Did our Course Design on Software Architecture meet our Student's Learning Expectations?

Eng Lieh Ouh
School of Information Systems
Singapore Management University
Singapore
elouh@smu.edu.sg

Benjamin Kok Siew Gan
School of Information Systems
Singapore Management University
Singapore
benjamingan@smu.edu.sg

Yunghans Irawan
Singapore Product Development Center
Indeed Singapore
Singapore
yirawan@indeed.com

***Abstract*— This Innovative Practice Full Paper discusses our course design on software architecture to meet the learning expectations of two groups of software engineers. Software engineers with working experiences frequently find themselves the need to upskill in their lifelong learning journey. Their learning expectations are shaped not just by their need to know but also other learning characteristics such as their working experiences. In many cases, we design courses based on the required learning outcomes and assessment criteria. In this paper, we wish to find out whether our course design on software architecture has met the learning expectations of our students over eight years. Our study data involves two groups of software engineers to upskill in two courses (1) software engineering practitioners taking a public software architecture design course (2) postgraduate students taking a Master's level software engineering programme. We explain how we evolve the course design based on their students' feedback after each run of the courses. Although the feedback of each run show encouraging results, we discover gaps in our design when we cumulatively analyze the trends of their learning expectations with their qualitative comments over the years. Our design is able to consistently meet some but not all the learning expectations that recur over the years and we discuss the reasons for this outcome and potential further interventions. We hope this discussion and trend analysis process can help course designers to improve their course design.**

Keywords*— *software engineering, learning expectations, software architecture, adult learner

I. Introduction

Software engineers constantly upskill to stay relevant and up to date with technology advancement. Many of them choose to attend courses, especially when the course is accredited by the organization or subsidized as part of the national initiative to promote lifelong learning. Software engineers who have working experiences and commitments differ in their learning expectations as compared to one without. This group of learners can better appreciate the constraints of software designs and expect to learn skills for them to apply at work. On the other hand, an undergraduate with little or no working experiences and commitments potentially choose to focus on the assessments to assess their level of competencies. We need to align our course design to their expectations to better address the learners' needs.

To understand the learning expectations of our students, we initially conduct surveys to gather their expectations for a course on software architecture. However, without going through the course in many cases, we discovered only high-level expectations of the course could be found. Typical comments on their expectations include "able to apply at work", "able to design the architecture" and "able to evaluate vendor's proposed architecture". These comments can guide to set the learning objectives but not the detailed design and delivery of the training. On the other hand, our course also needs to fulfil the required learning outcomes and assessment needs. In our case, participants can be subsidised by a nation-wide initiative for lifelong learning if they satisfy the learning outcomes based on the assessment criteria.

Along with this initiative, there is an accredited skills framework for software engineering designed by relevant industry experts and can be used to guide the course design of software architecture training. We first adopt a combination of both survey and the accredited framework to design our course and evolve our design with their feedback after each run. In this paper, we evaluate the effectiveness of our course design and it's evolution over the years to meet the learning expectations of our software engineers. We seek to address the following research question for the rest of the paper.

RQ - "Did our course design on software architecture meet the learning expectations of our students?"

We implement the course design in two courses catering to two groups of software engineers – industry practitioners taking our public course on software architecture and postgraduate students taking our software architecture module as part of their software engineering master' programme. We termed the former as public participants, the latter as postgraduate students and collectively termed both of them as learners or students for the rest of the paper. To address our RQ, we quantitatively analyse to evaluate the effectiveness of our course design and qualitatively explain interventions applied after each run. To further identify the trends in their learning expectations over the years, we evaluate their course ratings and gather over 1,000 of their feedback comments over eight years, cluster and classify them for discussion. A key contribution of this paper is the trend analysis process to evaluate the learning expectations based on the frequency of occurrences over a period. We hope this discussion and trend analysis process provides a basis for course designers to improve their course design in related software engineering courses for software engineers.

The rest of the paper is organized as follows: We first present background and related work in Section II. We give an overview of our course design in Section III and explain the course delivery in Section IV. We discuss the quantitative feedback ratings and analyze the qualitative feedback comments in Section V. We conclude this paper with a discussion on the threats to the validity of our results in Section VI and a conclusion in Section VII.

II. Background and Related Work

A. Characteristics of Adult Learners

Understanding the characteristics of adult learners is crucial to better design the course for the learners. Extensive work by Knowles [1, 2] and other educators in andragogy resulted in the development of new assumptions about adult learners. They characterize adult learners into six areas – (1)

need to know, (2) learner's self-concept, (3) role of experience, (4) readiness to learn, (5) orientation to learning and (6) motivation.

