2025, Vol. 8, No. 1. ISSN: 2463-3453

ATLAANZ Journal ASSOCIATION OF TERTIARY LEARNING ADVISORS OF AOTEAROA NEW ZEALAND

Embedding academic literacies in software architecture courses using threshold concepts and skills

Vanessa van der Ham

Te Mātāpuna Library and Learning Services, Auckland University of Technology, Aotearoa/New Zealand

Email: vanessa.vanderham@aut.ac.nz

Andre Breedt

Faculty of Medicine and Health, University of Sydney, Australia

Email: andrejohanbreedt@sydney.edu.au

Jing Ma

School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Aotearoa/New Zealand

Email: jing.ma@aut.ac.nz

Abstract

The use of threshold concepts and skills within tertiary education courses holds promise for helping students develop disciplinary competencies and capabilities. This paper describes a collaborative partnership at Auckland University of Technology to develop learning materials and teaching strategies to support students in designing and documenting a blueprint for a software solution in a software architecture paper. Using specific threshold concepts and skills relating to architectural drivers and documentation of views, a face-to-face workshop incorporating related academic literacies was delivered. This initial exploration provides a catalyst for further study that intends to gather faculty and student perceptions of the impact and support that this targeted intervention provides.

Keywords: software architecture, threshold concepts, threshold skills, academic literacies

van der Ham, V., Breedt, A., & Ma, J. (2025). Embedding academic literacies in software architecture courses using threshold concepts and skills. *ATLAANZ Journal 8*(1), Article 5. https://doi.org/10.26473/ATLAANZ.2025.1/005

Academic literacy refers to the range of abilities a student needs to acquire to "communicate competently in an academic discourse community" (Wingate, 2018, p. 350), and encompasses discipline-specific ways of thinking, engaging with knowledge, articulating understanding, and generating new knowledge (Jacobs, 2007). Students' development of subject-specific competencies is by no means straightforward and can take time to develop. Meyer and Land's influential notion of *threshold concepts* (Meyer & Land, 2003, 2005) posits that there are specific concepts within many disciplines which have the power to open new avenues of thinking and reveal subtle interconnections that allow for knowledge integration. They emphasize that a parallel process (or also a catalyst) for a student's changed thinking is the broadening of their use of language: "Threshold concepts lead not only to transformed thought but to a transfiguration of identity and adoption of an extended discourse" (Meyer & Land, 2005, p. 375).

If the deepening of conceptual understanding and the elaboration of language use are so intertwined within the learning of a specific subject, it follows that our teaching practices need to demonstrate a similar integration of course content and language use (Harran & Theunissen, 2019). Subject specialists, as insiders in the discourse community, are well placed to help students acquire these complex sets of capabilities, and there is growing recognition of the value of collaborative partnerships with language and learning specialists. Tertiary learning advisors (TLAs) are increasingly involved in helping make explicit the conventions and practices of academic disciplines (Macnaught et al., 2022; Maldoni & Lear, 2016; McWilliams & Allan, 2014).

One form of collaboration between subject specialists and TLAs is the embedding of academic literacies into specific assessments in courses. As represented in Figure 1, at Auckland University of Technology (AUT), this often involves TLAs working with course lecturers and their course materials to understand what the specific achievement of these outcomes might look like in assignment exemplars, and the support the student cohort might need in terms of achieving the outcomes.

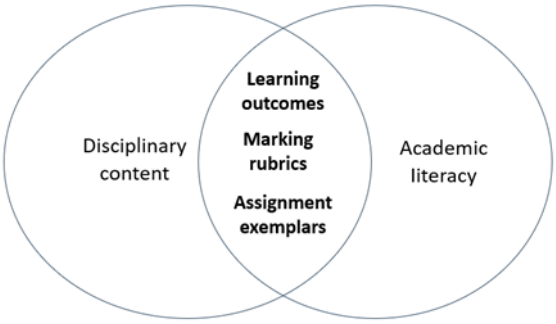

Figure 1. Embedding academic literacies in specific assessments

Where learning outcomes reflect identified threshold concepts within the discipline, this provides rich opportunities for TLAs to scaffold the student journey across these thresholds. As outsiders to the discipline, TLAs are uniquely situated in that their analysis of the disciplinary discourse and development of learning materials can coincide with navigating, to a significant extent, the content thresholds students are required to cross.

