2021

Self-Confidence of Undergraduate Students in Designing Software Architecture

Lotfi ben Othmane
Iowa State University, othmanel@iastate.edu

Ameerah-Muhsina Jamil
Iowa State University, amjamil@iastate.edu

The complete bibliographic information for this item can be found at https://lib.dr.iastate.edu/ece_pubs/284. For information on how to cite this item, please visit http://lib.dr.iastate.edu/howtocite.html.

Self-Confidence of Undergraduate Students in Designing Software Architecture

Abstract

Contributions: This paper investigates the relations between undergraduate software architecture students' self-confidence and their course expectations, cognitive levels, preferred learning methods, and critical thinking.

Background: these students, often, lack self-confidence in their ability to use their knowledge to design software architectures.

Intended Outcomes: Self-confidence is expected to be related to the students' course expectations, cognitive levels, preferred learning methods, and critical thinking.

Application Design: We developed a questionnaire with open-ended questions to assess the self-confidence levels and related factors, which was taken by one-hundred ten students in two semesters. The students answers were coded and analyzed afterward.

Findings: We found that self-confidence is weakly associated with the students' course expectations and critical thinking and independent from their cognitive levels and preferred learning methods. The results suggest that to improve the self-confidence of the students, the instructors should ensure that the students' have "correct" course expectations and work on improving the students' critical thinking capabilities.

Disciplines

Computer and Systems Architecture | Engineering Education

Comments

This is a pre-print of the following paper: ben Othmane, Lotfi and Ameerah-Muhsina Jamil, "Self-Confidence of Undergraduate Students in Designing Software Architecture." (2021).

Self-Confidence of Undergraduate Students in Designing Software Architecture

Lotfi ben Othmane * *IEEE Senior Member*, Ameerah-Muhsina Jamile*
* Iowa State University, Ames, IA USA

***Abstract*—Contributions: This paper investigates the relations between undergraduate software architecture students' self-confidence and their course expectations, cognitive levels, preferred learning methods, and critical thinking. Background: these students, often, lack self-confidence in their ability to use their knowledge to design software architectures. Intended Outcomes: Self-confidence is expected to be related to the students' course expectations, cognitive levels, preferred learning methods, and critical thinking. Application Design: We developed a questionnaire with open-ended questions to assess the self-confidence levels and related factors, which was taken by one-hundred ten students in two semesters. The students answers were coded and analyzed afterward. Findings: We found that self-confidence is weakly associated with the students' course expectations and critical thinking and independent from their cognitive levels and preferred learning methods. The results suggest that to improve the self-confidence of the students, the instructors should ensure that the students' have "correct" course expectations and work on improving the students' critical thinking capabilities.**

I. Introduction

Undergraduate students are expected to step directly into software developer positions and succeed. Typical undergraduate students are, however, not prepared for the ambiguity of the industry [1]. The lack of self-confidence makes them resistant to take opportunities and lead projects, and their capabilities are sometimes below the expectations of the employers [2]. *Self-confidence*, aka self-efficacy, perceived ability, and perceived competence, is a measure of one's belief in their ability to successfully execute a specific activity [3], [4], [5]. According to Bandura, the outcomes that people expect depend heavily on their self-confidence that they can perform the skill [5].

Self-confidence was considered a critical factor that impacts undergraduate students' abilities in programming [6], [7]. For instance, Heggen and Meyers [2] studied students' confidence before joining a program to develop real-word applications. They found that only 25% of the students were optimistic about their abilities in developing software systems before joining a pair-programming program and are far more confident in their leadership abilities after finishing the program. Hanks also measured their students' confidence after practicing with pair-programming and found that the confident students liked pair-programming the most, while the least confident students liked it the least [6].

Software architects gain cumulative architectural knowledge through experience; they make architectural decisions in ambiguous situations and learn by assessing the impacts of these decisions on software [8], [9], [10]. Teaching software architecture is challenging given the nature of software architecture and the characteristics of the learners [11]. For instance, software architecture (1) is a fuzzy concept, challenging to present as a tangible and useful concept to non-experienced software engineers while the learners are used to topics where the problems and solutions could be precisely defined, which do not apply to the case of architecture.

Software architecture students, like programming students, have, often, a self-confidence problem. For example, some of our students expressed, in Spring 2017, that they are not able to use their knowledge to design software architectures. The problem of self-confidence of software architecture students has been addressed, in our opinion, by focusing on practicing with design patterns [12] or adopting the clinical mode [13].

We conducted informal meetings with colleagues to assess the factors that may impact the self-confidence levels of software architecture students. The goal was to identify the basic aspects that we could act on to improve the students' confidence levels. The consultation led to the selection of variables: course expectations, cognitive levels, preferred learning methods (e.g., passive, active), and critical thinking.