Adults need to know the utility and value of the material that they are learning before embarking on learning. In adult learning, the first task of the teacher was to help the learner become aware of the need to know [3]. When adults undertake to learn something on their own, they invest considerable energy probing into the benefits they will gain from learning it and the negative consequences of not learning it. Self-concept of the adult learner assumes that the adult learner is self-directing and autonomous [4, 5]. Adults have a deep psychological need to be seen by others and treated by others as being capable of self-direction and resent and resist situations in which they feel that others are imposing their wills on them. The third characteristic is how adult learners deal with the role of the learner's prior experience. Adult learning practitioners believe that prior experiences are the richest resources available to adult learners. Adults tend to come into adult education activities with a greater volume and higher quality of experience than younger children. The fourth characteristic is that of readiness to learn. In adults, readiness to learn is dependent on an appreciation of the relevance of the topic. Adult learners tend to become ready to learn things that they believe they need to know or be able to do to cope effectively with real-life situations and problems. In adult learning theory, the view of an adult's orientation to learning is problem-centred, task-centred, or life-centred. Adults are motivated to learn to the extent they perceive that the knowledge will help them perform tasks or solve problems that they may face in real life. Thus, adults learn best when new knowledge, skills, and attitude are presented in the context of real-life situations. The sixth characteristic of adult learning addresses the motivation to learn. All normal adults were motivated to keep learning, growing and developing [3]. While adults are responsive to extrinsic motivation, they are mostly driven by internal pressure, motivation, and the desire for self-esteem and goal attainment.

These characteristics need to be considered during the course design. We emphasize the benefits of learning software architecture and the value of designing the right software architecture at the start of the course. We structure the course to empower the students to be self-directed in their learning by having activities to synthesize the software architecture concepts and derive the possible architecture designs. It also has to provide an opportunity for them to bring and discuss their prior project experiences and relate the concepts and exercises to real-life design situations so that they can apply in their workplace. The learning environment is conducive and adaptable to the student's commitments which can motivate them to learn.

B. Designing a Software Architecture Course

Interests to learn software architecture has evolved tremendously over the years but given its level of abstraction, remains a difficult subject for software engineers to grasp and for educators to teach. The skillset for one to be a competent software architect is multi-faceted, which increase the level of difficulty to be taught in a classroom environment. The role of a software architect typically entails one to have technical, analytical and effective communication skills. Technical skills that minimally include software design and programming experiences. Analytical skills are essential for the software architect to grasp the problem quickly, diagnose the possible root causes and make significant decisions for the project. An architect who is unable to make significant design decisions (principles) where much is unknown, where there is insufficient time to explore all alternatives, and where there is pressure to deliver is unlikely to succeed [6]. The architect should have effective communication skills, including speaking, writing, and presentation abilities to address complex problems with a seemingly simple design that are easy to grasp.

Rupakheti and Chenoweth [7] described their experiences and learnings in designing their software architecture course to undergraduates. Their systematic problem in getting architecture concepts across to undergraduates is similar to our challenge for students, primarily those with limited design experiences. Galster and Angelov [8] describe a framework involving the relationship of concept (software architecture), representation (architecture description), referent (software architecture practice) to the learner element in the learning space. In addition to the vagueness of the concept of software architecture itself, architecture problems are usually "wicked." Asking students to create architecture is different to, for example, asking them to write a Java program - students have a much clearer understanding of what the expected outcome is. Mannisto, Savolainen, and Myllarniemi [9] discuss on the means for teaching students what it takes to face software architecture design problems with some characteristics of wicked problems and providing students with some methodological tooling for coping with the problems in their profession as software architects. The industrial environment differs significantly from typical exercises in software architecture teaching. Constraints often dominate the development process. Ouh and Irawan [10] adopt an experiential risk learning model to design their software architecture course for undergraduates to address the challenges of teaching abstract software architecture concepts to undergraduates. The model comprises of activities to simulate risks that can happen in practical scenarios, and their role is to be able to recognise these risks, reflect on the causes and mitigate these risks.

Our design of the software architecture courses is targeted at adult learners with working experiences who differ in their learning needs and characteristics from undergraduates. We evaluate our course design in meeting our student's learning expectations, and we believe that our understanding of their learning expectations over an extended period of time is more conclusive than on specific runs.

III. DESIGN OF THE SOFTWARE ARCHITECTURE COURSE

In this section, we give an overview of the design decisions made for our software architecture training course. The course design is documented and submitted to the relevant authority of the skills framework committee for validation before the course can be launched.

A. Course Objective

To align with the learner's need to know and readiness to learn, we set our course key objective to equip our students with the industry-relevant skills in designing a software architecture. To achieve this, we review the National Infocomm Competency Framework [11] which comprises of competencies required for a particular role developed based on inputs of industry experts. We adopt these high-level competencies units relevant to an architect role and design our course based on these competencies.

- Explain how the application architecture fits into the broader context of organizational business goals and enterprise architecture.
- Design the architecture with an emphasis on the common application integration components.
- Describe software architecture with views and viewpoints.
- Analysis of software architecture designs with respect to the quality attributes and their trade-offs.

B. *Course Structure*

The course structure can be designed based on two broad categories: (1) for one in the academic community to address research problems (2) for one in the practitioner's community to gain practical skills. The former generally leads one to pursue higher education in the research community while the latter leads to apply these skills at their workplace. In this study, the course structure focus on the latter - professional software engineers to obtain skills to apply back to their workplace. This decision also aligns with the learner characteristics for our group of students in terms of the need to know, readiness, orientation and motivation to learn. Based on the skills framework [11], we describe our software architecture course design in two dimensions - underpinning knowledge and evidence of learning.

