Threshold Concepts and Skills in Software Architecture: Instructors' Perspectives

Usman Nasir
Department of Software Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
usman.nasir@bth.se

Muhammad Laiq
Department of Software Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
muhammad.laiq@bth.se

***Abstract*—Context: Software Architecture is an important subject and core course in the Software Engineering degree that teaches multiple co-mingled concepts and skills. The academic community believes that students find the course difficult to grasp and master. However, to revise the course curriculum, identification of threshold concepts and skills can help prioritize topics ensuring alignment with course learning objectives.**

Objective: The aim of the study was to identify threshold concepts and skills in Software Architecture to help instructors focus on redesigning the curriculum and improving didactics.

Method: We applied the Delphi technique to identify threshold concepts and skills from instructors with teaching experience in university-level Software Architecture courses.

Results: We identified eleven threshold concepts and nine threshold skills with more than 80% agreement among the participants. Six out of twenty-one threshold concepts and skills achieved 100% agreement from participants indicating high consensus. Furthermore, all participants agreed that applying skills to design Software Architecture is more difficult than understanding the underlying theoretical concepts.

Conclusion: The Software Architecture course is demanding, and the industry expects graduating students are prepared to design solutions for complex systems. The identified threshold concepts and skills can help academics to redesign Software Architecture courses, focus on hard to grasp topics, and offer support for skills that are difficult to master. Often theoretical concepts are considered more important than the skills required to apply them in practice. However, instructors agreed that students struggle to apply theoretical concepts in designing solutions. Thus, skills development should be equally emphasized.

***Index Terms*—Software Engineering Education, Teaching Software Architecture, Threshold concepts, Threshold skills**

I. Introduction

The software is a complex and abstract product primarily developed by Software Engineers to solve clients' problems. Software Engineers face many challenges as they try to design solutions to a given problem, including understanding the problem, designing, and creating solutions. The first step in designing the software solution is developing a blueprint that establishes the software's overall structure and constituent components. This blueprint is referred to as Software Architecture, describing building blocks (in terms of code units), their relationships, and communication and coordination mechanisms.

Software Architecture, its concepts, and its design techniques are considered a core part of the Software Engineering curriculum. The Software Architecture course offers multiple skills and capabilities to the students, ranging from analysis of requirements, design of architecture, making technical decisions, and documenting the architecture in various forms for different stakeholders. Typically, these are advanced-level courses that are offered when students have acquired an understanding of multiple aspects of software development, such as programming, requirement engineering, development processes, and software quality [1].

Software Architecture courses at the university level are usually demanding as they aim to teach realistic architectural design based on complex case problems [2]. Both learners and instructors consider Software Architecture a tricky subject due to its abstraction, fuzzy concepts, difficulties in designing architectural models, and students' lack of experience in software design and development [3].

Many researchers suggest that determining the threshold concepts in a subject could help its instructors to focus on redesigning the curriculum and improving didactics [4]–[6]. Thus, we are motivated to identify threshold concepts and skills to help instructors redesign the curriculum and improve prevalent teaching practices in Software Architecture courses. To the best of our knowledge, no studies have been carried out to date to explore threshold concepts and skills in the domain of Software Architecture.

The overarching objective of this study is to identify threshold concepts and skills in the Software Architecture course. Thus the contribution of this study is the collection of consensus-based threshold concepts and skills that can help instructors redesign and improve Software Architecture courses.

The outline of the paper is as follows. Section II, we describe the background, i.e., threshold concepts and skills, related work, and the Delphi technique. Section III describes the research method applied, data collection, study participants, the Delphi process, and data analysis. In Section IV, we present our findings. In Section V, we discuss our opinion, the implications of our work, and the study's limitations. Finally, in Section VI, we conclude our paper and describe future work.