This paper describes a collaborative partnership between lecturers in the School of Engineering, Computer and Mathematical Sciences, and TLAs at AUT. The focus is on the development of learning materials and teaching strategies in a software architecture course to help students navigate key threshold concepts in designing and documenting a blueprint for a software solution. The paper begins with a review of the literature on threshold concepts in Engineering education, moving to a focus on their use in software architecture courses. We then explain the pedagogic practices involved in embedding specific threshold concepts in the course at AUT. This is followed by an evaluation of our practices in terms of their alignment with Meyer and Land's (2006) three core guidelines for helping students overcome thresholds, and feedback on the intervention from the course lecturer.

Threshold Concepts in Engineering

In the last two decades, there has been growing interest in the development of the skills and competencies of Engineering students and reported cases of collaboration with academic literacy specialists in several disciplines, such as Mechanical Engineering (Harran & Theunissen, 2019), Engineering Biology and Chemistry (Mendieta et al., 2019),

Environmental Engineering (Wilkes et al., 2015), and Electrical Engineering (Skinner & Mort, 2009).

Supporting students in their efforts to think like engineers requires educational interventions that address concepts and capabilities they find challenging to master – potentially leading to a new way of thinking about their future career roles (Male & Bennett, 2015). As proposed by Meyer and Land (2003), threshold concepts are: *transformative* (they instigate a shift in perspective); *troublesome* (they can pose difficulties for students to grasp); *irreversible* (once internalized, they cannot easily be forgotten); *integrative* (concepts are inter-related in ways not apparent to the learner previously); and *bounded* (they are bordering gateways to other potentially unexplored concepts).

Two characteristics of threshold concepts that have received particular attention in engineering education initiatives are *troublesome* and *transformative.* Initiatives to address these threshold concepts have ranged from interventions aiming to retain students in an introductory electronics course (Harlow et al., 2011) all the way through to large scale projects involving the development of a completely new engineering curriculum (Male, 2012a; 2012b). A particular discipline-specific concept is potentially *troublesome* because it can be "ritualized, inert, conceptually difficult, alien or tacit, because it requires unfamiliar discourse, or because learners do not wish to change their customary way of seeing things" (Harlow & Peter, 2014, p. 8). For students first encountering a difficult threshold concept, a significant period of time can be spent in a transitional phase or *liminal* state (Meyer & Land, 2003) where they need to grapple with the concept – before they find a pathway through the barriers it poses (Baillie et al., 2006). The deeper understanding possibly attained through this process can be said to be *transformative* as it can lead to seeing the subject through *new eyes*.

For educators, identifying threshold concepts via dialogic interaction with their learners has been shown to lead to the shaping of teaching practices and assessments that are more geared towards what is conceptually troublesome, resulting in heightened learner awareness of cognitive strategies and improved problem-solving skills (Knight et al., 2013; Peter et al., 2014). To date, the innovative efforts in engineering education to incorporate threshold concepts point to rich opportunities across disciplines and programmes of study to adopt similar approaches.

Threshold Concepts and Skills in Software Architecture Courses

Software architecture can be described as the first step in designing software solutions for problems, and it involves high level decision-making in the design and organisation of constituent components for software systems. Courses in software architecture have become core to the software engineering curriculum (Nasir & Laiq, 2022; Wei & Zhao, 2023), offering skills and capabilities reflecting the demands of providing realistic architectural design for complex real-world problems. These include analysing the requirements of multiple stakeholders; making design and technical decisions to balance these requirements; and documenting the architecture in multiple views to address stakeholder concerns (Lago & Van Vliet, 2005).