The underpinning knowledge dimension describes the knowledge required for the student to achieve the learning objectives. For this dimension, the content is broken down into two parts - architectural thinking and software qualities. By definition, software architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and the environment, and the principles guiding its design and evolution [6]. For the first part on architectural thinking, students get to understand these architecture components and relationships among them. The students are exposed to the architectural styles primarily based on the work by Bass [12] and reference patterns. Examples of architectural styles include client-server, tiered computing data-centric, call-and-return, event-driven, layered and reference patterns include enterprise service bus, service-oriented architecture and cloud computing. These styles and patterns are introduced with an explanation of real-life industry case scenarios [13]. We seek to educate the student on how and when to implement these styles/patterns. For example, an enterprise service bus architecture design provides decoupling between systems but might not be suitable for small enterprises with limited integration needs.

For the second part on software qualities, we introduce the list of qualities in ISO 25010 Product Quality Model [14] and the Rational Unified Process [15, 16]. These are standardized list and process which they can reuse in their workplace. We seek to educate the student on each software quality, the trade-offs of implementing these qualities and how to mitigate them. For example, an implementation for two-factor authentication provides better security but trade-off usability of the system. We can mitigate this trade-off by delaying the authentication step until needed and not always at the start.

The evidence of the learning dimension describes the concrete artefacts and activities that can demonstrate the student's level of competency in the application of the knowledge gained. For this dimension, the students are required to produce design artefacts and we also evaluate their competency through observations and questioning in case-study discussions. These discussions allow sharing of their experiences, peer learning and allow the students to practice their analytical and communication skills. The selection of the case studies is based on industry relevance, and we have to decide on the domain of these case studies due to limited course duration. We decide to focus on information systems due to the high relevance of typical architecture designs most organizations need to have. The selected information systems case studies cover the front-end architecture with web, mobile architectures; backend architecture involving enterprise integration, cloud; and distributed architectures. These case studies are real-life scenarios, allowing the students to analyze, apply the concepts taught and mitigate the potential trade-offs within a realistic context.

IV. Course Delivery

In this section, we describe the course delivery and discuss the quantitative results. We also discuss how we evolve the course design based on qualitative comments.

A. *Course Conduct*

The course duration is structured around a 32.5 hours classroom contact time and can be delivered in 5 consecutive days or spread out over five weeks. This schedule makes it flexible for both the public participants and our postgraduate students. We conduct the course once a week on weekends for five weeks for the postgraduate students and five days in 1 week for public participants. For public participants, a schedule longer than consecutive five days or spread across weeks is undesirable due to work impact. For part-time postgraduates, having the course on weekends reduce their work impact and the need to take leave as the part-time postgraduate programme requires them to commit their time over 2.5 years.

Besides the differences in the delivery schedule of both courses, we seek to maintain a consistent course design for both our group of students. These two courses are taught by the same two instructors with a workload evenly spread. Both instructors have about ten years of industry working experiences when these courses are designed.

B. *Course Runs*

The public course is conducted four times a year with an average of 19 public participants per class. Each run of the class spans over five full days and conducted by two instructors to give the public participants additional exposure to the experiences of different instructors. We have trained 524 public participants from the beginning of 2011 to the end of 2017 over a total of 28 runs of the course. There is no cancellation of the scheduled classes since the beginning.

For the postgraduate course, it is designed as part of a software engineering master's programme where postgraduate students can complete it part-time for 2.5 years or full time in 1 year. The postgraduate software architecture course is conducted once a year with an average of 50 postgraduate students per class over eight years. Each run of the class spans five full days conducted on either on Saturdays or two weekday evenings over a total of 5 weeks. 399 postgraduate students have taken this course from 2010 to 2017. There is no cancellation of the classes due to low enrolment) since the beginning.

C. Student's Profile

The public participants on average, have five years of IT working experiences based on the initial course survey. However, their variation can span from a low of 2 years to over 15 years of software engineering experiences. Their roles also vary from junior software engineer to accredited software engineer.

The postgraduate students on average, have three years of IT working experiences based on their details for postgraduate admission. Their variation is smaller as compared to the public participants, primarily due to the standard admission criteria of the postgraduate programme. Their roles are mostly junior levels in software engineering and development.

D. Course Feedback Collection and Processing

For the public course, each public participant is given a list of feedback questions, and they are required to hand in their answers and comments at the end of the course. The survey questions relevant to our research questions are shown below. The first two metric-based questions are rated on a 5-point Likert scale ranging from 1–Poor, 2-Fair, 3-Good, 4-Very Good and 5–Excellent and the third question is to collect their qualitative feedback comment.

Q1. "What is the overall satisfaction level for this course?"

Q2. "How well does this course impart the knowledge and skills needed for you to apply and practice?"