We developed a questionnaire with open-ended questions to study the relationships between students' self-confidence and their expectations, cognitive levels, preferred learning methods, and critical thinking. We gave the questionnaire to the students who took the course in two subsequent semesters: Fall 2017 and Spring 2018. In total, 110 students out of 138 students took the survey. We coded the answers of each student using the descriptive coding method [14], and used the frequency technique as in-text analytics [15], [16] to assess the dependency between the students' self-confidence levels and their expectations, cognitive levels, preferred learning methods, and critical thinking.

The paper is organized as follows. Section II discusses related work. Section III describes the course design. Section IV describes the research method. Section V explores the collected data. Section VI analyses the relationships between self-confidence and expectations, cognitive levels, preferred learning methods, and critical thinking. Section VII discusses the impacts and limitations of the study and Section VIII concludes the paper.

II. Related Work

This section reports about existing work on exploring ways to teach software architecture.

Valentim et al. [17] performed a study with 17 post-graduate students on student perceptions of applying design thinking to design mobile applications. The students appreciated the process as they find it useful. However, they find it challenging to apply because they need to think creatively and generate ideas. Besides, they found the application of the techniques (e.g., workshops and brainstorming) useful but challenging given the lack of team connection and critical thinking [18].

Heesch and Avgeriou [19] surveyed 22 undergraduate software engineering students in the Netherlands, aiming to find out the natural reasoning process during architecting. They found that most of the students tried to understand and consider the architectural drivers and emphasize the quality attribute requirements. However, many students did not identify the most challenging requirements nor prioritize them. In addition, most of the students affirmed that they used the requirements to identify design options and preferred well-known solutions rather than unknown alternatives. They also found that while more than half of the students affirmed that they considered the pros and cons of alternative solutions, many did not consciously make trade-offs between requirements.

Schriek et al. [20] propose a card game to help novice designers design reasoning.[1] The cards represent the reasoning techniques: problem structuring, option generation, constraint analysis, risk analysis, trade-off analysis, and assumption analysis. The authors evaluated their technique's efficacy using twelve groups of students who took the software architecture course. The study showed that the cards trigger reasoning and lead to more discussion and reconsideration of previous decisions. The groups who used the card game identify more distinct design elements and spend more time reasoning with the design.

Rupakheti and Chenoweth experimented with teaching undergraduate students software architecture for a decade [12]. They found that teaching the topic is challenging because it contrasts the students' habits in the other computer science courses. For instance, software architecture requires addressing problems in large and complex software, use multiple complex solutions, and is designed from incomplete information. The authors described how they evolved the course from lecture-heavy to a hands-on course that teaches the students how to use architecture patterns to address Quality Attributes (QAs) in lab experiments. The authors found that the use of labs reinforced the students learning.

Ali and Solis [21] studied the perception of master students on the easiness of use, usefulness, and willingness to use the Attribute-Driven Design (ADD) method in the future. They found that the students find the architecture design method useful but not easy to use and are neutral in term of willingness to use the ADD.

Ben Othmane and Lamm [22] studied the factors associated with the mindsets of software architecture students. They found that students' mindset weakly correlates with their cognitive levels and is related to their expectations. They also found that the students who prefer practicing software architecture have more open mindsets than those who prefer quizzes.

We did not find studies on the self-confidence of undergraduate students to design software architecture–recall that the issue has been investigated for programming students [6], [7]. We initiate the discussion about measuring the students' self-confidence and assessing the factors that may impact it. Recall that this trait is essential for students to take the initiative and lead projects.

III. Course Description

The course Software Architecture Design is an undergraduate-level course for software engineering and computer engineering programs. Before taking the class, the students take a class on developing web applications. The course is given two times a year. Each semester, the class meets two times a week for 14 weeks, each of 75 min.

The goal of the course is to train the students in designing software architecture. The course uses the Attribute-Driven Design (ADD) method [23]. The students acquire the knowledge needed to design software architecture and learn how to apply the ADD method, which is a process-based approach to the design of software architecture [24], [23]. The objectives are:

1) understand and explain the importance of software architecture,
2) understand the relationships between software quality attributes and software architecture,
3) Gain ability to elicit software architecture drivers,
4) Understand the roles of a set of architecture styles, patterns, and tactics in software architecture,
5) Apply the attribute-driven method to design and evaluate software architecture.

The students work in groups on in-class activities. The activities include answering questions that need reflection, working on exercises, and simulating architecture meetings. The case studies provided by [23] were useful for the students to see the use of the techniques.

The students were requested to practice the knowledge that they acquire in the lecture sessions on group and individual assignments. The students work in groups on projects in three group assignments: gathering architectural drivers, designing the architecture of the new version of a given software and implementing the architecture they designed. The individual assignments enforce the experience that the students obtained from the project. The group assignments are related to an Internet of Things (IoT) project, while the individual assignments are related

---

[1] Design reasoning means using logic and rational thinking to make decisions.

TABLE I
Questionnaire.

