Using Architectural Kata in Software Architecture Course: An Experience Report

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

ABSTRACT

Software Architecture Courses in Software Engineering curricula often involve project-based group assignments where students have to design and document software architectures and evaluate other architectures.

This paper presents the design and implementation of the Architectural Kata workshop conducted in the Software Architecture course as a group exercise to teach designing, documenting, and evaluating software architecture.

Feedback on the workshop was collected from students using a survey questionnaire after the conclusion of the course. The results showed that besides acquiring skills to design and evaluate architectural designs, the workshop also supported skills to identify non-functional requirements, elaborate assumptions, and collaborate with their peers.

Observations and lessons learned from the workshop about students' feedback, planning, and workshop moderation are shared. Possible future improvements are also suggested.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Applied computing** → **Collaborative learning**; • **Software and its engineering** → **Software architectures**.

KEYWORDS

Architectural Kata, Active Learning, Experience Report, Designing Software Architectures

ACM Reference Format:
Usman Nasir. 2023. Using Architectural Kata in Software Architecture Course: An Experience Report. In *ECSEE 2023: European Conference on Software Engineering Education (ECSEE 2023), June 19–21, 2023, Seeon/Bavaria, Germany*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3593663.3593694

1 INTRODUCTION

Designing Software Architecture is a crucial activity in modern software development, particularly for large-scale projects. It serves many purposes beyond documentation, such as examining the problem domain from different angles, creating code structures, and documenting technical decisions [10].

The Software Architecture course is essential to the Software Engineering curriculum and aims to teach students about designing the architecture of the software, its patterns, and design techniques. The course imparts various skills, including identifying quality requirements, architecture design, technical decision-making, and documenting architecture for multiple stakeholders.

Educators teaching Software Architecture courses use multiple teaching strategies and learning activities, such as hands-on exercises, games, and group projects, to help their students learn the necessary knowledge and skills [8].

The Architectural Kata is a group exercise to improve architectural design and evaluation skills, and it is popular within the software development community [5]. This experience report showcases the design and implementation of the Architectural Kata as a learning activity in a Software Architecture course at a European Technical University. This report disseminates the workshop experience, the challenges faced, and practical advice for instructors who plan to use it for their teaching.

The report is divided into four parts starting from section 2. Section 2 provides details of the case course where the Architectural Kata workshop was incorporated, the need for active and collaborative teaching approaches to teach Software Architecture courses, and the original Architectural Kata and its format. Section 3 presents the design and implementation of the workshop held as a learning activity in the course. Finally, section 4 shares the lessons learned and recommendations for teachers.

2 CONTEXT

2.1 Case Course: Software Architecture and Quality

The case course titled "Software Architecture and Quality" is offered to undergraduate and graduate students studying Software Engineering degrees at a European Technical University[1].

The course aims to enable students to gain knowledge about software architecture and techniques to satisfy quality attributes in architectural design. Additionally, it targets students developing the competence and skills to design, document, communicate and evaluate software architectures. According to Oliveira et al. [8], these course goals are typical for any Software Architecture course.

The course, by definition, is *"fundamentals of software architecture"* course with an emphasis on foundational concepts and skills of designing and evaluating software architecture. Its teaching offers conceptual knowledge complemented with case examples drawn

[1] Course Syllabus

ECSEE 2023, June 19–21, 2023, Seeon/Bavaria, Germany

ACM ISBN 978-1-4503-9956-2/23/06.
https://doi.org/10.1145/3593663.3593694

from well-known software architecture textbooks. The course's learning activities consist of lectures that mainly focus on the following Software Architecture concepts:

- Quality attributes & their role in architectural design.
- Architectural drivers, Architectural patterns and tactics.
- Documenting Software Architectures.
- Software Architecture evaluation.

The assessments for the course include a problem-based group project and an individual final written examination. In the group project, students work on three project-related assignments to design, document and evaluate architectural solutions for a software project case.

- Assignment 1: The student groups are tasked to derive and specify the given case's architectural drivers (business case, quality attributes, architectural concerns, and constraints). This assignment serves as input for the subsequent assignment.
- Assignment 2: The student groups use architectural drivers to design and document the software architecture for the given case.
- Assignment 3: The student groups evaluate other groups' architecture using software architecture evaluation methods and submit a written evaluation report.

A course map representing course learning outcomes (CLOs) and assessments that evaluate the attainment of learning outcomes is shown in Table 1.

