ISSN: 1692-7257 - Volume 1 – Number 43 - 2024

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

*Digital Object Identifier: 10.24054/rcta.v1i43.2798*

# Insights into software architecture competencies integrating industry and academic perspectives

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

**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*

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

***Received:*** *November 8, 2023.* ***Accepted:*** *January 10, 2024.* ***Published:*** *February 25, 2024.*

***How to Cite:*** *W. L. Pantoja Yépez, A. F. Solano Alegría, A. Bandi, and J. A. Hurtado Alegría, "Insights into software architecture competencies integrating industry and academic perspectives", RCTA, vol. 1, no. 43, pp. 9–23, Feb. 2024. Recovered from https://ojs.unipamplona.edu.co/index.php/rcta/article/view/2798*

This work is licensed under a
Creative Commons Attribution-NonCommercial 4.0 International License.

**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.

**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.

## 1. INTRODUCTION

Software Architecture (SA) is a fundamental area of software engineering that ensures the quality of software products; hence, academia and industry have focused their expectations on designing a good curriculum [1]. However, teaching software architecture is still a challenging task; the teacher needs to address problems with a complexity similar to the real world, teamwork, and provide a special accompaniment, among other challenges [2].

The role of the architect is very challenging in any software project. An architect could be a person, team, or organization that designs the system's architecture (norma IEEE 1471-2000 [3]). The primary motivation of a software architect is to develop an architecture for the system. However, in addition, a software architect plays every other important role in the lifecycle of a software project. A software architect understands the development process, has knowledge of the business domain, and, in addition, has analysis and programming skills. An architect is a good communicator, knows the organization's policies, and plays an essential role in decision-making during the project. An architect is a catalyst who improves communication and develops understanding between clients and developers [4] [5]. An architect is considered to be a technical leader of the software project since, in all technical decisions, the architect's participation as a mediator of interests is essential [6].

The competencies students must achieve in the SA field are extensive and complex. The first step for an SA curriculum design that reduces the gap between what is taught in the classroom and what the industry demands is to know precisely the minimum competencies the software industry expects from recent graduates. The present article carries out a methodology composed of a series of steps to obtain this list of competencies. This section presents the introduction. Section 2 offers the problem. Section 3 describes the related work, while section 4 shows the design of the experience. Section 5 presents the analysis of the results. Finally, section 6 includes a set of conclusions and future work.

## 2. FUNDAMENTAL CONCEPTS OF SOFTWARE ARCHITECTURE TRAINING

This section includes the fundamental concepts around SA training.

### 2.1. Software Architecture

The concept of software architecture is continuously evolving, and it is essential to understand the various definitions that appear in the literature. A modern definition might involve a set of fundamental decisions regarding the structure of the software system, which guides the design and construction of the system [7] [8] [9]. This structure includes the organization of the components, the way they communicate, and the distribution of responsibilities among them [10]. Software architecture also addresses issues related to the quality of non-functional attributes, such as system performance, scalability, security, usability, and maintainability [4].

Some of the frequently mentioned characteristics of SA are [10]: (i) It is a primary system abstraction that stakeholders use to think, design, code, and communicate in terms of large conceptual blocks., (ii) It promotes high-level reuse and component reuse, (iii) It influences development productivity by reusing large frameworks to support the construction of product lines, (iv) It ensures quality throughout the software life cycle by explicitly addressing quality attributes such as modifiability, portability, scalability, and security.

### 2.2. Quality Attributes

Quality attributes refer to specific traits that a software product satisfies, and each attribute is associated with specific metrics that define the quality levels of a software product [11]. Quality attributes refer to specific traits that a software product satisfies, and each attribute is associated with specific metrics that define the quality levels of a software product [12]. Proper management of these quality attributes is crucial, as inappropriate management poses significant business risk. The architect must consider potential conflicts between quality attributes and resolve them through trade-offs.

### 2.3. Architectural Patterns

Architecture patterns refer to common solution structures to similar design problems. Each pattern describes a general software system structure or high-level behavior that must satisfy a product's functionalities, qualities, and constraints. These patterns are chosen based on early design decisions, such as satisfying functional requirements, non-functional requirements, and system constraints [13].

### 2.4. Competencies, skills and knowledge

Competence is defined as the ability to do something [14]. Knowledge can be understood as theoretical or practical understanding. For an individual, competence is composed of knowledge and skills.

According to Bass et al., in their book Software Architecture in Practice [10], they defines duties, skills, and knowledge. Duties, skills, and knowledge form a triad on which the architectural competencies of engineers are based. The skills and knowledge support the execution of the competencies (functions or duties) as shown in Fig. 1. The following is an example of these three concepts:

- "Designing an architecture" is a duty.
- "Thinking abstractly" is a skill.
- "Patterns and tactics" are part of a body of knowledge.