ID	Factor	Question
1	Expectation	What was your expectation of the course before taking it?
2	Cognitive level	Assume you are given a project and asked to design an architecture for it. How would you do the design?
3	Self-Confidence	How much confidence would you have about your design?
4	Critical thinking	What are the differences between designing the architecture of a Web application and the one of an IoT system?
5	Preferred learning method	What is/are the method(s) that helped you better learn software architecture?

to IT projects. This is expected to give the students an experience with the two domains.

IV. Research Method

The best solution to assess the relationships between student's self-confidence level and expected dependent variables (course expectations, cognitive levels, preferred learning methods, and critical thinking) is to specify a set of closed questions (e.g., using Likert scale and variable categories) and use inference statistics techniques. Since, we do not know the different categories for each of the dependent variables, we conducted a qualitative study. The study uses students' free-text responses to a questionnaire as the data source. We discuss the preparation of the study, the data collection, and the data analysis activities.

Preparation of the study. We discussed the course with colleagues and identified a set of factors that we expected to be associated with students' self-confidence, which are: (1) course expectations, (2) cognitive levels by the students, (3) preferred learning methods and (4) critical thinking. Therefore, we used expert opinions rather than literature review to identify the factors that may impact the students' self-confidence in designing software architecture. The factors were used to develop a set of questions to measure them, listed in Table I.

We developed an anonymous, electronic questionnaire using Google Form and made it available online for the students in November 2017 (for Fall 2017 semester) and April 2018 (for Spring 2018 semester).[2] (The students answer the questionnaire at the end of the semester.) The submissions were anonymous, but the students had to tell the instructor that they participated in the study to get their bonus points.

Data collection. One-hundred ten students answered the questionnaire in Fall 2017 and Spring 2018. We used the thematic analysis [14] method to extract insights from questionnaire responses. The thematic analysis approach is a method for identifying, analyzing, and reporting patterns (themes) within data [25]. It allows exploring phenomena through interviews, stories, or observations [26]. First, we read all the answers to the questions and extracted the thematic code representing each of the answers. A code is a word or short phrase identifying the essence of a portion of language-based or visual data [27]. At the end of this step, we assigned codes to each of the one-hundred-ten students' responses and obtained a set of categories for each of the factors of Table I.

The cognitive levels of the students according to Bloom taxonomy [28] are commonly assessed either using test questions or reflection write-ups [29]. We used in this study the verbalization[3] used by the students in their responses to (reflection) Question 2 to identify the cognitive level of each student. The association of the verbs to the different levels is based on the author's domain knowledge. For instance, Participant (`P20`) said *"The design would vary depending on what the project requirements and architectural drivers were. Once I decided on an optimal reference architecture, I would go through the iteration design process and make sure that appropriate design decisions were made to address every architectural driver that was identified in the project description."* The codes extracted from the statements are: apply the design process, select reference architecture, and evaluate. Since the code "evaluate" is classified in the cognitive levels as `Evaluation`, we ranked the student at level `Evaluation`–that is, the code associated with the higher cognitive level is selected.

Next, we counted the frequencies of the different codes/-categories/levels used in the responses to each of the questions of Table I and observed the patterns in these data. We discuss the data that we collected in Section V.

Data analysis. We represented the relationships between the students' self-confidence levels and each factor affecting their self-confidence using matrices–we use one matrix for each factor. The columns of a matrix are the self-confidence levels and the rows are the codes/code-categories of the factor being studied. The elements are the frequencies of the students who belong to the given factor category and given self-confidence level. We use Chi-square independence test [30] to evaluate the dependencies between self-confidence and the related factors.

V. Data Collection

This section summarizes the responses of the students to the questionnaire and discusses the results.

A. Self-confidence

This subsection discusses the results of the analysis of the responses to the question: `How much confidence would you have about your design?` We classified the extracted codes into five categories: high, moderate, fair, and no self-confidence, in addition to no definite answer category. Table II shows the codes that we used for each level, and Figure 1 shows the frequency of these levels. The number of students who have high self-confidence

---

[2] The project was granted an IRB exemption.

[3] See for example: https://adp.uni.edu/documents/ bloomverbscognitiveaffectivepsychomotor.pdf

TABLE II

Codes used to express self-confidence of the students in their architecture designs.