The levels of complexity inherent in designing real-world solutions means that software architecture is generally considered to be a demanding course (Nasir & Laiq, 2022). Thomas et al. (2017) point out that students studying computing engage in problem solving from the outset of their degrees, so they are familiar with the concept of software design – breaking down a problem into its constituent components and describing how the components work together. However, designing the components and how they connect in a software system to communicate a solution to a large and complex problem real-world problem requires bridging the gap "between the world of a variety of, often non-technical stakeholders, on one hand – the problem space, and the technical world of software developers and designers on the other hand – the solution space" (Lago & Van Vliet, 2005, p. 2). In software architecture courses, this involves acquiring key theoretical concepts in the discipline, and arguably more importantly, skills in applying these concepts to designing solutions. Sanders et al. (2012) and Thomas et al. (2012; 2017) propose that in addition to threshold concepts in computing, there are related transformative skills which are learned and improved over time by practice and can be difficult to acquire; these should be considered as thresholds as well.

Although threshold concepts and skills have been identified in computer programming (e.g. Boustedt et al., 2007; Eckerdal et al., 2006; Kallia & Sentence, 2021; Sanders et al., 2012) and software design (Thomas et al., 2017), it is only recently that focus has turned to software architecture as a sub-field of software design in an effort to identify and prioritise threshold concepts and skills in the curriculum and improve pedagogical approaches. Nasir and Laiq (2022) elicited the views of instructors with teaching experience in university-level software architecture courses, and identified eleven threshold concepts and

nine threshold skills, with consensus among the participants that their students struggle to apply the theoretical concepts in designing solutions. Of particular interest to our project are the concepts and skills related to: prioritising architectural drivers for design; specifying quality architecture attributes; associating architectural drivers with the views (models representing the system from different stakeholder perspectives); evaluating a given architecture; and, understanding the outcomes of architecture evaluation.

Collaboration in a Software Architecture Course

Students enrolled on the software architecture course at AUT examine in detail the "designing, documenting and assessing of software architecture" (Auckland University of Technology, 2022). The primary assessment for this course involves individual students designing software for a novel, large-scale system. The lecturer-TLA collaboration focused on developing learning material that provided formative support for this assessment. This would be delivered in the form of a face-to-face, two-hour workshop in the sixth week of the semester.

Context

The software architecture course runs over a semester (12 weeks) and in its latest iteration, a total of 80 students were enrolled. Although tied in with a number of programmes of study, the principal associated programme is that of the Master of Computer and Information Sciences. As indicated by the lecturer, students from this programme tend to have little work experience and can encounter a number of challenges in terms of the realities of software architecture design practices. On the other hand, another significant proportion of students doing the course come from the Master of Information Technology Project Management programme and generally enter the course with a wealth of industry experience. A picture clearly emerges of a student cohort that exhibits varying degrees of understanding and comfort levels regarding large-scale software architecture design. As the aim of the majority of students would be to find future employment in various roles within the IT sector, a primary objective is to provide learners with a solid and comprehensive grounding to ensure industry-readiness.

The Week 6 workshop, which was a TLA-led session with support from the lecturer, was the culmination of a series of formative assessment and teaching activities in support of the software architecture assessment. From Week 2 to Week 5, the lecturer had created in-class discussions around facets of the design assessment with additional online forum-based dialogue where questions could be posed. Weekly course activity and content also covered key areas relevant to the assessment: Week 2 (project proposals, with peer/lecturer feedback on choices); Week 3 (Stakeholder concerns); Week 4 (Quality attribute scenarios), Week 5 (Views). The Week 6 workshop focusing on academic literacies was timed to take place before the submission in Week 7 of a draft of the assessment. This submission was for the purposes of receiving formative feedback from the lecturer only. The final submission of the assessment for grading was in Week 8.

This sequence of formative assessment and teaching activity illustrates a commitment to sustainable assessment (Boud, 2000; Boud & Soler, 2016). Over the first half of the course, students have regular opportunities to: receive peer and lecturer feedback during different phases of their software architecture design process; identify and analyse expectations for a high-quality assessment; and engage in self-assessment to apply feedback received (e.g., through the ungraded draft submission). Our co-teaching initiative involved extending this sustainable assessment practice by:

- providing opportunities for students to revisit key theoretical concepts in software architecture design to check their understanding and correct misconceptions,
- deepening student awareness of how the interdependent constituent components of the architecture function together in a fully documented case, and
- making explicit the academic writing conventions associated with high-quality documentation of the architecture.