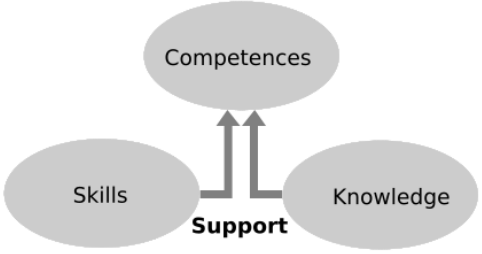


***Fig. 1.*** *Skills and knowledge support the execution of competencies.*
***Source*****:** [10].

## 3. THE PROBLEM

Traditionally, undergraduate engineering programs, such as Computer Science and Engineering, Information Technology, Software Engineering, and Systems Engineering, include many courses in which skills are developed and technical knowledge related to software construction is imparted through programming languages and development platforms [15]. However, students in these programs lack knowledge about SA and design problems [16] despite their growing importance for the software industry. Recent graduates lack sufficient skills in making appropriate design decisions, applying practices and knowledge related to software design in the business context [17].

Existing literature on SA education and practice point to different reasons for this lack of skills related to software architecture and design [18]. First, there is a considerable gap between the academic perception of SA and its practice in industry [19]. Sometimes, the problems that the industry considers most critical and challenging are not given due importance for research or training in second, there is a gap between the skills that the industry expects from graduates in Software Engineering and the skills taught in the curricula [20]. Third, many institutions do not have a clear vision of the topics in which software architects should be trained [21].

Designing a Software Architecture course curriculum requires aligning the course objectives with the expectations of the Software Industry. It is essential to define the competencies expected to be developed [16].

A competency is the sum of knowledge and skills; however, a competency is more than this; it implies the ability to satisfy complex demands by mobilizing and resorting to skills and attitudes in a particular context [22]. We found various competencies, skills, and knowledge that students should develop in the SA area. For example:

- The primary skill of the architect is to design, model, analyze, and evaluate software architectures [23] [24]. The architect must know how to apply patterns and frameworks to create quality applications [25]. The architecture of large-scale complex software systems, which have many requirements and millions of lines of code, requires very high modeling and abstraction skills [26].
- Knowledge of multiple technologies enables the software architect to choose the appropriate technologies for the project. Although software architects do not need to be technology experts, the architect must stay current on technology trends [27] [28].
- Understand the domain in which a system will live; this involves understanding the business, social, and operational environment in which a system must operate [29] .
- Analytical skills are essential for the software architect to quickly understand the problem, diagnose possible root causes, and make critical decisions for the Project [19] [30]. In addition, architects require the ability to find the root causes of high-level problems in existing designs, such as why a system runs too slowly or is not secure [28].
- Research skills are required to understand complex situations and solve problems [31] [32].
- Although architects should not be programming experts, they should have minimal programming skills to communicate with developers [33] [31].
- Architectural decision-making is another fundamental skill. An architect must learn to make design decisions in environments where much is unknown, where there is insufficient time to explore all alternatives, and where there is pressure to perform [19]. In addition, the architect must make decisions collaboratively.
- It requires systemic and holistic thinking and they consider problems from different perspectives [28].
- In addition to the above, communication skills such as speaking, writing, and presentations are required to address complex problems with a seemingly simple design that is easy [29] [34] for software architects to supervise and work closely with other members of the development team (Teamwork), such as programmers [35] [36]. Finally, negotiation and leadership are essential for an architect to lead, present, negotiate, and justify their designs and architectural decisions [35] [6].

Developing the above skills will likely require a great deal of time and experience. Therefore, the objectives of an undergraduate SA course must recognize the limitations of the target audience and work with the resources available to the teacher. The first step towards an SA curriculum design that decreases the gap between what is taught in the classroom and what the industry demands is to know precisely the minimum competencies the software

industry expects from recent graduates. Therefore, the research questions we pose are:

- What are the minimum competencies at the SA level that the industry expects from a recent graduate?
- How is academia addressing these industry-defined competencies?

## 4. RELATED WORKS

Niño & Anaya proposes a curricular reform of the software engineering area in systems engineering programs [38]. The main product is the definition of a map of professional competencies in software engineering, structured in first and second-level competencies. It also identifies the core subjects that contribute to developing the identified competencies.

Garousi et al. [18] conducted a literature review of studies that address the difficulty software engineering graduates have in starting their careers due to the misalignment of the skills learned in their undergraduate training with what the industry needs. This study allows educators and hiring managers to tailor their education/hiring efforts to prepare the software engineering workforce better.