Q3. "Please provide concrete actionable feedback comments (e.g. application of concepts taught, like best about the course, suggestions to improve the course)"

For the postgraduate course, we track the postgraduate students' feedback by asking them in a survey at the end of the semester to provide a rating on their opinions for the course. The survey questions relevant to our research questions are shown below. The first two metric-based questions are rated on a 5-point Likert scale ranging from 1–Poor, 2-Fair, 3-Good, 4-Very Good and 5–Excellent and the third question is to collect their qualitative feedback comment. These questions are not the same as the public participants as these questions are controlled at the university level.

Q1. "Overall module ratings"

Q2. "Department level average ratings"

Q3. "Please comment on the strengths and weaknesses of the module, and suggest possible improvements."

E. Course Review and Evolution of the Course

After each run of the course, the instructors and another external reviewer have a closedown meeting to discuss the gathered feedback. The response rate of the feedback is high as the participants are given sufficient time to do this. We agree on the needed improvements to evolve the course design and confirm the changes are implemented when we have the start-up meeting for the next run. The evaluation feedback is also scrutinised by both the teaching management and subsiding agency to ensure we are delivering a quality course for our participants to achieve their learning outcomes.

For the qualitative comments of both courses, we first remove empty or nil comments (e.g. "nil") and split extensive comments into smaller paragraphs of similar context. For example, a comment comprising of multiple sentences discussing the course duration and usage of tools can be split into two comments at the sentence level. Using Azure Machine Learning service [17], we invoke the service for each comment and analyze the results. There can be more than one paragraphs per student or none. As a result, the number of comments can be different from the number of students (or class size). The class size per course run, survey answers, and sentiment analysis of the comments for public participants are shown in Table I and Fig. 1 and the results for postgraduate students are shown in Table II and Fig. 2.

TABLE I. COURSE RATINGS-PUBLIC PARTICIPANTS

Run (class size)	Q1	Q2	Q3 Sentiment Ratings (Number of Comments - Positive / Neutral / Negative)
1 (11)	4	4	(26) - **69.23%** / 7.69% / 23.08%
2 (10)	3.7	4	(45) – **75.56%** / 13.33% / 11.11%
3 (17)	3.7	3.6	(30) – **80.00%** / 3.33% / 16.67%
4 (22)	3.7	3.8	(25) – **80.00%** / 16.00% / 4.00%
5 (14)	3.6	3.9	(19) – **68.42%** / 10.53% / 21.05%
6 (15)	3.9	3.5	(25) – **68.00%** / 16.00% / 16.00%
7 (22)	3.9	3.8	(34) – **64.71%** / 17.65% / 17.65%
8 (25)	4	3.7	(28) – **89.29%** / 3.57% / 7.14%
9 (23)	4.2	4	(29) – **82.76%** / 13.79% / 3.45%
10 (16)	4.1	4.1	(27) – **92.59%** / 7.41% / 0.00%
11 (23)	4.3	3.9	(44) – **77.27%** / 11.36% / 11.36%
12 (17)	4.2	4.1	(33) – **73.53%** / 14.71% / 11.76%
13 (15)	3.9	4.1	(29) – **82.76%** / 17.24% / 0.00%
14 (8)	4.1	3.7	(16) – **81.25%** / 18.75% / 0.00%
15 (20)	4.2	4.3	(37) – **81.08%** / 8.11% / 10.81%
16 (22)	4.1	3.9	(46) – **82.61%** / 15.22% / 2.17%
17 (15)	4.4	4.1	(43) – **76.74%** / 9.30% / 13.95%
18 (24)	4	4.2	(63) – **76.19%** / 14.29% / 9.52%
19 (20)	4.3	3.8	(47) – **89.36%** / 2.13% / 8.51%
20 (22)	4.3	4.1	(51) – **82.35%** / 15.69% / 1.96%
21 (24)	4.1	4.1	(63) – **82.54%** / 12.70% / 4.76%
22 (16)	4.5	4	(45) – **77.78%** / 11.11% / 11.11%
23 (18)	4.2	4.4	(25) – **76.00%** / 20.00% / 4.00%
24 (16)	4.3	4.3	(41) – **82.93%** / 7.32% / 9.76%
25 (22)	4.2	4.2	(38) – **78.95%** / 18.42% / 2.63%
26 (20)	4.62	4.1	(24) – **75.00%** / 12.50% / 12.50%
27 (23)	4.42	4.33	(19) – **68.42%** / 21.05% / 10.5%
28 (24)	4.63	4.46	(14) – **92.86%** / 7.14% / 0.00%
Avg (19)	**4.13**	**4.02**	**(35) – 79.19%/ 12.22% / 8.59%**

Fig. 1. Course Rating-Public Participant

TABLE II. COURSE RATINGS-POSTGRADUATE STUDENTS

Run (class size)	Q1	Q2	Q3 Sentiment Ratings (Number of Comments - Positive / Neutral / Negative)
1 (75)	4.25	4.22	(29) – **72.41%** / 3.45% / 24.14%
2 (41)	4.22	4.147	(28) - **71.43%** / 17.86%/ 10.71%
3 (51)	4.26	4.12	(23) - **73.91%**/ 13.04% / 13.04%
4 (44)	3.98	4.11	(34) - **70.59%** / 11.76% / 17.65%
5 (39)	4.27	4.12	(27) – **74.07%** / 7.41% / 18.52%
6 (49)	4.15	4.11	(30) – **66.67%** / 16.67% / 16/67%
7 (38)	4.14	4.03	(15) – **73.33%** / 13.33% / 13.33%
8 (62)	3.90	4.00	(29) – **68.97%** / 10.34% / 20.69%
Avg (50)	**4.15**	**4.10**	**(27) – 72.06%/ 11.27% / 16.67%**