Case-based Approach

We used an anonymized high scoring student case report available online in Canvas for the students as an exemplar. The sample case used in the intervention documented the architecture for a virtual reality grocery shopping experience. The case was selected on the basis that it was accessible for the students, and by extension provided a useful contextual vehicle for engaging with key threshold concepts and skills. Case-based learning activities can help students to understand threshold concepts by seeing them applied; however, the

cases used in activities should be understandable and relatable for the students (Nasir & Laiq, 2022; Wei & Zhao, 2023). Participants in the study by Nasir and Laiq (2022) shared that their students often struggled with case examples when they were systems they had never encountered before. For the intervention, it was also seen as important to use authentic student writing to model the application of concepts because text produced by academics or professionals in a field can intimidate students by setting unachievable standards in the learning process (Wingate, 2012).

Co-construction and Modelling of Threshold Concepts

A key threshold concept in designing software solutions is analysing the requirements of multiple stakeholders – people who will be using the software, those who have to manage the design process, those who need to market the system, and those charged with maintaining it. A second closely linked threshold concept is making design and technical decisions for solutions which balance the requirements of the stakeholders. Documenting these solutions in multiple ways to address all stakeholder concerns constitutes a third threshold concept that students need to master. These concepts had been identified as key thresholds for software architecture students at AUT and were embedded in the course via the curriculum as well as the marking criteria. As a result, they underpinned the literacy support we designed.

The key overarching idea informing our intervention was that students needed to *explain* and *justify* their design choices to multiple stakeholders. This entailed a repeated movement between *description* (assembling the details of the design) and *analysis* (bringing the details together to present arguments). This became the organizing principle around which the teaching session and its associated activities were designed.

Co-constructing Ideas around Stakeholders and their Needs

The lead-in activity for the teaching session presented the scenario of a virtual reality grocery store in the form of a visual image accompanied by six question prompts (Who? What? Where? When? How? Why?). Students used these questions to brainstorm and share the possible software design considerations involved – first in groups and then followed by a whole class discussion. Given the diverse backgrounds of the student cohort, we could not assume that they were familiar with the central role of stakeholders in the product design

process. The realistic and relatable scenario at a very fundamental level served as a stepping stone for sensitising students to the importance of rationalising design choices to address the needs of multiple stakeholders. Furthermore, it provided a scaffold for introducing the relationship between description and analysis central to documenting a persuasive software solution (see Figure 2).

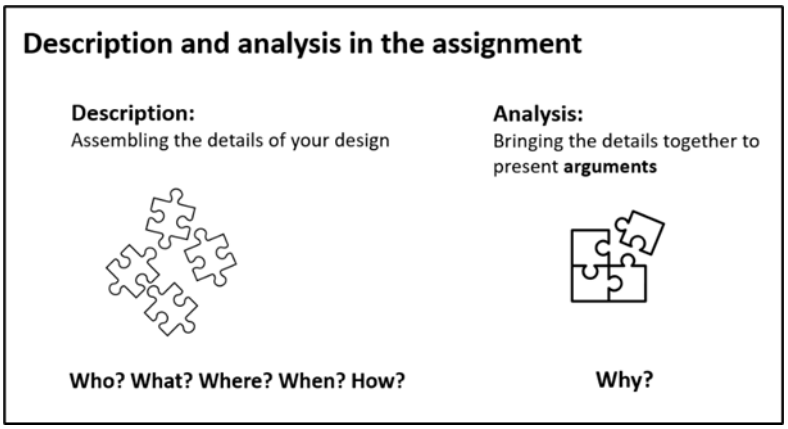

Figure 2. Description/Analysis (Workshop slide)

Activating Prior Learning to Balance Stakeholder Requirements

Having established the importance of addressing stakeholder concerns, the discussion turned to establishing the need for a novel software design. This was initiated through pair-based discussion of a set of question prompts (see Figure 3), which targeted key elements present within the virtual grocery store documentation relating to the software solution's purpose.