Rupakheti & Chenoweth [29] describe the ten-year history of teaching an undergraduate SA course in an undergraduate software engineering degree program. Included are descriptions of what they perceive to be the realistic goals of teaching SA at this level. They describe that the primary goal of the course is to prepare students for high-level design situations in the industry by employing new and pervasive technologies and processes in their projects and correcting architectural problems in existing systems.

Kiwelekar & Wankhede present a set of learning objectives and their classification using the Revised Bloom's Taxonomy (RBT) [39]. The analysis highlights the generic cognitive skills required for architecture modeling. One of the potential benefits of classifying learning objectives is that different educational processes, such as instruction, learning, and assessment, can be effectively aligned using the classification of learning objectives presented in this study.

Paulisch et al. [38] studied the detailed role description of an architect, and this included which competencies they need to be achieved.

Taking into account the related studies, we identified that there is no list of minimum competencies in SA topics aligned with the needs of the software industry, which would subsequently allow teachers to characterize the needs and training strategies for the role of software architects in undergraduate programs.

## 5. METHODOLOGY

In this research, we used the action research method proposed by Putman & Rock [40], which iteratively performs the stages of planning, acting, and reflecting.

To know the minimum competencies in the area of SA that the industry expects in recent graduates of systems engineering and related programs, we carried out a first cycle of action research following the following steps:

- Identify the problem and define the research questions guiding the action research process.
- Analyze theory to develop an in-depth and synthetic understanding of the research topic.
- Create a research plan.
- Execute the plan and analyze the data collected.
- Reflect on the results of the action research.
- Develop a next cycle based on the research data.

Below, we explain each of the previous steps in detail.

### 5.1. Step 1 - Identify the problem and define the research questions guiding the action research process

The problem to be solved and the research questions are described in Section 3 and framed in the difficulties in training undergraduate SA competencies aligned with the expectations of the software industry.

### 5.2. Step 2 - Analyze theory to develop an in-depth and synthetic understanding of the research topic

We conducted a literature review looking for experiences in software architecture courses, and from there, we were able to create a solid conceptual basis, defining key concepts such as competence, skill, and knowledge. In addition, we could compile the competencies sought to be developed in the courses.

### 5.3. Step 3 - Create a research plan

According to the research questions established in this cycle, which seek to know the competencies in the SA area that the industry expects from recent graduates, we selected three appropriate data collection methods: survey, workshop, and focus group. Table 1, shows the specific instruments we used to collect data. Each instrument allows us to find and refine the list of expected competencies, as shown in Fig. 2.

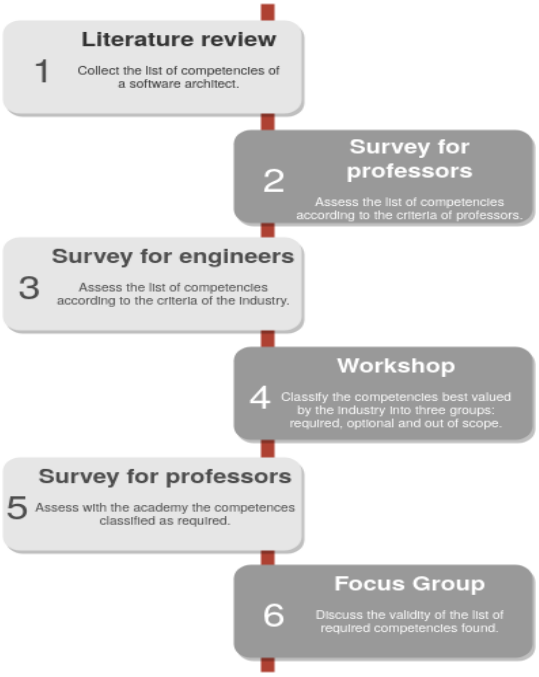


***Fig. 2.*** *Data collection instruments and their purpose.*
***Source:*** *Elaborated by the authors.*

Finally, we chose the appropriate statistical analysis techniques and tools for the investigation:

- Google Forms to capture survey data.
- Jamboard for a workshop with industry engineers to classify competencies.
- Studio R for processing data and generating statistical graphs.

***Table 1****: Data collection instruments and their purpose.*

| No | Instrument | Goal |
|---|---|---|
| 1 | Literature review. | Collect the list of competencies of a software architect using the literature review. |
| 2 | Professors' evaluation survey. | Evaluate the list of competencies according to the professors' criteria. |
| 3 | Engineers' evaluation survey. | Evaluate the list of competencies according to the engineers' criteria. |
| 4 | Workshop with industry engineers. | Classify the competencies best valued by industry into three groups: mandatory, optional, and out of scope of an undergraduate course. |
| 5 | Survey of mandatory competencies to professors. | Evaluate the competencies classified as mandatory with the academy. |
| 6 | Focus group with professors. | Discuss the validity of the list of mandatory competencies found. |

***Source:*** *Elaborated by the authors*

### 5.4. Step 4 - Execute the plan and analyze the data collected

After planning the activities to be carried out in the action research cycle, we executed the actions. Next, we explain each of the instruments executed and the results obtained.