II. Background

A. Threshold Concepts & Skills

According to Meyer and Land [5], in every course (discipline), there are specific concepts that are difficult to grasp, change the students' perception and often open new ways of thinking. Such concepts are irreversible, transformative, and challenging, thus requiring considerable effort from students to grasp them. Meyer and Land [5] describe threshold concepts as having the following characteristics:

1) **Transformative:** Thinking style is altered after understanding the concept.
2) **Troublesome:** Potentially difficult for students to grasp or understand.
3) **Irreversible:** Once acquired, they cannot be forgotten.
4) **Integrative:** Concepts are tied/linked in a previously unknown manner.
5) **Bounded:** Define boundaries within the discipline.

There is a significant body of work available across multiple disciples with identified threshold concepts and the theory's applicability. Many have argued that "not all thresholds within some disciplines are concepts" [7] and others argue that students not only find some concepts difficult but also have problems with learning how to apply them [8].

An expansion of the definition of the threshold concept has been proposed in disciplines where "practice" or "applying" (i.e., skill) is more important than theoretical learning [6]. Skill is defined as *"an ability to perform a function, acquired or learned with practice"* [9]. Akin to the threshold concept, there are transformative skills that are difficult to acquire or master. For any skill to be considered as a threshold skill, the following criteria are described by Thomas et al. [9]:

1) **Troublesome:** Skills can be complex and thus need time to learn and master.
2) **Transformative:** The learner starts to solve problems they could not solve earlier or solve them in a better manner.
3) **Integrative:** Once a skill is acquired, the learner sees other way to apply it.
4) **Semi-irreversible:** Once a skill is acquired, it is unforgettable but needs review or practice to apply continually.
5) **Associated with practice:** Need to keep practicing to maintain proficiency.

Computing Education and Software Engineering (or software solution design in particular) are two such domains where theoretical foundations need to be complemented with capabilities or skills [8], [9].

B. Related Work

There are very few studies that have previously investigated the threshold concepts related to Computing and Software design. Still, we have not been able to find a single study that has directly explored threshold concepts and skills in Software Architecture.

The seminal work by Eckerdal et al. [10] is the first one to explore the theory of threshold concepts in the context of computer science topics and presented two core topics, abstraction, and object-orientation, as possible threshold concepts.

Boustedt et al. [11] is another work that has empirically investigated both students' and instructors' views about threshold concepts in computing education. They used informal interviews and questionnaires for data collection and reported **object-oriented programming and pointers** as two threshold concepts.

Kallia and Sentance [12] used a three-round Delphi approach for the identification of threshold concepts in programming (computing education). The potential threshold concepts reported by computing teachers are **function arguments, function call, program control flow, function parameters and their passing, procedural decomposition, recursion, variables and their scope, and abstraction**. Their conclusion also reports that "most participants based their suggestions on the troublesome characteristic," ignoring transformative and integrative characteristics.

Sander et al. [8] performed an exploratory study to identify threshold concepts in computing and interviewed students about their perceptions. During interview transcription analysis, they observed that most students spoke of difficulties in applying their learning. This reinforces their view that programming (writing computer programs) is a skill and may not be treated as a collection of concepts. This deviation led them to propose that "the acquisition of particular skills, and not the concepts related to these skills, can act as thresholds."

Thomas et al. [9] use the idea of threshold skills and threshold concepts to identify threshold skills related to software design activity. They used experimentation and asked students to design a software solution. The students (subjects) modeled solutions using static and behavioral diagrams that were later evaluated by a panel of expert teachers to identify which software design skills were not mastered by the participants. They conclude that software design (as an activity) is a threshold skill.

The work, as mentioned earlier, has focused on identifying threshold concepts and threshold skills in computing and software design. However, in this study, we are identifying threshold concepts and skills in the Software Architecture course.

C. Delphi Technique

The Delphi method is a well-structured and organized approach for collecting a consensus view of experts and is applied as a data collection method for exploratory topics.

The fundamental goal of a Delphi technique is to bring together participants (experts) to answer a specific question and carry out rounds to reach a consensus. The procedure and number of rounds vary according to the research question. Conventionally expert opinions are collected in the first round. In the second round, group feedback is obtained about the responses. More rounds can be conducted until a consensus is reached. Typically 75% agreements on any opinion in the second or third round are considered enough to reach a consensus opinion [13].