Table 1: Course learning outcomes and assessments

	Course Learning Outcomes	Assessments
CLO1	Develop a simple software architecture to guide system development and achieve desired qualities	Assignment 1 & 2
CLO2	Make trade-offs between quality requirements and design decisions	Assignment 2
CLO3	Evaluate software architecture to assess meeting development needs and quality characteristics	Assignment 3
CLO4	Critique software architectural design decisions from an architectural perspective	

2.2 Teaching Software Architecture courses

Software Architecture courses are advanced-level courses in Software Engineering degrees usually offered in the last semesters or senior years once students have learned programming, requirement engineering, development processes, and software quality.

Software Architecture is a challenging course to teach from an instructional point of view, as architectural design concepts are multifaceted [2]. Students are required to understand many non-technical (business needs, concerns) and technical aspects (architectural patterns, tactics) [10].

Many instructors use the modelling of real-world systems to teach theory to students and prefer project-based learning pedagogical methodology for teaching software architecture [8]. Students often need help designing architectural models (solutions) in such courses as they lack experience and skills [2].

Dojo is a hands-on workshop session widely used in classroom settings where students can practice programming (Coding Dojo) or software testing (Test Dojo) in groups for collaborative learning. Rocha et al. used Designing Dojo to teach design patterns. They reported that their practice-oriented workshop significantly improved students' skills in designing software and applying design patterns [11].

Workshops where students collaborate with their peers, learn and practice ("learn-by-doing" approach) to design and evaluate architecture can be used by instructors in software architecture courses [3]. Paolo Ciancarini [1] used Neward's [7] Architectural Kata as an academic exercise in his Software Architecture course. However, he has not shared any further information on the impact of the activity in the class.

Further details about the Architectural Kata and its format are explained in the next section.

2.3 Architectural Kata

The concept of "kata" originated from Japanese martial arts training, where learners repeatedly practice a specific sequence of movements to master a particular skill [6]. Code Kata is a popular exercise for enhancing developers' expertise in programming languages or programming practices [12].

Architectural Kata, inspired by Code Kata, is a group exercise aimed at improving software architecture design skills, allowing participants to gain experience and receive feedback from other participants [7]. Such activities are conducted to promote good architectural practices, focusing on specific architectural styles, such as monolithic or micro-services architecture, or specific quality requirements (security, data privacy, performance etc.) [5].

The Architectural Kata has multiple variations with different objectives and targeted audiences. Multiple Kata is an architectural design training exercise that spans a day or a week to train software development practitioners within an organisation [5]. Recruitment Kata (or Kata interview) is used to evaluate applicants' design skills in the recruitment process [5].

Architectural Katas are events where software development practitioners gather to participate in Architectural Kata exercises. These events are held at local software development community meetups, conferences, workshops, and training events. O'Reilly Media organises Architectural Katas as a competitive event for software architecture teams [4]. Competition participants are given a challenge faced by an actual organisation, and they have to solve it within the given time frame, which is almost a month. The teams submit their solutions to a panel of judges who select finalists.

In Architectural Kata, the project or design challenge that requires an architectural solution is referred to as *kata problem*, which are realistic high-level problem descriptions with details about users, requirements and additional problem context. The problem descriptions leave room for assumptions and technical choices in the solution.

There are several public repositories for kata problems, of which one primary source is Neward's website [7]. Here, an active community of developers and practitioners continually adds new kata problems to the repository.

The Architectural Kata is set up with participants divided into teams that work together. The moderator assigns a kata problem to each team. The moderator also acts as the customer of the kata problem during the exercise. Ted Neward's [7] format of the Architectural Kata is divided into three phases:

(1) **Discussion Phase:** In this phase, teams explore the given kata problem. The teams must consider stakeholders' concerns and technological constraints to design a solution to satisfy the requirements described in the kata problem. A team may ask the customer for clarification. Teams can make their own assumptions but need to specify them when presenting their architecture.
(2) **Peer-review Phase:** In the second phase, teams present their architectural design to other teams and answer the questions that are posed to them.
(3) **Voting Phase:** In this phase, participants of other teams individually vote on the presented solution. A three-point scale: Thumbs up, Thumbs-meh and Thumbs down, is used to vote. Once voting is complete, the team departing the stage chooses the next team to present their solution.

This exercise allows software developers to learn and practice designing software architecture as a new skill. Besides software developers, even experienced software architects can experiment with new ideas and approaches, thus appreciating the Architectural Kata exercises [5].

3 WORKSHOP

In the recent iteration of the case course, an Architectural Kata workshop was conducted to support students in learning and developing new concepts or skills. The workshop was designed as a learning activity for students to practice the skills needed to complete their assignments. This involved working collaboratively to identify architectural drivers, design software architecture, and evaluate architecture created by others.