#### *5.4.1. Literature review*

After a review of the literature, we obtained a comprehensive list of 35 competencies of software architects (see Table 6 in the Annexes). These competencies are classified into the following categories: Architecture Creation, Architecture Analysis and Evaluation, Architecture Documentation, Working with Existing Systems, Other Competencies, Requirements Management, Product Implementation, Product Testing, and Selection of Tools and Technology. In the following steps, we seek to analyze this list and filter the competencies that the industry expects in recent graduates. In subsequent sections, we will continue to use the same competency identifiers given in Table 6.

#### *5.4.2. Professors' evaluation survey*

The list of competencies in Table 6 was submitted for evaluation by a group of nine teachers in the area related to Software Engineering. The group of professors was asked to evaluate the Software Architecture competencies acquired by their students during the undergraduate training process in each of their universities. For each competency, they were asked to evaluate the level of importance (or dedication) given to it in the Program, following the following scale: (1) Not important, (2) Not very important, (3) Neutral, (4) Important, (5) Very important. The results of this experience can be seen in Fig. 3.

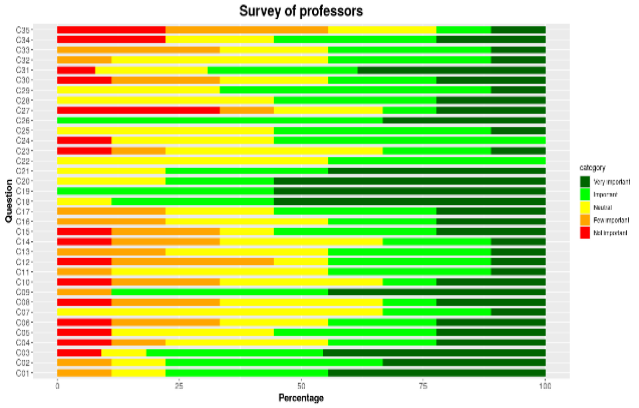


***Fig. 3.*** *Evaluation of the 35 Software Architecture competencies by university professors.*
***Fuente:*** *Elaborated by the authors.*

Subsequently, we passed the competency ratings from Fig. 3 using the following formula:

$$Score = VotesNotImportant \times 0 + VotesFewImportant \times 1 + VotesNeutral \times 2 + VotosImportant \times 3 + VotosVeryImportante \times 4$$

Thus, the ten most essential competencies, as rated by professors, are shown in Table 2 below.

***Table 2:*** *The ten most essential competencies, as rated by professors*

| 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 |

***Source:*** *Elaborated by the authors*

### *5.4.3. Engineers' evaluation survey*

The list of competencies in Table 6 was submitted for assessment by a group of 21 engineers from the industry who are working as software architects or related tasks. The group of engineers was asked to evaluate the Software Architecture competencies they expect from a recent graduate of a systems engineering program or related career. For each competency, they were asked to rate the level of importance for the industry according to the following scale: (1) Not important, (2) Not very important, (3) Neutral, (4) Important, (5) Very important. The results of this experience can be seen in Fig. 4.