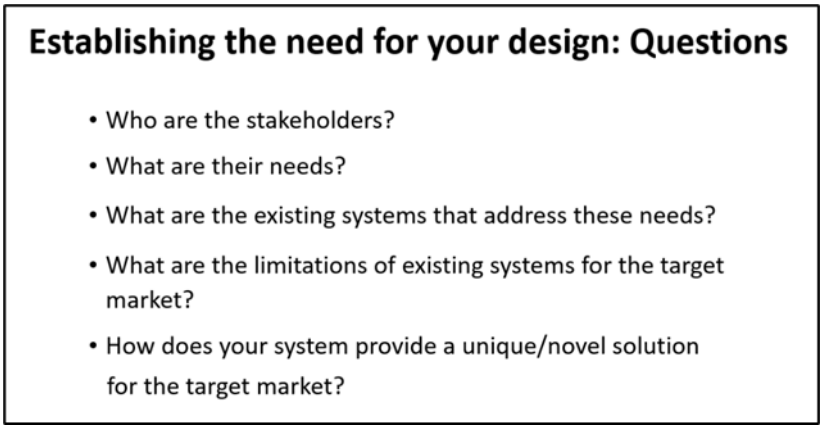

Figure 3: Question prompts (Workshop slides)

There were two learning objectives with these prompts. Firstly, they provided students with an opportunity to activate prior learning related to course material covered in preceding weeks. Secondly, they prepared students for the task of applying their prior learning to a specific context and subsequently documenting their software solution.

Ideas that emerged during the pair-work task were then elicited during a whole class discussion. This was followed by student groups analyzing a sample extract from the virtual grocery store documentation to see how the authors had addressed the question prompts to establish the need for their design. Attention could then be drawn to functional language features that assisted the virtual grocery store designers in providing a persuasive rationale for the value and novelty of their software design. The facilitator then briefly demonstrated how the discussion and sample text related to Week 3 course material – material that explained how software architectures were influenced by multiple stakeholders. Figure 4 shows the pedagogic sequence followed to engage students in navigating concepts for the course:

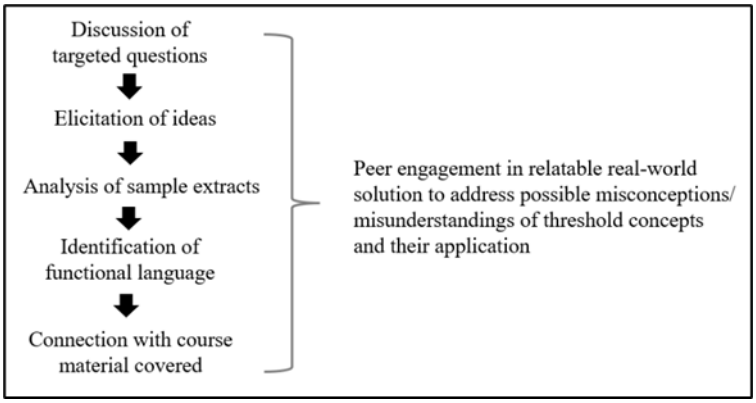

Figure 4: Engaging students in navigating threshold concepts

This sequence was repeated for the other key parts of the software documentation highlighted in the workshop, namely: architectural drivers, quality attribute scenarios and their relationship to both stakeholder concerns and the consideration of multiple views for each scenario developed.

The emphasis on activating prior learning in the pedagogic sequence served as a powerful diagnostic tool for the lecturer to see in real time which software architecture concepts proved most challenging and troublesome for their cohort. Just as importantly, it supported a student-centered approach, which gave learners opportunities to engage with the content and with each other – creating a safe learning dynamic within which to wrestle practically with the primary threshold concepts embedded within the course material and marking criteria.

Bringing the Details Together to Address Stakeholder Concerns

Applying the pedagogic sequence mapped out in the previous section to multiple parts of the software documentation for the virtual grocery store case held a number of pedagogic advantages. Firstly, it allowed for a gradual unpacking of the key elements of a considerably complex software solution document in digestible chunks. Each sequence provided a chance for students to anticipate the possible relationship between description and analysis within a particular section and discuss their ideas. They then used their prior learning in service of analyzing extracts from the virtual grocery store case to identify how the authors had answered the questions they had discussed. This careful scaffolding allowed for successful

priming of learners to be able to recognize functional language used to document a software solution. This facilitated a manageable process of unpacking the details required for key areas of documentation and understanding how they are sequenced and brought together by writers.