ID	Confidence level	Codes
1	Confident	very confident, confident, pretty confident
2	Moderate	somewhat confident, moderate, decent, somewhat confident, relative, quite confident
3	Fair	fair confidence, not very/extreme-ly/overly confident
4	No confidence	not confident, not great
5	No definite answer	no definite answer, no answer, not sure

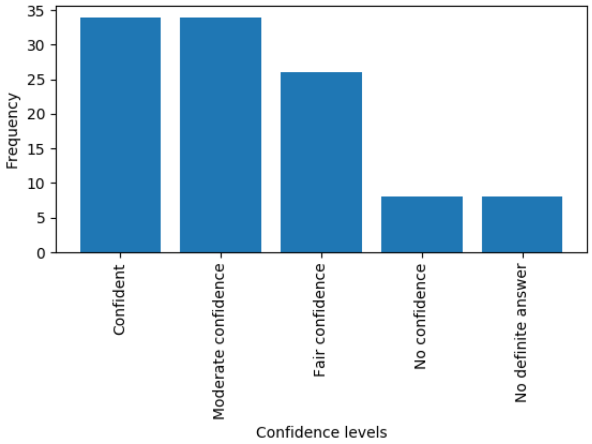

Fig. 1. Frequency of self-confidence levels.

level is 34 (31%). These students seems to be comfortable applying design processes, such as ADD. For instance, student `(P24)` expressed that by saying that *"I feel I would be very confident in my design because I think the design process does a good job of ensuring the architecture considers and satisfies all the drivers. So as long as I am successful in compiling a thorough list of drivers, I think the design will turn out well."*. Out of the remaining students, we see that eight students (about 7%) did not provide definitive answers. Students who elaborated their answers expressed the need for references to ensure the efficacy of their designs.

B. *Student' expectations about the course*

This subsection discusses the analysis results of the answers to the question: `What was your expectation of the course before taking it?` Figure 2 shows the frequency of the students' expectations about the course. In general, most of the students expected the course to be about the design of architecture (32.7%), architecture styles (20.9%), and design process (14.5%)–note that some students specified more than one course expectation category. We observe that eight students related the course to other courses and two students relate the course to experiences they had in their internships. We also observe that the number of students who did not have a clear expectation about the course is 38 (34.5%). The reason for this high percentage is possibly due to the fact that the course is required for their programs.

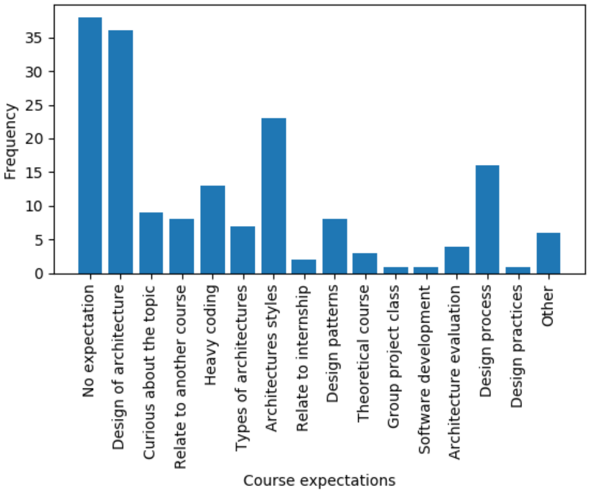

Fig. 2. Frequency of expectations.

TABLE III

Cognitive levels of the students.

ID	Level	Codes
1	Creating	adjust design process
2	Evaluating	identify trade-offs, identify risk, architecture evaluation
3	Analyzing	analysis
4	Applying	identify architecture drivers, get requirements, meet stakeholders, create design, apply the design process, do as in assignments, modify reference architecture
5	Understandi	select reference architecture, select architecture style, select architecture type, make diagrams
6	Remembering	
7	Irrelevant	

C. *Cognitive levels*

This subsection reports the results of the analysis of the replies to the question: `Assume you are given a project and asked to design an architecture for it. How would you do the design?` We coded the responses of the students. We classified the extracted codes based on the new Bloom taxonomy cognition levels [28]. Table III shows the classification of the codes to Boom's cognition categories, and Figure 3 provides the frequency of the cognitive levels.

We observe that most of the students have "applying" and "understanding" cognitive levels. The number of students who provided irrelevant answers is sixteen (14.5%). Many of these students specified that they need more details to decide how to proceed with the design or provided non-useful answers such as *"I would probably try and layout the entire system's architecture in one go because the process of iterations confused me."* `(P11)`. In addition, we note that eight students had the "evaluating"

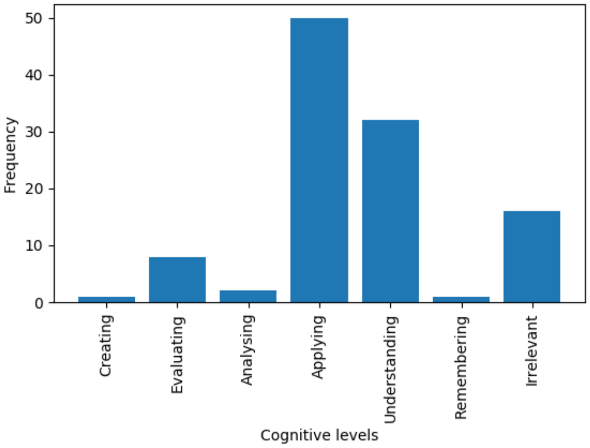

Fig. 3. Frequency of the cognitive levels.