Fig. 2. Course Rating-Postgraduate Students

The quantitative feedback ratings and sentiment analysis from both groups of students are generally positive and above average, giving us a form of encouragement that the learning expectations of these students are met. The average ratings also do not set off any alarms with the teaching management or the subsiding agency over the years. As part of the continuous improvement of our course, we review the qualitative comments after each run, especially on the suggestions to improve the course. Although some of these comments are given by only one or two participants in each run, we view these comments as potential risks to the course in the long run and we put in actions to mitigate these risks.

One mitigation action that we put is the tool used for the exercises and assessments. We initially design to use a commercial tool to document software architectures but many feedback that they might not have the chance to use the tool back at work and it takes time to learn the tool within the limited course duration. We end up giving an alternative option to using pen and paper. We also receive suggestions to enhance our contents of other software methodologies and technologies. We view these suggestions as potential industry needs and plan to include them in future runs. Some areas that we include are microservices design, serverless computing and big data architecture for machine learning.

Another common suggestion is to have more class discussions and team exercises for peer learning. Many participants felt that discussions and team exercises enable them to learn from other peers and apply what they learn. On the other hand, they feel that assessments take up too much of the course time. We realise we need to balance the time for these activities as compared to assessments. We still need to formally assess them as part of their coursework grades or competency assessments to be eligible for subsidies. Over the years, we incorporate many of the team exercises and discussions as part of the assessments and such feedback become less common. However, there is a limit to these adjustments as a certain level of individual quantitative assessments are still required. We end up accepting this risk in subsequent runs. Another example of a risk that we accept is on our focus on designing software architectures for information systems. We infrequently encounter enrolled students who are developing embedded systems and wish the course can be structured around those kinds of systems. In this case, we decide to continue our focus on information systems and ensure our course outline are updated clearly before they enrol for the course. We also make a point to explain this rationale at the start of the course. In subsequent runs, we still see cases of students having background in embedded systems attending the course but most of them understand the course design and express their interests to learn about information systems instead.

V. TREND ANALYSIS OF THE COURSE DESIGN

In the previous section, we evaluate the course design based on the quantitative results of each course run. However, we acknowledge that the quantitative analysis for each run might not be conclusive to address the RQ. In this paper, we continue to analyze the qualitative comments compiled over the years to identify trends and evaluate if we did meet the student's learning expectations.

We consolidate a total of 966 feedback comments from the public participants and 215 feedback comments from the postgraduate students for a total of 1181 feedback comments. We cluster them into ten key themes based on coding by the two instructors. The coding process involves randomly 20% of the total set of comments and repeats if the differences in the themes between the instructors are more than 10%. Below are the ten themes and their descriptions. Table III and Fig. 3-4 summarize these themes.

(1) **Number of case study discussions/demos.** Students felt the need to increase the number of discussions and class demos to better understand the relevance of the concepts taught.

(2) **Number of practice exercises.** Students felt the need for more exercises to apply and practice the concepts taught.

(3) **Level of guidance.** Students require more assistance either in understanding the knowledge or working on the assignments and exercises.

(4) **Depth of Content.** Students felt the need to go in-depth for certain topics.

(5) **Course duration.** Students felt that it is not sufficient to effectively deliver the course topics within 5 days.

(6) **Workshop Duration.** Students felt that the workshop duration is too tight and suggest to reduce the number.

(7) **Experiences Sharing.** Students suggest more sharing of experiences either from the instructors or from other students.

(8) **Tool usage.** Students felt the need to use PC and tools on the workshop assignments.

(9) **Content Coverage.** Students felt the need to increase the breadth of content coverage by reducing the assignments and make the materials more concise

(10) **Theoretical Level.** Students felt the need to reduce the theoretical concepts and more on the practical aspects of a solution.

Each comment is evaluated to be in none, one or more than one theme. For example, "Can introduce more tools and examples on the subject" can be related to two themes – "Number of sample practice exercises" and "Tool Usage". We identified a total of 372 relevant comments, 300 from the public participants and 72 from the postgraduate students. About 69% (809 out of 1181) of the total comments are either positive comments (e.g. "Course content very good") or neutral comments (e.g. "apply methodology learned to design.") and these comments are discarded from this analysis. For each set of comments in each theme, we analysis their occurrences yearly over the years. There are at least five runs of the courses in total each year and involve at least 120 of public participants and postgraduate students yearly.

We group these themes into the following three categories of expectations based on their frequency of occurrences:

a. **Recurring Expectations** - Recurring expectations are comments that occurred at least once within a year after applying interventions.

b. **Sporadic Expectations** - Sporadic expectations are comments that occurred at least once within a two year period after applying interventions.

c. **Managed Expectations** - Managed expectations refer to comments that occurred before but due to interventions ceased to occur in subsequent years.