Researchers use the Delphi technique, where the evidence seems to be limited or dependent, to collect opinions from a panel of experts, which gradually and systematically evolves into a consensus-based view [14]. Some of the key advantages of using Delphi are that it helps avoid the experts' confrontation and ensures anonymity as consensus is being developed in Delphi rounds [13], [14].

III. Study Design

Software Architecture is considered as designing for the bigger picture where the solution revolves around the software solution's building blocks thus, concepts and mastery of skills are very important aspects to be identified.

This study aims to identify threshold concepts and skills in Software Architecture. More precisely, this study is motivated to answer the following research question:

RQ: What are threshold concepts and skills in Software Architecture?

A. Research Method

Cousin [4] believes that an inquiry into threshold concepts should incorporate multifaceted views involving students, instructors, and practitioners.

For this study, we choose to elicit instructors' perspectives for identifying threshold concepts and skills. Instructors' perspectives are broader and more valuable as they teach similar or overlapping concepts in the disciplinary program [15].

Many researchers [16]–[18] have identified threshold concepts from instructors' perspectives in their respective subject areas using expert interviews, focus group sessions or traditional surveys. Barradell [15] has pointed out that one significant challenge of conducting interviews with instructors is that the results lead to disparate opinions. The disparity of views could be due to the lack of homogeneity of respondents' experience and level of interaction with course contents. To address this challenge, we decided to use the Delphi method [19].

Delphi technique is flexible, economical, and convenient to elicit expert opinions, i.e., people can respond at their convenience. In contrast, interviews require scheduling in advance, which becomes a challenge for instructors. Furthermore, interview responses might pose challenges during analysis, whereas Delphi survey responses are not simpler to analyse [20].

In the following subsections, we describe Delphi's process, survey questionnaire, and participants (experts).

1) Delphi Process: Figure 1 represents the two-round Delphi process applied to collect threshold concepts and skills of Software Architecture in this study.

- **Survey design and piloting**: In the first step, a questionnaire as a data collection instrument was designed based on open-ended and closed-end questions. The draft questionnaire was piloted with a senior colleague (who is well versant with software architecture as a knowledge area and was not part of the Delphi expert panel) to check the language and instrument design. Based on the feedback, minor language adjustments were carried out.
- **Round-1**: Following the classical Delphi procedure [19], we initiated the first round by emailing the questionnaire to participants, asking open-ended questions presented in Table I to gather an initial list of potential threshold concepts and skills based on instructors' perspectives.
- **Round-2:** In the second round, we summarized the collected potential threshold concepts and skills. A new questionnaire was developed and sent out to collect participants' agreements/disagreements based on a 4-point Likert scale (Strongly Agree, Agree, Disagree, Strongly Disagree). The second round produced the consensus opinion of the participants about potential threshold concepts and skills.

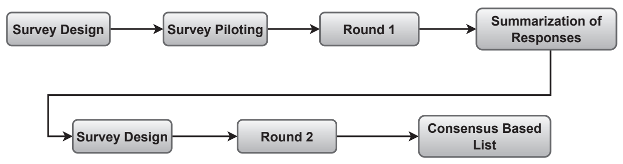

Fig. 1. Delphi Process

2) Survey Questionnaire: Our questionnaire design is inspired by Nicola-Richmond et al.'s [6] survey instrument designed to collect threshold concepts and capabilities (skills) in the field of occupational therapy.

The survey questions (1-3) (Table I) were used to collect concepts and skills from instructors' knowledge, and experience are adapted from Nicola-Richmond et al.'s [6] study, whereas questions (4-6) used to collect instructors' perceptions regarding the difficulty of understanding theoretical concepts and applying skills are adapted from Kallia and Sentance's work [12].

3) Conceptual Characteristics: It is not essential that any potential concept or skill within the discipline exhibit all characteristics to be considered a threshold concept or skill [6]. We chose troublesome, transformative, and irreversible concepts and skills as threshold concepts and skills relevant to Software Architecture.