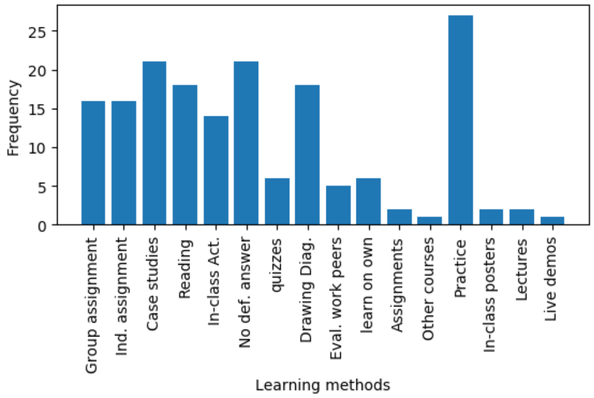

Fig. 4. Frequency of learning methods.

cognitive level, one student had the "creating" level, and one student had the "remembering" level.

D. Preferred learning methods

This subsection discusses the results of the analysis of the answers to the question: `What is/are the method(s) that helped you better learn software architecture?` Figure 4 shows the frequency of preferred learning methods–a student can specify a set of methods.

We observe that the number of students who prefer practice is 27 (24.5%) and the number of students who prefer reading is 18 (16.3%). We cannot distinguish students who prefer active learning methods from students who prefer passive learning methods because each student can specify both active and passive learning methods, e.g., reading and practice. The figure shows, however, that the active learning methods are specified by the students more frequently than the passive learning methods.

E. Critical thinking

We assess students critical thinking by evaluating their abilities to identify the differences between the architectures of Web applications and IOT-based software. This subsection reports the results of the analysis of the replies to the question: `What are the differences between designing the architecture of a Web application and the one of an IoT system?` Table IV provides the codes that we derived from the responses, which are classified into six categories: architecture drivers, patterns of the solutions' structures, architectural knowledge, simplicity, technology stack, configuration management, no definite answer. Note that some students identified difference in more than one category; i.e., a student could discuss performance, which is an architecture driver, and distribution of the system's components, which is a pattern of the solution' structure.

TABLE IV
Codes associated with the critical thinking aspects.

ID	Aspect	Codes
1	Architecture drivers	reliability, interoperability, scalability, architecture drivers, availability, performance, and security requirements
2	Patterns of the structures of the solutions	communication pattern, components structure (e.g., modularity), control of physic, Objects vs logic computation, interacting actors, access to arch. components, flexibility to add components, integration of complex software
3	Architectural knowledge	Reference architecture, architecture styles, architecture patterns
4	Simplicity	simplicity and complexity
5	Technology stack	technology stack, security protocols, use of hardware, complexity of software
6	Configuration management	Configuration management
7	No definite answer	

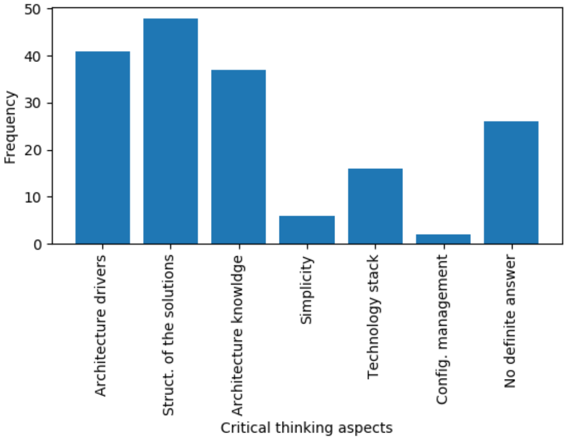

Fig. 5. Frequency of critical thinking aspects.

Figure 5 provides the frequency of the critical thinking aspects. We observe that the number of students who expressed that the two types of systems use different architecture structure patterns is the largest, 48 (43.7%), the number of students who discussed the differences in the technology stack is 16 (14.4%). This is a good results as the students are expected to have limited experience with the technology stack but are expected to reason about

TABLE V
Students' course expectations vs self-confidence levels.

Expectation	1	2	3	4	5
No expectation	10	12	7	7	2
Design of architecture	11	12	9	2	2
Curious about the topic	6	1	0	1	1
Relation to another course	1	3	4	0	0
Heavy coding	2	4	5	1	1
Types of architecture	0	3	4	0	0
Architectures styles	6	10	7	0	0
Relate to internship	1	1	0	0	0
Design pattern	2	2	2	1	1
Theoretical	0	1	0	0	2
Group project class	0	0	0	0	1
Software development	1	0	0	0	0
Architecture evaluation	1	3	0	0	0
Design process	7	4	2	1	2
Design practices	1	0	0	0	0
Other	2	3	1	0	0

the architecture drivers, patterns, and tactics. We observe also that the number of students who did not provide definitive answers is 26 (23.6%). Some of these 26 students reported "they do not know", did not answer the question, or provided non-useful answers such as *"I thought this was a survey, not a test."*

VI. Analysis of the relationships between self-confidence and course expectations, cognitive levels, and preferred learning methods, and critical thinking

In this section, we analyze the relationships between students' self-confidence levels and their expectations, critical thinking, cognitive levels, and preferred learning methods. We use in this analysis the Chi-square independence test [30] and the items frequencies.