Taken together, the sets of question prompts used to introduce each of the document sections formed a comprehensive checklist for students to use when documenting their own software solutions. Furthermore, associated exposure to the language of documenting software solutions embedded in the questions asked, and the exemplar texts combined to sensitize students to the key threshold concepts that needed to be wrestled with in what is widely identified as a challenging course.

Discussion of Intervention

One of the challenges of the use of threshold theory in curriculum development is that threshold crossing by students is very difficult to isolate and evidence (Nicola- Richmond et al., 2018). While it is beyond the scope of this paper to engage with assessment of the impact of our intervention, this will certainly be a consideration in future iterations. For the current initial foray into academic literacies and threshold skills, we base our evaluation of the workshop on guidelines specified by Meyer and Land (2006) for helping students overcome knowledge thresholds. The first is that they need to engage with the knowledge, "represent it in new ways and connect it to their lives" (Olaniyi, 2020 p. 1). Secondly, students need peer assessment to help them cross the liminal space into deeper understanding. Finally, students need recursive exposure to the knowledge and the opportunity to revisit content at their own pace.

The relatable and easy-to-understand case of a virtual reality store, accompanied by activities eliciting students' existing knowledge, resulted in high levels of engagement by the students. In the discussion components that prefaced each treatment of a key design component, students were active participants in co-constructing knowledge. Second, the students were discussing the questions with their peers, helping them to identify possible shared understandings or misconceptions in a safe environment as they crossed the threshold space. Thirdly, the activities in the workshop provided them with a different or recursive take on the concepts covered in the course modules – they were revisiting them from the perspective of a scaffolded application in the sustained case example. Finally, with the

provision of the workshop notes and related exemplar on Canvas, they could continue the learning journey in their own time.

Feedback from the Course Lead highlights the importance of meeting the students where they are in their journeys, and engaging them in disciplinary discourse:

> ... the workshop's success lies in its ability to meet students at their level, regardless of their prior experience. By having learning advisors, who were new to software architecture, guide the session, students were encouraged to start from scratch, holding a fresh perspective on complex concepts. The advisors' focus on teaching students how to think critically, identify key concepts, and articulate their ideas through the effective use of discipline-specific language was transformative. This approach not only clarified the threshold concepts and skills, such as prioritizing architectural drivers and documenting views but also empowered students to approach their assessment with greater confidence and clarity, bridging the gap between theoretical understanding and practical application.
>
> In terms of student satisfaction, in the year we implemented the intervention, the course achieved a score of 4.85/5 in the Student Paper Experience Questionnaire (SPEQ), placing it in the top 5% of over 500 courses within the Faculty of Design and Creative Technologies. One student commented specifically on the materials: "The special learning materials helped me prepare for my assignment. They taught me everything from scratch, even though I am not an expert in system design."

Conclusion

Our intervention in a software architecture course provides an example of engaging learners in collaborative activities using accessible, real-world contexts. This engagement, coupled with formative feedback and recursive exposure to learning materials can help students navigate threshold concepts in a discipline (Meyer & Land, 2006).

Our intervention has been a small-scale exploration into the potential value of using concept thresholds and skills in combination with academic literacy support in a single course. At a programme and curriculum level, identifying a core group of threshold concepts on which to focus learners' attention could assist them in uncovering "connections within the discipline that transcend individual course boundaries" (Boustedt et al., 2007, p. 507).

Collaborative partnerships between subject specialists and learning specialists across multiple courses within a specific discipline hold great promise here.

Acknowledgments

The authors would like to thank Associate Professor Roopak Sinha for spearheading this initiative and for his continued support throughout.

References

Auckland University of Technology. (2022). *Course Descriptor.* https://paperdescriptorreport.aut.ac.nz/PaperDescriptor/PaperDescriptor?courseCode=COMP806&date=2022-03-01&saveFormat=pdf

Bailie, C., Goodhew, P., & Skryabina, E. (2006). Threshold concepts in Engineering education. Exploring potential blocks in student understanding. *International Journal of Engineering Education*, *22*(5). 955–962. http://www.ijee.ie/articles/Vol22-5/06_ijee1823.pdf

Boud, D. (2000). Sustainable assessment: Rethinking assessment for the learning society. *Studies in Continuing Education, 22*, (2), 151–167. https://doi.org/10.1080/713695728