4) Participants: For Delphi, relevant people should be used as panel experts, preferably subject experts [14]. We identified nine (09) instructors working at a well-reputed technical university in Sweden. These experts have experience in teaching Software Architecture course[1] that offers the following major concepts and skills:

- Architectural drivers, Quality attributes and their role in architectural design, Patterns and Tactics, Views.
- Architectural design process, Architectural evaluation, Documenting Software Architectures.

[1]https://bit.ly/3DmcJHk

TABLE I
DELPHI SURVEY QUESTIONS

1. *What software architecture concepts or skills do you think students find most difficult to grasp/understand? And why?*
2. *What concepts or skills do you think transformed your students' views, interpretation, or understanding of software architecture? And why?*
3. *What software architecture concepts or skills do you think will be unlikely to be forgotten or unlearned by students? And why?*
4. *Do you think that students can design software architecture even without developing an understanding of underlying theoretical concepts? (Yes/No/not sure)*
5. *Do you think students face difficulties in applying skills taught in a Software Architecture course even if they develop a theoretical understanding of corresponding concepts? (Yes/No/not sure)*
6. *Which one is difficult for students: (a) applying skills to design a Software Architecture or (b) understanding concepts of designing Software Architecture? (a/b/not sure)*

A total of six (06) instructors participated in our study and supported primary data collection. Following the Delphi guidelines during the Delphi process, the participants were anonymous to each other. Table II presents the study participants overall teaching experience in years.

TABLE II
STUDY PARTICIPANTS

Participant	Teaching Experience in Years
P1	$>$ 10 years
P2	$>$ 10 years
P3	$>$ 10 years
P4	$>$ 2.5 years
P5	$>$ 2 years
P6	$>$ 1.5 years

IV. RESULTS AND ANALYSIS

The first round of the Delphi collected 72 threshold concepts and skills (22 troublesome concepts and 13 skills). Two researchers independently summarized responses by merging repeating concepts and skills into single statements. After summarizing, 53 potential threshold skills and concepts were identified for validation and made part of the unified list for the second Delphi round.

After the second round, concepts and skills with more than 80% of participants' agreement (Strongly Agree and Agree) were selected as a threshold concept or skill. Kallia et al. [12] used 75% agreement of their participants as consensus criteria for identifying threshold concepts in computer programming from teachers' perspectives. In contrast, we have used more stringent criteria (80% agreement) to improve the confidence of consensus.

A. Threshold Concepts in Software Architecture

Table IV presents the consensus-based list after Delphi's round 2 for threshold concepts. In total, we identified eleven threshold concepts that satisfy the consensus criteria of agreement of more than 80% of the participants. Out of eleven concepts, three concepts achieved 100% consensus among the participants:

1) Architectural Drivers.
2) Architectural Patterns/Styles.
3) Difference between architectural decisions at Design-time, Build-time, and Run-time.

Architectural Drivers are the functional requirements, quality attributes, assumptions, and stakeholder concerns that are the basis of architecture design [21]. In practice, the architectural drivers are very descriptive and often have domain-specific language, alien to software architects. It is important that an architect fully understands these drivers before designing the solutions [21].

According to one of the participants, the student "fail to grasp architectural drivers (domain-specific) and do not recognize their importance" as an input in the architecture design process. This failure may result in an ineffective architectural solution or design. On an interesting note, architectural drivers are built upon concepts not taught in Software Architecture courses directly, and students are expected to have been taught these in pre-requisite courses.

B. Threshold Skills in Software Architecture

Table V presents the consensus-based list after Delphi's round 2 for threshold skills. In total, participants agreed with nine (9) threshold skills with 80% consensus criteria. Out of these nine, three skills achieved a 100% agreement:

1) Identifying weaknesses in software architecture design.
2) Associating architectural drivers with the views.
3) Documenting views.