A. Students' course expectations.

The Chi-square test confirms a weak association and dependency between the students' self-confidence levels and their course expectations, with χ^2 of 79.6, p-value 0.04, and Cramer V 0.17. Table V provides the frequencies of the course expectations vs. self-confidence levels of the students. We observe that the students who have no expectations are either confident (10 students) or have moderate self-confidence (12 students). The results suggest that the students who have no expectation, expect the course to be about design of architecture expectation, or were curious about the topic have better self-confidence levels. This suggests that the course instructors should ensure that the students have correct expectations from the course.

B. Critical thinking.

The Chi-square test confirms a weak association and dependency between the students' self-confidence levels and their critical thinking, with χ^2 of 39.88, a p-value of 0.022, and Cramer V 0.15. Table VI provides the frequencies of the students' critical thinking aspects vs. their self-confidence levels. We observe that the students who have high self-confidence discuss more the differences between Web-based and IOT-based applications in terms of architecture drivers and patterns of the solutions' structures and, to a lesser frequency, the architecture knowledge and technology stack while the students who have moderate self-confidence discuss the differences in the patterns of the structure of the solutions and the architecture knowledge and, to lesser frequency, the differences in the architecture drivers between the two software types. Thus, we observe students who have high, moderate, and fair self-confidence are mainly able to identify the differences in the architecture drivers, patterns of the solutions' structures, and architecture knowledge between the two architecture types and the students who did not express their critical thinking capability have mostly fair self-confidence.

TABLE VI
Students' self-confidence levels vs the architecture aspects that they mentioned when differentiating the architectures of Web applications and IOT-based software.

Identify Diff. btw. concepts	1	2	3	4	5
Architecture drivers	18	10	9	3	1
Patterns of the structures of the solutions	16	21	4	3	4
Architectural knowledge	6	17	12	1	1
Simplicity	0	1	3	1	1
Technology stack	5	3	3	3	2
Configuration management	2	0	0	0	0
No definite answer	4	7	10	2	3

TABLE VII
Relationship between cognitive levels and self-confidence levels.

Cognition category	1	2	3	4	5
Creating	0	2	0	0	0
Evaluating	1	3	5	3	2
Analyzing	3	0	0	0	0
Applying	23	29	22	8	6
Understanding	18	18	8	1	1
Remembering	0	0	1	0	0
Irrelevant	6	7	5	1	3

C. Student' cognitive levels.

The Chi-square test confirms the independence between the students' cognitive levels and their self-confidence levels, χ^2 of 31.73 and p-value of 0.13-the significance level is low. Table VII provides the frequencies of the students' cognitive levels vs. their self-confidence levels. We observe that the students who have applying cognitive levels do not necessarily have high self-confidence levels, and the students who have understanding cognitive levels do not necessarily have low self-confidence levels. However, we observe that the students who have creating and evaluating cognitive levels have mostly moderate, fair, and no self-confidence. The paradox that high performers exhibit under-self-confidence is documented in other domains such as accounting [31]. A possible reason is that the high performers know the limit of their abilities.

D. Students' preferred learning methods.

The Chi-square test confirms the independence between the students' self-confidence levels and their preferred

TABLE VIII

Students' self-confidence levels vs preferred learning methods.

Learning method	1	2	3	4	5
Group assignments	4	5	4	3	0
Individual assignments	5	4	4	2	1
Case studies	4	11	1	1	4
Reading	5	8	4	0	1
In-class group activities	5	4	4	1	0
No definitive answer	9	7	4	1	0
Quizzes	2	1	2	0	1
Drawing diagrams	6	5	5	0	2
Evaluate work of peers	1	1	3	0	0
Learning on own	1	1	2	1	1
Assignments	0	1	0	1	0
Other courses	0	0	0	1	0
Practice	8	9	7	2	1
In-class posters	0	1	1	0	0
Lectures	0	1	0	0	1
Live demo	1	0	0	0	0

learning methods, χ^2 of 63.69 and p-value of 0.34. Table VIII provides the frequencies of the students' preferred learning methods vs. their self-confidence levels. We observe that the students who prefer to learn from case studies have mostly moderate self-confidence level (11 out of 21), the students who prefer practice have mostly high, moderate, or fair self-confidence levels (24 out of 27 students); the students who prefer reading have mixed self-confidence levels; and the students who did not provide a definitive answer have mostly good or moderate self-confidence) While there is no statistical evidence, the detailed analysis suggests that providing the students with practice opportunities helps them gain better self-confidence levels.