The expectations for themes (1), (2), (3) and (4) (more case study discussions/class demos, more practice exercises, a higher level of guidance and more content depth) are constantly recurring throughout the years. We grouped these expectations under the type of recurring expectations due to the occurrences within a year after interventions are applied. We did increase the number of discussions and demos over the years, but this type of comments persists throughout the years. It is a challenge to create or find relevant content in software architecture with the right level of detail based on the student's profile. Our students have a large variation in terms of their years of experiences and work domains. This profile variation is generally wider among the public participants as compared to postgraduate students. This is because our postgraduate students are admitted into the programme based on a common set of admission criteria. The expected level of guidance and content depth also varies with the student's profile. While students who have less number of years in software designs require a higher level of guidance with less content depth so that they are not overwhelmed, students with more years of experiences expect more in-depth content and require less level of guidance.

The expectations for the themes (5), (6) and (7) (longer course and workshop duration, more experience sharing) are sporadically recurring throughout the years. We grouped these expectations under the type of sporadic expectations as they infrequently occur over the years. We attribute the recurrence of (5) and (6) to the student's ability to do the assignments within the workshop duration. We encountered course runs when all participants finished their workshop earlier than expected and also otherwise. Students attribute the reason of work commitments as the cause for being unable to complete their assignments on time and ask to extend the deadline. For (7), we do encourage experiences and increase the opportunities for sharing, but it is still subject to the student's openness to share their experiences. On the other hand, it is a design challenge to balance the time between experiences sharing and coverage of content within the limited five days course. We partially mitigate this challenge with pre-course readings materials. However, students may not have time or motivation before class to digest these readings. Based on these sporadic occurrences, we have yet to find the right balance.

The expectations to the themes (8), (9) and (10) (using tools for assignments, coverage of contents in more domains and reducing the level of theoretical contents) occur in the earlier years but cease to occur after. We grouped these expectations under the type of managed expectations. Several participants suggest using tools to work on the assignments. We tried this out and many participants took more time to learn the tool instead. We eventually decide to leave the choice to the learners and they can use either pen and paper or their preferred tools. To address comments relating to contents being overly theoretical, we identified and modified them between course runs over the years. For course coverage, we send pre-course surveys to identify potential students who might not be our intended audience early and manage their expectations in the course coverage. Based on the data, we seem to have managed their expectations in these aspects.

TABLE III. 10 THEMES FROM THE FEEDBACK COMMENTS

No	Theme (Type of Expectation)	Selected Comments	Number of Public Participants (%)	Number of Postgraduate Students (%)
1	Number of case study discussions/ demos (Recurring)	"Bringing in more of real-life project examples would be a lot more helpful." "Propose to have more industry related workshops which can guide and make participants go through the thought processes of a junior architect. Maybe can include a small portion on what a solution architect goes through in an actual job environment" "Please give us more examples, will be helpful for those with limited tech background."	**104 (34.67%)**	2 (2.78%)
2	Number of practice exercises (Recurring)	"More personal exercise to ensure participants are able to apply software/solution architect" "Practical exercises after each chapter, to industries needed" "Include more practical exercises"	21 (7.00%)	9 (12.50%)
3	Level of guidance (Recurring)	"More 'hand-holding' for the workshop in groups. Perhaps, one before individual." "Although this is meant to be a technical course, some participants may not have sufficient background knowledge. Hence, some ideas can be simplified or use of some products may be helpful" "Go through exercises with participants, some may not have any foundation knowledge"	38 (12.67%)	**26 (36.11%)**
4	Depth of Content (Recurring)	"More in-depth workshops" "Could have spent more time going through course materials or in greater depth." "the architecture design portion should probably go into more depth with examples"	16 (5.33%)	9 (12.50%)
5	Course duration (Sporadic)	"Pace of the course. There are too much to cover within the 5-days." "The course probably covers too much in just 5 days but it is probably because SA covers a lot of scopes. Probably it can zoom into 1 or more modules in depth" "Reduce the number of participants per class, seems that instructor gets a lot of questions, some may not have chance to ask. Or increase the length of course"	12 (4.00%)	19 (26.39%)
6	Workshop Duration (Sporadic)	"There are too many assignments. Reduce the number of assignments." "The workshops might be too lengthy and not suitable for participants new to IT" "Workshops are too stressful, within a short timeframe"	23 (7.67%)	4 (5.56%)
7	Experience Sharing (Sporadic)	"Ask participants to share their architecture to know the application of concepts in real life" "2 instructors to engage in panel like discussion in conducting classes" "Encourage more participation of classmate to share what challenges and issues they have faced as SA and how they handle this situation."	27 (9.00%)	0 (0.00%)
8	Tool usage (Managed)	"Using computers to do the exercises, to reduce the time taken to re-draw diagrams." "The workshop can be done using pc with tools rather than have to draw diagrams by hand" "Appreciate if exercise is done using computer system instead of paper"	9 (3.00%)	1 (1.39%)
9	Content Coverage (Managed)	"More non-Java/c# examples like on c++ or other languages" "Relation to other systems? Real-time , Embedded & C/C++, SOA techniques" "I think that it would be useful to include how to evaluating software or libraries"	39 (13.00%)	0 (0.00%)
10	Theoretical Level (Managed)	"Certain computations are maybe slightly low level." "Some parts on performance are a bit too theoretical, need more intuitions" "Lesser calculations, more on concept and architect mindset."	11 (3.76%)	2 (2.78%)