In evaluating the architecture (a threshold skill), it is important to identify risky elements and decisions regarding

TABLE III
THRESHOLD CONCEPTS AND SKILLS (FREQUENCY)

Concept/Skill	Count (before summarizing)	Count (after summarizing)
Troublesome Concepts	22	15
Troublesome Skills	12	9
Transformative Concepts	10	8
Transformative Skills	7	5
Irreversible Concepts	16	11
Irreversible Skills	5	5
Total	**72**	**53**

TABLE IV
THRESHOLD CONCEPTS BASED ON CONSENSUS (ROUND 2)

Threshold Concept	Participants' Agreement (Strongly Agree and Agree)
Architectural Drivers	100%
Architectural Patterns / Styles	100%
Difference between architectural decisions at Design-time, Build-time, and Run-time	100%
Architectural Decisions	83%
Architectural Tactics	83%
Conversion of software architecture into source code	83%
Deployment view	83%
Documenting multiple views and their rationale of view selections	83%
Influence of architectural decisions and their impact on the system's quality	83%
The effect of Architectural styles/patterns on quality attributes	83%
Variation points	83%

the software architecture for early-stage mitigation. Multiple respondents pointed out that students struggle with identifying flaws in architecture developed by them. Students also struggle to present their architecture in different projections (views). One interesting comment from a participant was that students question the need to document multiple design projections (behaviour vs. relationships as two different views).

C. Perception of theoretical concepts and their realization in practice

Survey questions 4-6 were used to obtain participants' perceptions regarding the importance of understanding theoretical concepts and skills to apply those concepts. The responses are presented in Table VI. It can be seen from the results that 50% of the respondents agree that students could design software architecture even without understanding theoretical concepts, and everyone agrees that applying skills to design software architecture is more difficult than understanding the underlying theoretical concepts.

Likewise, 66% of the participants believe that students face difficulties applying learned concepts in practice. These results are in line with Thomas et al. [9] finding, i.e., software design itself is a threshold concept, and students struggle in designing software architecture.

V. DISCUSSION

A. Threshold Concepts and Skills

Our study has identified eleven threshold concepts and nine threshold skills of software architecture. These numbers are in contrast to the number of threshold concepts and skills reported within computing/programming education domains, as they have only pointed to two to a maximum of four concepts. However, this sizable number of threshold concepts and skills does reinforce the notion that software architecture is a difficult course to teach [3].

Concepts of architectural patterns/styles are fundamental design approaches, and the instructors considered them as troublesome concepts. This perception aligns with the view that software solution design is a skill that is difficult to master, as suggested by Thomas et al. [9].

The results include six threshold concepts and skills (3 each) that have achieved 100% agreement of participants. Furthermore, compared to other studies identifying threshold concepts, our study has used a stringent criterion for identification and consensus. We used 80% of the participant's agreement as a selection criterion to improve the confidence of agreement, unlike [12], as they used 75% agreement.

B. Implication for Teaching

The identified threshold concepts and skills could be used to improve the Software Architecture course curriculum and didactics.

Architectural drivers are the business goals, requirements, and constraints that are taken as input for software architectural solution [21]. Teaching the theoretical foundations of architectural drivers and skills to identify and associate these with views can be achieved using the case-based learning approach. Course instructors should provide software architecture case examples to students during lecturing, and assignments should also be case-based [22].

Instructors should ensure that the chosen case examples are relatable and easily understood. Multiple participants shared that they have seen students struggle with case examples in

TABLE V
THRESHOLD SKILLS BASED ON CONSENSUS (ROUND 2)

Threshold Skill	**Participants' Agreement (Strongly Agree and Agree)**
Identifying weaknesses in software architecture design	100%
Associating architectural drivers with the views	100%
Documenting views	100%
Understanding the outcomes of architecture evaluation	83%
Prioritising architectural drivers for design	83%
Specifying quality attribute requirements	83%
Evaluating a given architecture	83%
Applying architectural patterns/styles	83%
Applying tactics	83%

TABLE VI
THEORY V.S PRACTICE

Question	**Response (Count)**
Do you think that students can design software architecture even without developing an understanding of underlying theoretical concepts?	Yes (3) No (2) Not Sure (1)
Which one is difficult for students: (a) applying skills to design a Software Architecture or (b) understanding concepts of designing Software Architecture?	A (6) B (0) Not Sure (0)
Do you think students face difficulties in applying skills taught in a Software Architecture course even if they develop a theoretical understanding of corresponding concepts?	A (4) B (2) Not Sure (0)

the recommended textbooks, as they were legacy systems or students have never seen such systems in real life (telephony switching system).