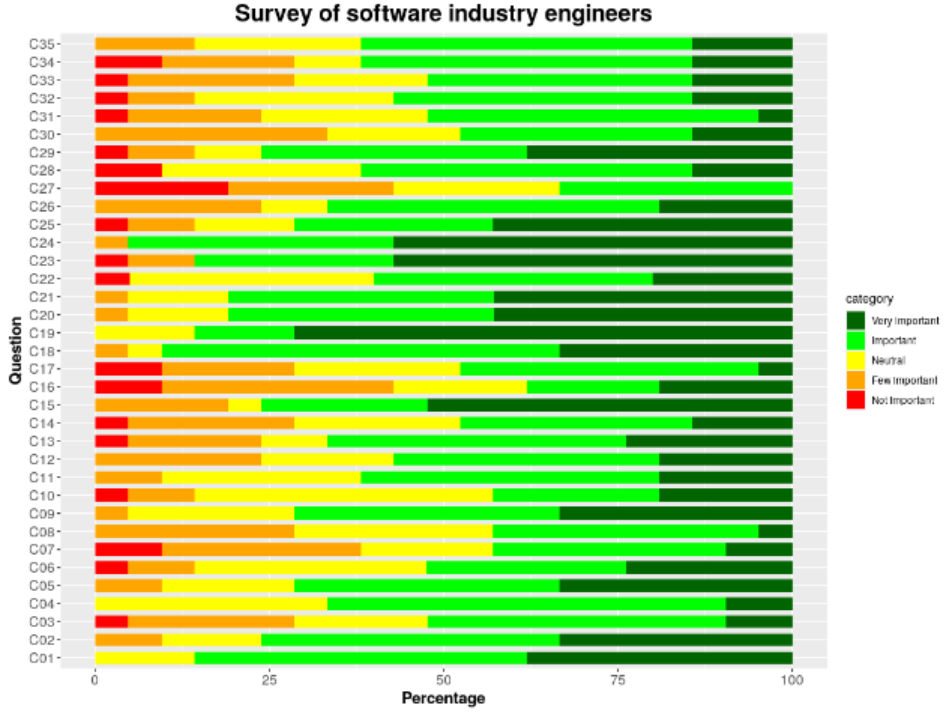


***Fig. 4.*** *Evaluation of the 35 Software Architecture competencies by software industry engineers.*
***Source:*** *Elaborated by the authors.*

Subsequently, we passed the competency ratings from Fig. 4, using the following formula:

$$Score = VotesNotImportant \times 0 + VotesFewImportant \times 1 + VotesNeutral \times 2 + VotosImportant \times 3 + VotosVeryImportante \times 4$$

Thus, the ten most essential competencies according to the assessment of the engineers can be seen in Table 3. In addition, we place in bold the competencies that coincide with the evaluation of the professors: C19, C18, C20, C21, and C02. Fig. 5 shows graphically the two sets of competencies and their overlaps.

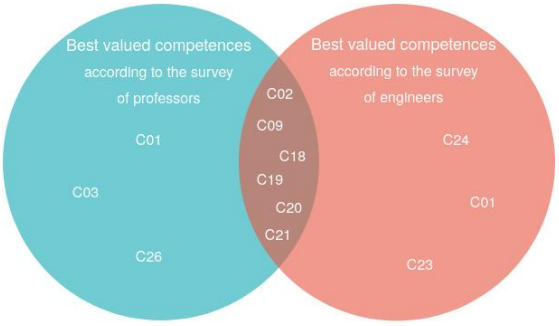


***Fig. 5.*** *Top-rated competencies according to academia and industry.*
***Source:*** *Elaborated by the authors.*

Table 7 (in the Annexes section).
The steps involved in this activity were as follows:

- Classify the competencies of the given stickers into three groups: mandatory, optional, and out of scope for a recent graduate. Each of the five members can decide to place the sticker in the corresponding column. In case of doubt, they can rely on the opinions of their colleagues. They can also depend on the results of the survey—maximum time: 10 minutes.
- Review as a group the classification made and make any necessary adjustments. This review is a space to refine the group work—maximum time: 15 minutes.
- Socialize the results. It is a brief justification of the classification made to the researcher. One member can do it with the support of the others—maximum time: 5 minutes.

The final result of this activity can be seen in Fig.

***Table 3:*** *The ten most valued competencies according to the survey of engineers (In bold are the coincidences between industry and 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:*** *Elaborated by the authors*

### *5.4.4. Workshop with industry engineers*

The next step was to classify the competencies best valued by industry into three groups: mandatory, optional, and out of scope for an undergraduate course. The list of competencies in Table 6 is too broad to be addressed in an undergraduate university course. Therefore, this activity seeks to classify and reduce that list by looking for the most relevant competencies for a training course.

This collaborative activity was conducted virtually and synchronously using Google Meet and a Jamboard with "stickers" for each competency. In addition, the activity counted four systems engineers with extensive experience in tasks related to SA. The profile of the engineers can be seen in

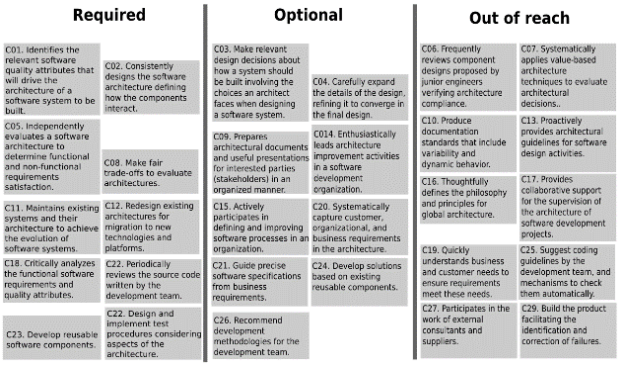


***Fig. 6.*** *Final result of the workshop with the classification of the competencies.*
***Source:*** *Elaborated by the authors.*

### *5.4.5. Survey of mandatory competencies to professors*

The purpose of this last survey was to find out how the competencies classified as mandatory as a result of the collaborative activity are being addressed in the universities. The steps carried out for this survey were as follows.