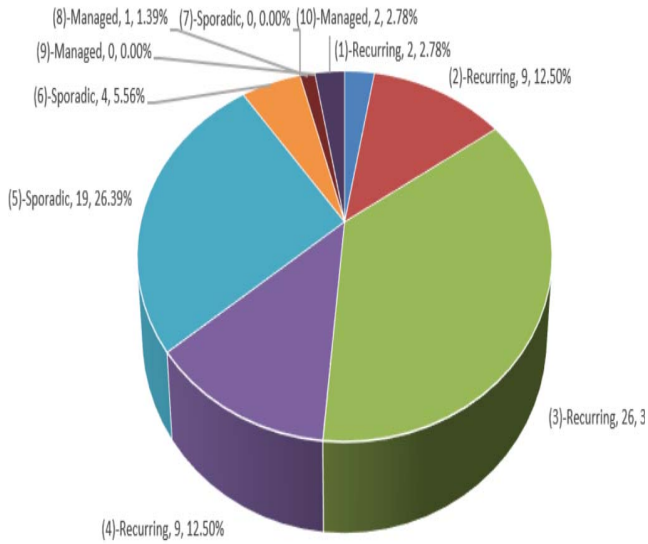

FIG. 3 NUMBER OF COMMENTS PER THEME, OCCURRENCES AND PERCENTAGE (POSTGRADUATE STUDENTS)

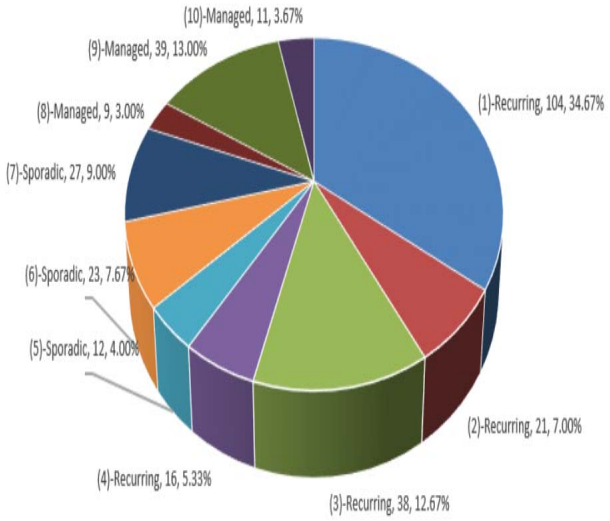

FIG. 4 NUMBER OF COMMENTS PER THEME, OCCURRENCES AND PERCENTAGE (PUBLIC PARTICIPANTS)

We also analyse the differences in expectations between public participants and postgraduate students with two key observations. 34.67% of the public participants felt the need to have more case study discussions and class demos (theme 1) to assist them in applying in their workplace as compared to only 2.78% of postgraduate students. On the other hand, 36.11% of the postgraduate students' significant feel the need to have more guidance (theme 3) as compared to 12.67% of the public participants. Since both courses are designed with the same contents and delivered by the same instructors, we attribute these observations to student's profile variation in terms of their years of experiences and work domains. Our profile of postgraduate students on average has fewer years of working experiences as compared to the public participants. The lack of sufficient working experiences might have limited their understanding, require more guidance and also lead to their high proportion of comments on the course duration being too short to cover the wide scope of the content.

Based on the above results to address RQ, we conclude that we meet the managed expectations. However, we still have work on addressing recurring and sporadic expectations. Two key design challenges are identified leading to the unmet sporadic and recurring expectations. These are the challenge to cover the extensive content within the limited duration and the challenge to adapt to the variation of student's profile, such as years of experiences and work domains. These challenges are likely inter-related - Students with fewer experiences or having unrelated work domains might require more time for guidance and vice versa.

One possible intervention to address these challenges is to divide the course into two levels - introductory and advanced. This approach increases the overall duration to cover more content and allow more experiences sharing. The level of guidance can also be higher in the introductory course. However, the downside to this approach is that the participants are away from work longer if they choose to attend both courses. Another possible intervention is to design the course for public participants and postgraduate students separately. Two different methods proposed are the problem-based and case-based methods. While case-based learning (CBL) uses a guided inquiry method and provides more structure during small-group sessions, problem-based learning (PBL) uses a more unguided approach [18]. CBL with a higher level of guidance is suitable for postgraduate students while PBL with a lower level of guidance is suitable for public participants who have working experiences. Both interventions attempt to address the challenges by minimizing the dependency of the course adaptation against the variation of the students' profile. However, more work has to be done to evaluate further if these interventions or a combination of interventions can address the challenges.