It is a well-known fact that in undergraduate-level courses, theoretical concepts often are considered more important than the skills required to apply them in practice. Our survey respondents confirm that they feel that students struggle to apply theoretical concepts in designing architectural solutions (see Table VI). Thus, skills development should be prioritized.

The three threshold skills (see Table V) with 100% agreement by participants must be part of the course learning objectives. The skill of identifying weaknesses in software architecture design is hard to acquire and master, as students have limited exposure to the domain, designed solutions, and mechanisms to identify weaknesses in architectural design [3].

Identifying weaknesses in software architecture design and the corresponding threshold concept of architecture drivers are related. A major weakness of any design is its non-compliance with its architectural drivers. Again, the case-based learning method can be effective during lecturing and practical exercises. Hands-on design workshops to teach can be planned in such a manner where students are given a case-based problem (architectural drivers) to design a solution shown to their peers to identify the design's weaknesses.

Architectural design decisions are the lynchpin of any solution, mostly taken on the basis of architectural drivers. It is common practice that design decisions and their rationale are noted for traceability when architecture is designed. Decision-making and documenting rationales are troublesome concepts and skills. The students' understandability of architectural drivers improves as they repeatedly review, decide and change their architectural decisions. Gamification of this has proven to be helpful, where researchers have developed a card-based game (DecidArch [23]) to learn architectural design decision-making.

Game-based and case-based learning activities can help students master the identified threshold concepts and skills. Instructors can use the identified threshold concepts and skills to align their learning activities and even adopt their case examples. Also, the research community can focus on developing interactive tools and learning games and target these identified threshold concepts and skills to improve the didacts for the Software Architecture course.

C. *Limitation of Study*

We have collected teachers' perceptions at a single university, which is a limitation. Although the case university follows a standardized course curriculum and participants are experienced teachers of the software architecture domain; still, these opinions and assertions have limited generalizability.

Another possible limitation is using the instructors' perspective only and not triangulating it with the student or practitioner's perspectives. We believe that due to the exploratory nature of this study, instructors' perspective suffices for now, but for a broader analysis, students or practitioners should be involved.

VI. CONCLUSION AND FUTURE WORK

Software Architecture is a core course, and the academic community believes that the course teaches multiple co-mingled concepts that are difficult to grasp by students. This study has identified threshold concepts and skills in

Software Architecture, which instructors can use to evaluate and redesign their courses around threshold concepts and skills. Threshold concepts and skills are based on instructors' perspectives, thus, it could help other instructors focus their learning activities around them. These identified threshold concepts and skills could be prioritized to ensure students acquire, understand and apply them fully in realistic scenarios.

In the future, we intend to extend this work by collecting students' and practitioners' perspectives as we have only collected instructors' perspectives. The students will add more threshold concepts and validate them from learners' perspectives. Additionally, the software development industry desires software architecture designing skills in a newly hired graduate. Incorporating practitioners' views would help in learning from job market needs and demands.

REFERENCES

[1] The Joint Task Force on Computing Curricula, "Curriculum guidelines for undergraduate degree programs in software engineering," Association for Computing Machinery, New York, NY, USA, Tech. Rep., 2004.

[2] T. Mannisto, J. Savolainen, and V. Myllarniemi, "Teaching software architecture design," in *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, 2008, pp. 117–124.

[3] M. Galster and S. Angelov, "What makes teaching software architecture difficult?" in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 2016, pp. 356–359.

[4] G. Cousin, *Researching Learning in Higher Education: An Introduction to Contemporary Methods and Approaches*. Routledge, 2008.