- Search the internet for regional, national, and international universities with Systems Engineering and related programs.
- For each university, search the web for the profiles of professors with a Software Architecture teaching profile.
- Write e-mails to the selected professors inviting them to participate in the survey.

The survey was completed by 18 universities, with one professor from each institution (see Table 4).

***Table 4:*** *Professors who participated in the mandatory competencies survey*

| 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 |

***Source:*** *Elaborated by the authors*

The question asked in the survey was: "*Regarding the mandatory competencies, we would like to know how they are being addressed in your Institution by answering according to the Likert scale: (1) Not addressed, (2) Little addressed, (3) Neutral, (4) Very much addressed, (5) Totally addressed*". The responses to this survey can be seen in Fig. **6** where it can be seen that all competencies have some degree of "Not addressed" and "Little addressed."

Calculating totals, Table 5 shows the percentages of the competencies in their five categories. It can be seen that 16.1% of the universities do not address the mandatory competencies, and 11.7% do not address them very little.

***Table 5:*** *Professors who participated in the mandatory competencies survey*

| Category | Score |
|---|---|
| Not addressed | 16.1% |
| Little addressed | 11.7% |
| Neutral | 22.8% |
| Very addressed | 22.8% |
| Fully addressed | 26.7% |

***Source:*** *Elaborated by the authors*

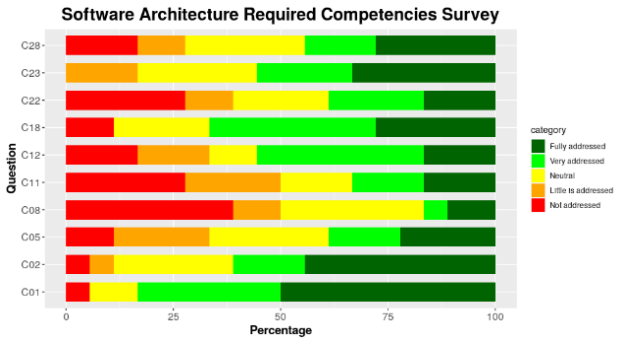


***Fig. 6.*** *Evaluation of the 10 Software Architecture competencies by several universities.*
***Fuente:*** *Elaborated by the authors.*

### *5.4.6. Focus group with professors*

The objective of this focus group was to share experiences among a group of professors from several universities in the way undergraduate SA is taught. A list of professors at regional, national, and international levels was organized, and the invitation to the event was sent by e-mail with an attached document containing details of the meeting (introduction, roles, agenda, and questions to be addressed). Although five professors initially accepted the invitation, at the time of the synchronous meeting, only professors from three universities showed up: Universidad del Cauca, Institución Universitaria Colegio Mayor del Cauca, and Universidad Nacional de la Plata (Argentina).

The following questions were discussed during the focus group:

1. *Question 1:* How are you developing the SA competencies in your universities? (In which courses, which semesters, what topics are addressed, and what strategies are used to recreate a real environment, among others).
2. *Question 2:* What do you think of these results, do you agree, and what would you change about

the next classification? We surveyed a group of software industry professionals on what SA competencies they expect from a recent undergraduate graduate. The competencies were classified into three categories: a) mandatory, b) optional, and c) out of reach for a recent graduate. The results of this experiment can be seen in Fig. What do you think of these results, do you agree, and what would you change about this classification?

The following is a summary of the two questions' most essential points.

**Question 1:**

- "In our institution, the careers strongly emphasize software development, but their orientation is towards learning coding in various programming languages. The subject of SA does not have enough emphasis".
- "In our university, there are three Software Engineering courses in addition to the courses called Project 1 and 2. The subject of SA is studied with greater emphasis in the Software Engineering II course. However, the subject is so broad that it is not covered in its entirety".
- "In our Institution we have two careers related to Software Engineering and there are courses that are organized according to the software life cycle, that is, we have the subjects of analysis, design, implementation, verification and maintenance. However, Software Architecture is dealt with in a chapter in a month within the subject of Software Design".