VI. Threats to Validity

As the design of software architecture is confined within the context and environment, our findings are also confined by the profile of our adult learners. Although these results are accumulated over many years and our students' profile vary to a certain extent, these results will require further validation when the profile of students changes significantly. In this paper, we studied the course design for two groups of software engineers who have a certain degree of industry experiences. These results might not be conclusive for other participants, such as undergraduates with little or no working experiences.

The course is designed for software engineers who are developing information systems architectures. Even though these architectural thinking concepts are also applicable in other types of system such as embedded systems, the results in this paper require further validation if the course design has to be adapted significantly for a learner who is working on other types of systems.

We also acknowledge that this evaluation of only a software architecture course limits the generalization to other courses. Although many of the analysis results are due to the variation of student's profile, we still need to validate these results in other courses in our future work.

VII. Conclusion

We believe that with the technology revolution and advancements in this rapidly changing IT industry, more software engineers are attending courses to upskill in their lifelong learning journey and enable them to perform better in their workplaces. There are challenges in designing the course to meet their expectations given the level of abstraction of software architecture concepts and learning characteristics of the adult learners. Can we effectively meet the learning expectations of these students in our architecture course?

In this paper, we first describe our software architecture course design based on established industry skills framework and understanding the characteristics of our adult learners. We conduct this course to two groups of adult learners – industry practitioners taking a public course and postgraduate students taking the course for their software engineering master's programme. Based on the quantitative analysis of the ratings and feedback comments for the course runs over eight years, the results show the effectiveness of our course design to meet the learning expectations of our students to learn software architecture. We also explain key interventions we put in as we evolve the course over the years based on the qualitative feedback comments in each run.

However, when we further analyze the qualitative feedback comments across the runs and over the years, we discover many areas of improvements, suggesting potentially unmet expectations. We cluster these comments into ten unique themes and classify these themes into three categories of expectations based on the frequency of occurrences of these comments. These three categories are recurring, sporadic and managed expectations. We analyze the themes and comments in these three categories and realise we can meet the managed expectations but not the recurring or sporadic expectations despite interventions. Recurring expectations refer to comments occur every year, sporadic expectations refer to comments occur within a two year period and managed expectations refer to comments occur before but ceased after that. We identified two key design challenges leading to the unmet sporadic and recurring expectations coverage of the extensive content within the limited duration and adapting to the variation of student's profile such as years of experiences and work domains. We also propose possible further interventions to address these challenges in the future.

We hope this discussion and trend analysis can help course designers to improve or evaluate a course design in related software engineering courses for software engineers.

REFERENCES

[1] S. O'Toole, and B., Essex, "The adult learner may really be a neglected species." Australian Journal of Adult Learning 52, no. 1: 183-191, 2012.

[2] M. S. Knowles, E. F. Holton III, and R. A. Swanson, The adult learner. Routledge, 2014.

[3] A. Tough, The adult's learning projects: A fresh approach to theory and practice in adult learning. Ontario Inst. for Studies in Education, 1971.

[4] J. S. Bruner, "The act of discovery." Harvard Educational Review.31: 21-32, 1961.

[5] R. W. White, "Motivation reconsidered: The concept of competence." Psychological review 66, no. 5, 297, 1959.

[6] P. Kruchten, "Mommy, where do software architectures come from." In Proceedings of the 1st Intl. Workshop on Architectures for Software Systems, pp. 198-205, 1995.

[7] C. R. Rupakheti, and S. Chenoweth, "Teaching software architecture to undergraduate students: an experience report." In Proceedings of the 37th International Conference on Software Engineering-Volume 2, pp. 445-454, IEEE, 2015.

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

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

[10] E. L. Ouh, and Y. Irawan, "Exploring Experiential Learning Model and Risk Management Process for an Undergraduate Software Architecture Course." In 2018 IEEE Frontiers in Education Conference (FIE), pp. 1-9. IEEE, 2018.

[11] "National Infocomm Competency Framework." [Online]. Available: http://www.ssg.gov.sg/wsq/Industry-and-Occupational-Skills/National-Infocomm-Competency-WSQ.html

[12] L. Bass, P. Clements, and R. Kazman, Software architecture in practice. Addison-Wesley Professional, 2003.

[13] E, L. Ouh, and S. Jarzabek, "An adaptability-driven model and tool for analysis of service profitability." In International Conference on Advanced Information Systems Engineering, pp. 393-408. Springer, Cham, 2016.

[14] "ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models." 2011. [Online]. Available: https://www.iso.org/standard/35733.html.

[15] P. Kruchten, The rational unified process: an introduction. Addison-Wesley Professional, 2004.

[16] P. Kruchten, "The software architect." In Working Conference on Software Architecture, pp. 565-583. Springer, Boston, MA, 1999.

[17] Azure Text Analytics API - Detect sentiment, key phrases and language from your text [Online]. Available: https://azure.microsoft.com/en-in/services/cognitive-services/text-analytics/

[18] M. Srinivasan, M. Wilkes, F. Stevenson, T. Nguyen, and S. Slavin, "Comparing problem-based learning with case-based learning: effects of a major curricular shift at two institutions." Academic Medicine 82, no. 1, 74-82, 2007.