Boud, D., & Soler, R. (2016). Sustainable assessment revisited. *Assessment and Evaluation in Higher Education*, *41*(3), 400–403. https://doi.org/10.1080/02602938.2015.1018133

Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K, & Zander, C. (2007). Threshold concepts in computer science: Do they exist and are they useful? *ACM SIGCSE Bulletin, 39*(1) 504–508. https://doi.org/10.1145/1227504.1227482

Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., Zander, C. (2006). Putting threshold concepts into context in computer science education. In *Proceedings of the 11th Annual SIGCSE Conference: Innovation & Technology in Computer Science Education* (pp. 103–107). http://dx.doi.org/10.1145/1140123.1140154

Harlow, A., & Peter, M. (2014). Mastering threshold concepts in tertiary education: "I know exactly what you are saying and I can understand it but I've got nowhere to hook it." *Waikato Journal of Education Te Hautaka Mātauranga o Waikato*, *19*(2), 7–23. https://doi.org/10.15663/wje.v19i2.95

Harlow, A., Scott, J. Peter, M., & Cowie, B. (2011). 'Getting stuck' in analogue electronics: Threshold concepts as an explanatory model. *European Journal for Engineering Education, 36*(5), 435–447. https://doi.org/10.1080/03043797.2011.606500

Harran, M., & Theunissen, H. W. (2019). Navigating the engineering literacy divide: Design report collaboration practice realities. *Journal of Engineering, Design and Technology*, *17*(1), 77–101. https://doi.org/10.1108/JEDT-07-2018-0112

Jacobs, C. (2007). Towards a critical understanding of the teaching of discipline-specific academic literacies: Making the tacit explicit. *Journal of Education, 41*, 59–81, https://hdl.handle.net/10520/AJA0259479X_23

Kallia, M., & Sentence, S. (2021). Threshold concepts, conceptions and skills: Teachers' experiences with students' engagement in functions. *Journal of Computer Assisted Learning, 37*, 411–428. https://doi.org/10.1111/jcal.12498

Knight, D., Callaghan, D., Baldock, T., & Meyer, J. F. H. (2013). Identifying threshold concepts: Case study of an open catchment hydraulics course. *European Journal of Engineering Education*, *39*(2). 125–142. http://dx.doi.org/10.1080/03043797.2013.833175

Lago, P., & Van Vliet, H. (2005). Teaching a course on software architecture. In *Proceedings of the 18th Conference on Software Engineering Education & Training (CSEET'05)* (pp. 35–42). https://doi.org/10.1109/CSEET.2005.33

Macnaught, L., Bassett, M., van der Ham, V., Milne, J., & Jenkins, C. (2022). Sustainable academic literacy development: The gradual handover of literacy teaching. *Teaching in Higher Education, 27*(2)*,* 1–19. https://doi.org/10.1080/13562517.2022.2048369

Maldoni, A. M., & Lear, E. I. (2016). A decade of embedding: Where are we now? *Journal of University Teaching & Learning Practice, 13*(3), 2–20. https://doi.org/10.53761/1.13.3.2

Male, S. A. (2012a). *Engineering thresholds: An approach to curriculum renewal final report.* Australian Government Office for Learning and Teaching. https://ltr.edu.au/resources/PP10_1607_Baillie_Report_2012.pdf

Male, S. A. (2012b). *Integrated Engineering foundation threshold concept inventory*. Australian Government Office for Learning and Teaching, Sydney. https://ltr.edu.au/resources/PP10_1607_Baillie_Threshold_concept_2012.pdf

Male, S. A., & Bennett, D. (2015). Threshold concepts in undergraduate engineering: Exploring engineering roles and value of learning. *Australasian Journal of Engineering Education*, *20*(1), 59-69. https://doi.org/10.7158/D14-006.2015.20.1

McWilliams, R., & Allan, Q. (2014). Embedding academic literacy skills: Towards a best practice model. *Journal of University Teaching and Learning Practice, 11*(3). https://doi.org/10.53761/1.11.3.8

Mendieta, J., Chidlow, R., Han, D., Bingham, T., Lin, S., Hardley, L., & Lin, L. (2019). *Integrating academic literacy support into the curriculum.* University of Auckland, Library and Learning Services. https://doi.org/10.17608/k6.auckland.16831210.v2

Meyer. J. H. F., & Land, R. (2003). Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practicing within the disciplines. In C. Rust (Ed.), *Improving student learning: Improving student learning theory and practice – ten years on* (pp. 312–424). Oxford Centre for Staff and Learning Development.