**Question 2:**

- "I agree with all the cards that have been defined as mandatory, optional, and out of scope. Furthermore, competencies can be classified into two types. The first those related to the development of a new system, and the second, to the maintenance of a software system along with its architecture."
- "The competency related to doing source code reviews, in principle causes me some strangeness, but it could be related to reviewing that the coding is correct according to the proposed architecture."
- "I would say that developing all the mandatory skills, I would say, is somewhat costly for undergraduate students. For example, just to understand the value of doing a good requirements specification we have to expose our kids to complex domains where unfamiliar terms appear. If it were a simple domain students would not see the need to specify, but do the coding directly. Similarly with SA, we need to involve large projects to see the importance of the different architectural styles".
- "On occasion we tried to expose the students to modify an existing system (developed by the professors) for them to study and modify the architecture. This experience was very costly to develop in each semester to prevent students from overdoing the work. This meant that we are now working with new projects, but with changing requirements during the course of the semester."
- "In the corporate environment, it is becoming increasingly common for developers to migrate to another company that offers them a better salary. This results in having to find new developers and challenge them to modify existing systems."
- "I am of the opinion that competency 19: *Quickly understands business and customer needs to ensure that requirements meet these needs* should not be in the Out-of-Scope column but in Mandatory. This competency is important to quickly appropriate the business domain."
- "When comparing the competencies classified as mandatory by the industry sector, I feel that many of them our students fail to develop in the different subjects and projects that are developed in class. We are left with the concern about how to get students to get there."

### 5.5. Step 5 - Reflecting on the results of action research

At the end of the action research cycle, we have found answers to the two research questions. As a first finding, we have a set of minimum competencies (from the mandatory category) at the SA level that the industry expects from a recent graduate:

1. C01: Clearly identifies the relevant software quality attributes that will drive the architecture of a software system to be built.
2. C02. Consistently design the software architecture by defining how components interact with each other.
3. C05: Independently evaluates a software architecture to determine functional and non-functional requirements satisfaction.
4. C08. Impartially performs a trade-off analysis to evaluate architectures.

5. C11. Maintains existing systems and their architecture to achieve the evolution of software systems.
6. C12. Redesigns existing architectures for migration to recent technologies and platform.
7. C18. Critically analyzes functional and quality attribute software requirements.
8. C19. Understands business and customer needs quickly to ensure that requirements meet these needs.
9. C22. Periodically performs reviews of the source code written by the development team.
10. C23. Develops reusable software components.
11. C28. Designs and implements test procedures considering architectural aspects (component/service types, integration).

To the previous set of competencies, we have added competency C28 at the suggestion of the professors' focus group.

As a second finding, and in response to research question 2, it is generally difficult for the academy to cover the teaching of this set of competencies. It is evidenced by the professors' survey and the focus group responses. We were able to show with the survey that 16.1% of universities do not address mandatory competencies and 11.7% do not address them very much.

### 5.6. Step 6 - Develop a next cycle based on research data

This cycle of action research will be the main input to develop the second cycle of action research, which is related to the creation of training patterns that will guide the teacher in the design of SA courses for undergraduate programs.

## 6. CONCLUSIONS AND FURTHER WORK

An SA course according to the needs of the industry is essential in the study plans of systems engineering and related programs; however, the undergraduate students' training, with the skills that the industry demands, has many challenges. The discrepancy between what is taught in universities and what the software industry expects is a problem that is evident through this study. Aligning software architecture courses with industry requirements is crucial for computer science, systems engineering, and related programs curricula. However, imparting the skills demanded by the industry to university students poses numerous challenges. Knowing the skills required by the industry is the first step to creating courses that help the employability of recent graduates. The identification of which competencies can be incorporated with less effort and greater efficiency makes it possible to draw a route in which universities can begin that path towards meeting the expectations of the software industry.

Taking into account the related studies, we were able to identify that there is no list of minimum competencies in SA topics aligned with the needs of the software industry, which subsequently allow professors to characterize the needs and training strategies of the role of software architect software in undergraduate programs.

The first step to find an effective solution on how to teach SA is to obtain the list of minimum competencies to develop from the undergraduate level. To obtain this list we have followed a series of steps in a sequential and systematic manner involving teachers and engineers from the industry. We have managed to classify the SA competencies into the categories: mandatory, optional and out of scope for an undergraduate course. Taking into account the related studies, we were able to identify that there is no list of minimum competencies in SA topics aligned with the needs of the software industry, which subsequently allow teachers to characterize the needs and training strategies of the role of software architect. software in undergraduate programs.

Furthermore, we have identified two clearly differentiated groups of competencies. The first is related to the development of a new system, and the second, to the maintenance of a software system. Clearly the competencies of the second group are much more complex to develop from the academy. It is more difficult for students to understand and modify the architecture of an existing system than to propose a new one from scratch. However, this second group of skills is the most in demand by the industry.