VII. Impacts and limitations of the study

This paper explores a set of factors that we believe are related to undergraduate students' confidence levels, i.e., confidence in their abilities to design software architecture after taking a course on software architecture. The study found that the students' self-confidence is weakly associated with their expectations from the course and their critical thinking to differentiate between the architectures of Web-based and IOT-based applications and does not depend on their cognitive levels and preferred learning methods. Figure 6 depicts these relationships–the color indicates the associated factors.

We reiterate that the students who have high cognitive levels did not have high self-confidence levels, and self-confidence is not associated with the cognitive levels.[4]

We also did not see significant patterns from the analysis of the students' answers who do not have confidence in their ability to design software architecture. We found that these students have varying preferred learning methods (including practice), varying expectations from the class, and different cognitive levels. The results suggests that to improve the self-confidence of the students, the instructor should ensure that the students' have "correct" course expectations and work on improving the students' critical thinking capabilities.

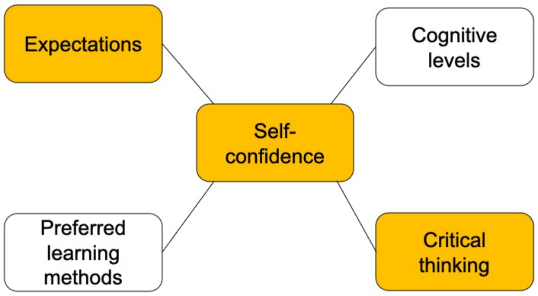

Fig. 6. Self-confidence and expected related factors. The yellow boxes indicates the variables associated with self-confidence.

TABLE IX

Relationship between the critical thinking and cognitive levels.

Differences topics	1	2	3	4	5	6	7
Architecture drivers	1	2	1	23	11	0	3
Patterns of the structures of the solutions	0	5	2	26	12	0	3
Architectural knowledge	0	1	0	22	9	1	4
Simplicity	0	2	0	0	2	0	2
Technology stack	1	3	0	10	2	0	0
Configuration management	0	0	0	0	2	0	0
No definite answer	0	1	0	7	8	0	10

The main limitations of the study follow. First, we did not use a repeatable process to identify the factors that affect the students' self-confidence. The factors used in the study were identified in brainstorming sessions with colleagues: there would be other factors that impact the students' self-confidence that could be worth studying.

Second, the students provided their responses in text, and the authors coded the responses. We acknowledge that the coders' perspective impacts the data extraction, which applies to qualitative research, in general. We, however, revisited the data extraction several times to reduce this limitation. We also cross-checked often the students' answers. For instance, we used Table IX, which describes the relationships between the students' cognitive levels and their critical thinking. The Chi-square test of dependency confirms the dependency between the two factors, χ^2 of 57.71 and p-value of 0.01. The table shows that some of the students (12 out of 22) are not associated with specific cognitive level but provided differences between the architectures of IOT-based and Web-based applications, which leads us to double-check our coding.

The study shows that self-confidence is associated with the critical thinking of the students. This suggests that instructors can change their students' self-confidence by giving them knowledge about alternative solutions for solving given architecture problems, so they understand that there are conditions and implications of using architecture knowledge to solve architecture problems before asking them to apply architecture design methods [32]. The results suggest that the instructors should ensure

---

[4] We note that we cannot correlate the data with the students' assessment scores in the class because we did not request that in the IRB before starting the study.

that the students' expectations are aligned with the course goals and try to use case studies that show contrasts, e.g., performance needs for Web applications and IOT-based software.

VIII. Conclusions

In this paper, the study analyzed the relationships between students' self-confidence levels and their expectations, preferred learning methods, cognitive levels, and critical thinking. The study found that the students' self-confidence levels depend on their expectations from the course and their critical thinking capability but did not find dependency relationships between the self-confidence and students' cognitive levels or preferred learning methods.To improve the self-confidence of the students, the instructor should ensure that the students' have "correct" course expectations and work on improving the students critical thinking capabilities.

References

[1] R. T. Mowday, "Leader characteristics, self-confidence, and methods of upward influence in organizational decision situations," *The Academy of Management Journal*, vol. 22, pp. 709–725, Dec. 1979.

[2] S. Heggen and C. Myers, "Hiring millennial students as software engineers: A study in developing self-confidence and marketable skills," in *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials*, p. 32âĂŞ39, 2018.

[3] C. A. Shoemaker, "Student confidence as a measure of learning in an undergraduate principles of horticultural science course," *HortTechnology*, vol. 20, pp. 683–688, 2010.

[4] A. Bandura, *Social Foundations of Thought and Action: A Social Cognitive Theory*. Prentice-Hall series in social learning theory, Prentice-Hall, 1986.

[5] D. Druckman and R. A. Bjork, *Learning, Remembering, Believing: Enhancing Human Performance*, ch. Self-Confidence and Performance, pp. 173–206. Washington, DC: The National Academies Press, 1994.

[6] B. Hanks, "Student attitudes toward pair programming," pp. 113–117, 2006.

