.3210722.

[36] S. Mohan, S. Chenoweth, and S. Bohner, "Towards a Better Capstone Experience," in *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, in SIGCSE '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 111–116. doi: 10.1145/2157136.2157173.

[37] Z. S. H. Abad, M. Bano, and D. Zowghi, "How Much Authenticity Can Be Achieved in Software Engineering Project Based Courses?," in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training*, in ICSE-SEET

’19. Montreal Quebec Canada: IEEE Press, 2019, pp. 208–219. doi: 10.1109/ICSE-SEET.2019.00030.

[38] M. Niño and R. Anaya, “Hacia un enfoque basado en competencias para la enseñanza de la ingeniería de software utilizando investigación-acción,” in *Encuentro Internacional de Educación en Ingeniería ACOFI*, Cartagena de Indias, 21017. [Online]. Available: https://acofipapers.org/index.php/eiei/article/view/586

[39] A. W. Kiwelekar and H. S. Wankhede, “Learning objectives for a course on software architecture,” in *European Conference on Software Architecture*, 2015, pp. 169–180.

[40] S. M. Putman and T. Rock, *Action Research: Using Strategic Inquiry to Improve Teaching and Learning*. SAGE Publications, 2016. [Online]. Available: https://books.google.com.co/books?id=AX1ZDwAAQBAJ

ANEXOS

A. Competencias de los arquitectos de software según la revisión de la literatura

Tabla 6: *Competencias de los arquitectos de software según la revisión de la literatura*

Id	Descripción
	Creación de una Arquitectura
C01	Identifica claramente los atributos de calidad relevantes del software que conducirán la arquitectura de un sistema de software a construir.
C02	Diseña consistentemente la AS definiendo cómo los componentes interactúan entre sí.
C03	Toma decisiones de diseño relevantes sobre cómo debe construirse un sistema involucrando las elecciones que enfrenta un arquitecto al diseñar un sistema de software.
C04	Expande cuidadosamente los detalles del diseño perfeccionándolo para converger en el diseño final.
	Análisis y Evaluación De Una Arquitectura
C05	Evalúa independientemente una AS para determinar la satisfacción de los requisitos funcionales y no funcionales.
C06	Revisa frecuentemente los diseños de los componentes propuestos por ingenieros junior verificando el cumplimiento de la arquitectura.
C07	Aplica sistemáticamente técnicas de arquitectura basadas en valor para evaluar decisiones arquitectónicas.
C08	Realiza imparcialmente un análisis de compensación (trade-off) para evaluar arquitecturas.
	Documentación de Arquitectura
C09	Prepara organizadamente los documentos arquitectónicos y presentaciones útiles para las partes interesadas (stakeholders).
C10	Produce estándares de documentación que incluyan la variabilidad y el comportamiento dinámico.
	Trabajando con sistemas existentes
C11	Mantiene fácilmente los sistemas existentes y su arquitectura para lograr la evolución de los sistemas software.
C12	Rediseña las arquitecturas existentes para la migración a nuevas tecnologías y plataformas.
	Otras competencias
C13	Proporciona pro-activamente pautas arquitectónicas para las actividades de diseño de software.
C14	Lidera con entusiasmo actividades de mejora de la arquitectura en una organización desarrolladora de software.
C15	Participa activamente en la definición y mejora de procesos de software en una organización.
C16	Define reflexivamente la filosofía y los principios para la arquitectura global.
C17	Proporciona colaborativamente apoyo a la supervisión de la arquitectura de los proyectos de desarrollo de software.
	Gestión de requisitos
C18	Analiza críticamente los requisitos de software funcionales y de atributos de calidad.
C19	Comprende rápidamente las necesidades del negocio y del cliente para asegurar que los requisitos satisfagan estas necesidades.
C20	Captura sistemáticamente los requisitos del cliente, de la organización y del negocio en la arquitectura.
C21	Crea especificaciones de software claras a partir de los requisitos de negocio.

Fuente: *elaboración propia*

Tabla 6: *Competencias de los arquitectos de software según la revisión de la literatura*

Id	Descripción
	Implementación del producto
C22	Realiza periódicamente revisiones del código fuente escrito por el equipo de desarrollo.
C23	Desarrolla componentes de software reutilizables.
C24	Desarrolla soluciones basado en componentes reutilizables existentes.
C25	Asegura el cumplimiento de los lineamientos de codificación por parte del equipo de desarrollo.
C26	Recomienda metodologías de desarrollo para el equipo de desarrollo.
C27	Supervisa el trabajo de consultores y proveedores externos.
	Testing del producto
C28	Establece procedimientos de prueba considerando aspectos de la arquitectura (tipos de componentes/servicios, integración).
C29	Construye el producto facilitando la identificación y corrección de fallos.
	Evaluación de tecnologías futuras
C30	Evalúa explícitamente soluciones empresariales de software y hace recomendaciones.
C31	Gestiona cuidadosamente la introducción de nuevas soluciones de software en una organización.
C32	Analiza objetivamente el entorno de TI actual y recomienda soluciones para las deficiencias encontradas.
C33	Desarrolla documentos técnicos de calidad y los presenta a los interesados de la organización.
	Selección de herramientas y tecnología

C34	Realiza estudios confiables de viabilidad técnica de nuevas tecnologías y arquitecturas para la organización.
C35	Evalúa objetivamente herramientas comerciales y componentes de software desde una perspectiva arquitectónica.

Fuente: *elaboración propia*

B. Perfil de los ingenieros que participaron en el workshop

Tabla 7: *Perfil de los ingenieros que participaron en el workshop.*

No	Empresa	Años Exp.	Cargo	Funciones del cargo
1	Inxeption	5	Full-stack Developer	Desarrollo y evolución de aplicaciones web.
2	EPAM	+10	Senior Software Developer	Aplicar buenas prácticas de desarrollo (principios SOLID, Clean Code, Clean Design, etc) al código fuente nuevo y/o existente. Establecer estrategias de comunicación entre diferentes microservicios y poner en práctica estrategias de DevOps
3	Amazon	+10	Software Development Engineer.	Diseño y revisión de microservicios en un equipo de desarrollo de software
4	Universidad Autónoma de Occidente - Cali	+10	Arquitecto de Software	Diseño de soluciones de software empresarial, administración de servicios On-premise y Cloud. Evaluación de proveedores de tecnología y control del proceso de tercerización de desarrollo de software

Fuente: *elaboración propia*