The mandatory competencies found in this research, in the opinion of some professors, are difficult to address in their entirety in undergraduate programs. This indicates that the academy is not covering the minimum skills that the industry demands of a recent graduate.

These competencies found will be the basis of future projects that allow finding training strategies that allow designing SA courses according to the needs of the software industry.

## ACKNOWLEDGMENTS

We give special recognition to all the professors and engineers who participated in the surveys, focus groups, and workshops during this research.

## REFERENCES

[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

## ANNEXES

### A. Competencies of software architects according to the literature review

***Table 6:*** *Competencies of software architects according to the literature review.*

| Id | Description |
|---|---|
| | Creation of an Architecture |
| C01 | Clearly identifies the relevant software quality attributes that will drive the architecture of a software system to be constructed. |
| C02 | Consistently design the software architecture by defining how components interact with each other. |
| C03 | Makes relevant design decisions about how a system should be built involving the choices an architect faces when designing a software system. |
| C04 | It carefully expands the details of the design, refining it to converge in the final design. |
| | Analysis and Evaluation of an Architecture |
| C05 | Independently evaluates a software architecture to determine functional and non-functional requirements satisfaction. |
| C06 | Frequently reviews component designs proposed by junior engineers verifying compliance with the architecture. |
| C07 | Systematically applies value-based architectural techniques to evaluate architectural decisions. |
| C08 | Impartially performs a trade-off analysis to evaluate architectures. |
| | Architectural Documentation |
| C09 | Organized preparation of architectural documents and presentations useful for stakeholders. |
| C10 | Produces documentation standards that include variability and dynamic behavior. |
| | Trabajando con sistemas existentes |
| C11 | Easily maintains existing systems and their architecture to achieve the evolution of software systems |
| C12 | Redesigns existing architectures for migration to recent technologies and platforms. |
| | Other Competencies |
| C13 | Proactively provides architectural guidelines for software design activities. |
| C14 | Enthusiastically leads architecture improvement activities in a software development organization. |
| C15 | Actively participates in defining and improving software processes in an organization. |
| C16 | Reflectively defines the philosophy and principles for global architecture. |
| C17 | Collaboratively provides architecture oversight support for software development projects. |
| | Requirements Management |
| C18 | Critically analyzes functional and quality attribute software requirements. |
| C19 | Understands business and customer needs quickly to ensure that requirements meet these needs. |
| C20 | Systematically captures customer, organizational, and business requirements in the architecture. |
| C21 | Creates clear software specifications from business requirements. |

***Source:*** *Elaborated by the authors*

***Table 6:*** *Competencies of software architects according to the literature review.*

| Id | Description |
|---|---|
| | Product Implementation |
| C22 | Periodically reviews the source code written by the development team. |
| C23 | Develops reusable software components. |
| C24 | Develops solutions based on existing reusable components. |
| C25 | Ensures compliance with coding guidelines by the development team. |
| C26 | Recommends development methodologies for the development team. |
| C27 | Monitors the work of consultants and external suppliers. |
| | Product Testing |
| C28 | Establishes test procedures considering architectural aspects (types of components/services, integration). |
| C29 | Builds the product by facilitating the identification and correction of faults. |
| | Evaluation of Future Technologies |
| C30 | Explicitly evaluates enterprise software solutions and makes recommendations. |
| C31 | Carefully manages the introduction of new software solutions in an organization. |
| C32 | Objectively analyzes the current IT environment and recommends solutions for the deficiencies found. |
| C33 | Develops quality technical documents and presents them to organizational stakeholders. |
| | Selection of Tools and Technology |
| C34 | Performs reliable technical feasibility studies of recent technologies and architectures for the organization. |
| C35 | Objectively evaluates commercial tools and software components from an architectural perspective. |

***Source:*** *Elaborated by the authors*

### B. Profile of the engineers who participated in the workshop

***Table 7:*** *Profile of the engineers who participated in the workshop.*

| No | Company | Year Exp. | Job | Job functions |
|---|---|---|---|---|
| 1 | Inxeption | 5 | Full-stack Developer | Development and evolution of web applications. |
| 2 | EPAM | +10 | Senior Software Developer | Apply good development practices (SOLID principles, Clean Code, Clean Design, etc.) to new and/or existing source code. Establish communication strategies between different microservices and implement DevOps strategies. |
| 3 | Amazon | +10 | Software Development Engineer. | Microservices design and review in a software development team. |

| | | | | |
|---|---|---|---|---|
| 4 | Universidad Autónoma de Occidente - Cali | +10 | Software Arquitect | Design of enterprise software solutions, management of On-premises and On-premises and Cloud services. Evaluation of technology providers and control of the software development outsourcing process. |

***Source:*** *Elaborated by the authors*