The workshop's main objective was to support students in learning the skills of identifying non-functional requirements, architectural drivers, designing an architectural solution for a given problem and evaluating an architectural design solution.

The students attending the workshop had prior knowledge of functional and non-functional requirements, architectural concerns, software architecture design concepts (patterns and tactics), modelling of system context and architectural structures from course lectures.

The following sections give a more detailed account of the workshop's design and implementation.

3.1 Design

The workshop's design is adapted from Neward's Architectural Kata [7] and Ciancarini's classroom exercises [1]. The design of the Architectural Kata workshop is as follows:

Planned Duration: Two hours

Group size: 5-6 students in a group (Teacher may form the groups)

Kata problems: 5-6 kata problems

Teacher's role: To act as Architectural Kata's moderator and as a customer/stakeholder for the kata problems.

The format of the workshop has two phases:

(1) **Discussion and Design Phase:** In this phase, student teams explore the assigned kata problem, identify architectural drivers (quality requirements, constraints etc.) and design an architectural solution to satisfy the stated requirements.
(2) **Peer review and Voting Phase:** In this phase, student teams present a vision of their architectural solution to other teams and answer their questions. Then other students evaluate and vote on the presented solution.

Figure 1 shows how the Architectural Kata workshop's phases and activities support the course learning outcomes as shown in table 1.

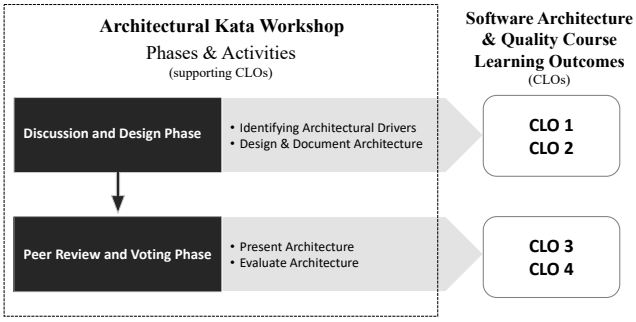

Figure 1: Workshop supporting course learning outcomes

In the Discussion and Design Phase of the workshop, participants, as a team, document an architectural design of the solution for the kata problem as a presentation to their peers (referred to as a vision of the architecture). The presentation should include an overview of the problem, identified stakeholders' needs & non-functional requirements, a system context diagram, and a representation of the architectural solution (code structures in the form of a block diagram). The students must also explain the reasoning behind their architectural and technical decisions.

In the Peer review and Voting Phase of the workshop, after each presentation, participants are asked to individually assess the presented solution and vote using the following three-point scale adapted from Ciancarini's exercise [1]:

- Nailed it: The presented vision addressed all the key questions and had feasible design decisions.
- Missed a few things: The presented vision missed some key questions or important aspects.
- Missed it, badly: The presented vision missed key questions or made major invalid assumptions.

3.2 Implementation

The planned duration of the workshop was two hours, with the first twenty minutes for introduction and team formation, forty minutes for the discussion and design phase, and then the next hour for the peer review and voting phase. The workshop duration was set to two hours to align with the time slots allocated for all activities of the case course.

Several kata problems from Neward's repository [7] were selected for the workshop. The primary criterion for selecting the

kata problems was that the students could easily comprehend them. For instance, Check Your Work (a grading system for programming assignments) was chosen as a kata problem for the workshop (Figure 2).

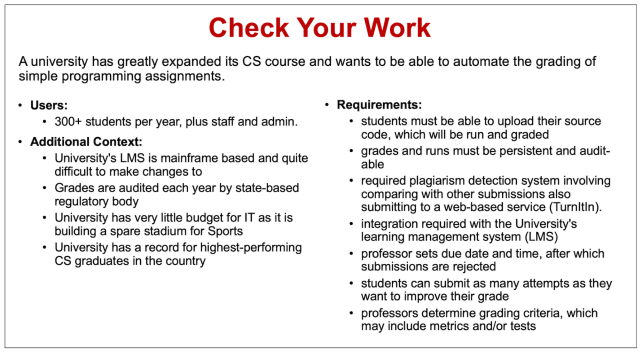

Figure 2: Kata problem: Check Your Work [7]

After a brief introduction to the workshop, students were randomly divided into five teams (of five to six members) and every team was assigned a kata problem. Each student team had 40 minutes to develop an architecture to address the requirements specified in the kata problem. The teams could make reasonable assumptions about requirements and use any technology stack.