Meyer, J. H. F., & Land, R. (2005). Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning, *Higher Education*, *49*(3), pp. 373–388. https://doi.org/10.1007/s10734-004-6779-5

Meyer, J. H. F., & Land, R. (2006). *Overcoming barriers to student understanding: Threshold concepts and troublesome knowledge*. Routledge.

Nasir, U., & Laiq, M. (2022). Threshold concepts and skills in Software Architecture: Instructor's perspectives. In *Proceedings of the 29th Asia-Pacific Software Engineering Conference. (ASPEC)* (pp. 547–553). http://doi.org/10.1109/APSEC57359.2022.00076

Nicola-Richmond, K., Geneviève, P., Larkin, H., & Taylor, C. (2018). Threshold concepts in higher education: A synthesis of the literature relating to measurement of threshold crossing. *Higher Education Research and Development*, *37*(1), 101–114. https://doi.org/10.1080/07294360.2017.1339181

Olaniyi, N. E. E. (2020). Threshold concepts: Designing a format for the flipped classroom as an active technique for crossing the threshold. *Research and Practice in Technology Enhanced Learning*, *15*(2) 1−15. https://doi.org/10.1186/s41039-020-0122-3

Peter, M., Harlow, A., Scott, J. B., McKie, D. Johnson, E. M., Moffat, K., & McKim, A. M. (2014). *Threshold concepts: Impacts on teaching and learning at tertiary level.* University of Waikato, Commission Report for Teaching and Learning Research Initiative. https://researchcommons.waikato.ac.nz/handle/10289/9058

Sanders, K., Bostedt, J., Eckerdal, A. McCartney, R., Moström, J. E., Thomas, L., & Zander, C. (2012) Threshold concepts and threshold skills in computing. In *Proceedings of Nineth Annual International Conference on International Computing Education Research* (pp. 23–30). https://doi.org/10.1145/2361276.2361283

Skinner, I., & Mort, P. (2009), Embedding academic literacy support within the Electrical Engineering curriculum: A case study. *IEEE Transactions on Education*, *52*(4), 547–555. http://doi.org/10.1109/TE.2008.930795

Thomas, L., Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Sanders, K., & Zander, C. (2012). A broader threshold: Including skills as well as concepts in computing education. In *Threshold concepts: From personal practice to communities of practice. Proceedings of the National Academy's Sixth Annual Conference and Fourth Biennial Threshold Concepts Conference* (pp. 27–29). Trinity College.

Thomas, L., Boustedt, J., Eckerdal, A., McCartney, A., Moström, J. E., Sanders, K., & Zander, C. (2017). In the liminal space: Software design as a threshold skill. *Practice and Evidence of the Scholarship of Teaching and Learning in Higher Education, 12*(2), 331–351. https://doi.org/10.1080/00048623.2015.1062260

Wei, L., & Zhao, Q. (2023). Teaching reform and practice of software architecture for postgraduates in universities. In L. F. Fing et al. (Eds.), *Proceedings of the 2nd International Conference on Education, Language and Art (ICELA, 2022)* (pp. 468–473). https://doi.org/10.2991/978-2-38476-004-6_58

Wilkes, J., Godwin, J., & Gurney, L. J. (2015). Developing information literacy and academic writing skills through the collaborative design of an assessment task for first year engineering students. *Australian Academic and Research Libraries*, *46*(3), 164–175. https://doi.org/10.1080/00048623.2015.1062260

Wingate, U. (2012). Using academic literacies and genre-based models for academic writing instruction: A literacy journey. *Journal of English for Academic Purposes*, *11*, 26–37. https://doi.org/10.1016/j.jeap.2011.11.006

Wingate, U. (2018). Academic literacy across the curriculum: Towards a collaborative instructional approach. *Language Teaching*, *51*(3), 349–364. http://dx.doi.org/10.1017/S0261444816000264