[7] V. Ramalingam, D. LaBelle, and S. Wiedenbeck, "Self-efficacy and mental models in learning to program," in *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, p. 171âĂŞ175, 2004.

[8] U. v. Heesch and P. Avgeriou, "Mature architecting - a survey about the reasoning process of professional architects," in *2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, pp. 260–269, June 2011.

[9] A. Tang, M. Razavian, B. Paech, and T. Hesse, "Human aspects in software architecture decision making," in *2017 IEEE International Conference on Software Architecture (ICSA)*, (Gothenburg, Sweden), April 2017.

[10] V. Clerc, P. Lago, and H. van Vliet, "The architect's mindset," in *Software Architectures, Components, and Applications* (S. Overhage, C. A. Szyperski, R. Reussner, and J. A. Stafford, eds.), (Berlin, Heidelberg), pp. 231–249, Springer Berlin Heidelberg, 2007.

[11] M. Galster and S. Angelov, "What makes teaching software architecture difficult?," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, pp. 356–359, 2016.

[12] C. R. Rupakheti and S. Chenoweth, "Teaching software architecture to undergraduate students: An experience report," in *Proc. of the 37th International Conference on Software Engineering - Volume 2*, ICSE '15, pp. 445–454, 2015.

[13] M. McCracken, I. Hsi, H. Richter, R. Waters, and L. Burkhart, "A proposed curriculum for an undergraduate software engineering degree," in *Thirteenth Conference on Software Engineering Education and Training*, pp. 246–257, March 2000.

[14] J. Saldaña, *The Coding Manual for Qualitative Researchers*. Sage, 2015.

[15] M. R. Mehl, *Handbook of multimethod measurement in psychology*, ch. Quantitative Text Analysis, pp. 141–156. American Psychological Association, 2006.

[16] M. Gentzkow, B. Kelly, and M. Taddy, "Text as data," *Journal of Economic Literature*, vol. 57, pp. 535–74, Sep. 2019.

[17] N. M. C. Valentim, W. Silva, and T. Conte, "The students' perspectives on applying design thinking for the design of mobile applications," in *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pp. 77–86, May 2017.

[18] J. Heywood, *Empowering Professional Teaching in Engineering: Sustaining the Scholarship of Teaching*. Synthesis Lectures on Engineering, Morgan and Claypool, March 2018.

[19] U. van Heesch and P. Avgeriou, "Naive architecting - understanding the reasoning process of students," in *Software Architecture: 4th European Conference, ECSA 2010, Copenhagen, Denmark, August 23-26, 2010. Proceedings* (M. A. Babar and I. Gorton, eds.), (Berlin, Heidelberg), pp. 24–37, Springer Berlin Heidelberg, 2010.

[20] C. Schriek, J. M. E. van der Werf1, A. Tang, and F. Bex, "Software architecture design reasoning: A card game to help novice designers," in *Proc. 10th European Conference on Software Architecture (ECSA)*, (Copenhagen, Denmark), pp. 22–38, Dec. 2016.

[21] N. Ali and C. Solis, *Exploring How the Attribute Driven Design Method Is Perceived*, pp. 23–40. Morgan Kaufmann, 2014.

[22] L. Ben Othmane and M. Lamm, "Mindset for software architecture students," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 306–311, Jul 2019.

[23] H. Cervantes and R. Kazman, *Designing Software Architectures: A Practical Approach*. Addison-Wesley Professional, 1st ed., 2016.

[24] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, "Generalizing a model of software architecture design from five industrial approaches," in *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, pp. 77–88, 2005.

[25] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[26] L. M. Connelly, "What is phenomenology?," *MedSurg Nursing*, vol. 19, no. 2, p. 127âĂŞ 129, 2010.

[27] N. Bricki and J. Green, "A guide to using qualitative research methodology," 2 2007.

[28] L. W. Anderson, D. R. Krathwohl, P. W. Airasian, K. A. Cruikshank, R. E. Mayer, P. R. Pintrich, J. Raths, and M. C. Wittrock, *A taxonomy for learning, teaching, and assessing : a revision of Bloom's taxonomy of educational objectives*. New York, US: Pearson, 2000.

[29] M. Harvey, C. Baumann, and V. Fredericks, "A taxonomy of emotion and cognition for student reflection: introducing emocog," *Higher Education Research & Development*, vol. 38, no. 6, pp. 1138–1153, 2019.

[30] W. G. Cochran, "The χ^2 test of goodness of fit," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 315–345, 1952.

[31] S. P. Ravenscroft, T. R. Waymire, and T. D. West, "Accounting students' metacognition: The association of performance, calibration error, and mindset," *Issues in Accounting Education*, vol. 27, no. 3, pp. 707–732, 2012.

[32] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, "A general model of software architecture design derived from five industrial approaches," *Journal of Systems and Software*, vol. 80, no. 1, pp. 106 – 126, 2007.