[5] J. Meyer and R. Land, "Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practising within the disciplines," in *ISL10 Improving Student Learning: Theory and Practice Ten Years On*. Oxford Brookes University, 2003, pp. 412–424.

[6] K. M. Nicola-Richmond, G. Pépin, and H. Larkin, "Transformation from student to occupational therapist: Using the d elphi technique to identify the threshold concepts of occupational therapy," *Australian occupational therapy journal*, vol. 63, no. 2, pp. 95–104, 2016.

[7] C. Baillie, J. A. Bowden, and J. H. F. Meyer, "Threshold capabilities: threshold concepts and knowledge capability linked through variation theory," *Higher Education*, vol. 65, no. 2, pp. 227–246, 2013. [Online]. Available: http://www.jstor.org/stable/23351712

[8] K. Sanders, J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, L. Thomas, and C. Zander, "Threshold concepts and threshold skills in computing," in *Proceedings of the ninth annual international conference on international computing education research*, 2012, pp. 23–30.

[9] L. Thomas, J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, K. Sanders, and C. Zander, "In the liminal space: software design as a threshold skill," *Practice and Evidence of the Scholarship of Teaching and Learning in Higher Education*, vol. 12, pp. 333–351, 2017.

[10] A. Eckerdal, R. McCartney, J. E. Moström, M. Ratcliffe, K. Sanders, and C. Zander, "Putting threshold concepts into context in computer science education," in *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2006, p. 103–107. [Online]. Available: https://doi.org/10.1145/1140124.1140154

[11] J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, M. Ratcliffe, K. Sanders, and C. Zander, "Threshold concepts in computer science: Do they exist and are they useful?" in *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 504–508. [Online]. Available: https://doi.org/10.1145/1227310.1227482

[12] M. Kallia and S. Sentance, "Computing teachers' perspectives on threshold concepts: Functions and procedural abstraction," in *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, ser. WiPSCE '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 15–24. [Online]. Available: https://doi.org/10.1145/3137065.3137085

[13] F. Hasson, S. Keeney, and H. McKenna, "Research guidelines for the delphi survey technique," *Journal of Advanced Nursing*, vol. 32, no. 4, pp. 1008–1015, 2000.

[14] C. Okoli and S. D. Pawlowski, "The delphi method as a research tool: an example, design considerations and applications," *Information & management*, vol. 42, no. 1, pp. 15–29, 2004.

[15] S. Barradell, "The identification of threshold concepts: A review of theoretical complexities and methodological challenges," *Higher Education*, vol. 65, no. 2, pp. 265–276, 2013.

[16] M. Kallia and S. Sentance, "Threshold concepts, conceptions and skills: Teachers' experiences with students' engagement in functions," *Journal of Computer Assisted Learning*, vol. 37, no. 2, pp. 411–428, 2021. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85091417374

[17] P. Alston, D. Walsh, and G. Westhead, "Uncovering "threshold concepts" in web development: An instructor perspective," *ACM Trans. Comput. Educ.*, vol. 15, no. 1, mar 2015. [Online]. Available: https://doi.org/10.1145/2700513

[18] P. Davies and J. Mangan, "Threshold concepts and the integration of understanding in economics," *Studies in Higher Education*, vol. 32, no. 6, pp. 711–726, 2007. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-35748961411

[19] H. A. Linstone, M. Turoff *et al.*, *The delphi method*. Addison-Wesley Reading, MA, 1975.

[20] J. Brown, "Interviews, focus groups, and delphi techniques," in *Advanced Research Methods for Applied Psychology*. Routledge, 2018, pp. 95–106.

[21] H. Cervantes and R. Kazman, *Designing software architectures: a practical approach*. Addison-Wesley Professional, 2016.

[22] E. L. Ouh and Y. Irawan, "Applying case-based learning for a postgraduate software architecture course," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 2019, pp. 457–463.

[23] P. Lago, J. F. Cai, R. C. de Boer, P. Kruchten, and R. Verdecchia, "Decidarch: Playing cards as software architects," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.