During this first phase, the moderator approached each team to support them, observe their work and answer their questions.

Once the 40 minutes were up, all student teams presented their proposed architecture. The student teams presented the vision of their solution (Figure 3).

All participants had the opportunity to read and familiarise themselves with the kata problems shared in the workshop before the presentations, ask questions, seek clarification, and even criticise design decisions. Students evaluated and provided feedback on their peers' solutions using Mentimeter (online audience poll), which ensured the anonymity of the evaluators.

Throughout the second phase, the moderator provided support to the student teams during their presentations but did not offer any feedback on their presentation.

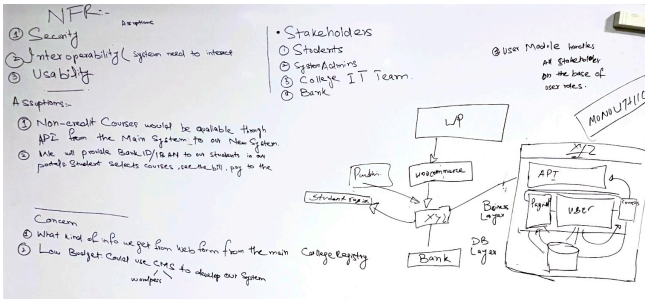

Figure 3: Student team's architectural solution presentation

4 LESSONS LEARNED AND RECOMMENDATIONS

4.1 Student's feedback

Initially, informal feedback was collected at the end of the workshop. Later, after the course was completed, a survey questionnaire was sent to 26 students who attended the workshop. Participation in the survey was voluntary, and 11 students returned the surveys, resulting in a response rate of 42%.

The survey findings revealed that most students had reported that they learned skills such as identifying non-functional requirements, making, and elaborating assumptions, evaluating architectural designs, and collaborating with others. The survey responses, shown in figure 4, indicate that the workshop was effective in teaching students software architectural design and evaluation skills. However, some students had divergent views, with a small percentage indicating that the workshop offered little help documenting the software architectural design (40% of responses). It should be noted that the documentation produced during the workshop was a simple presentation (vision), which contrasted sharply with the detailed architectural documentation required for the course assignment. The latter assignment called for documenting multiple aspects of the architecture design solution.

In response to a question about other skills acquired from the workshop, participants mentioned learning to "scope the application" and "map the quality attributes with suitable architectural design patterns" as skills.

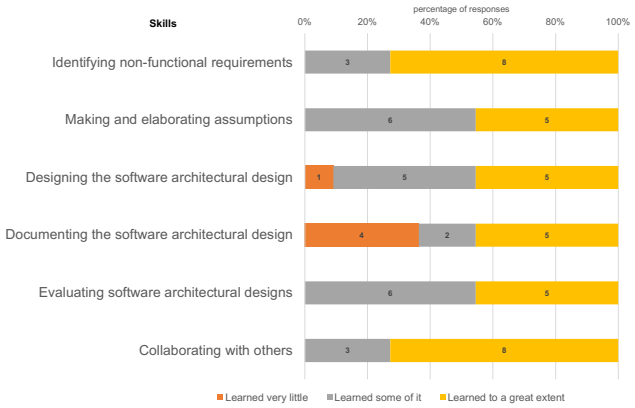

Figure 4: Student response on skills learned in the workshop

The response to open-ended ended questions provided valuable insights into the strengths and weaknesses of the workshop design. When asked about the challenges during the workshop, participants commented that "we couldn't fully understand the work initially". They felt that it took some time for them to understand how to do the given task. Participants also reported knowledge gaps or disparities in the expertise of their team members as hindrances in communication and collaboration.

One central theme in participants' comments was the challenge of working in a short time frame and the difficulty in understanding the ideas of other members. Participants felt that the limited time

frame to develop an architectural solution was challenging, making it hard to prepare their presentations.

Students were asked about the usefulness of the workshop in completing the group assignments of the software architecture course. Most respondents found the workshop to be helpful in completing Assignment 1 (identifying architectural drivers) and Assignment 2 (designing and documenting architecture). However, 20% of respondents felt that the workshop provided limited help in completing Assignment 3 (evaluating architecture).

4.2 Observations

As a moderator, it was observed that 40 minutes allocated to the discussion and design phase was insufficient. Student teams struggled to complete their presentation, and one student team encountered additional challenges completing their design task due to other team members participating remotely.

The additional context provided in the kata problem caused team discussions to derail. Student teams wasted their time as they over-analysed the given problem.

As the workshop's problem mimicked real-world scenarios, students practised decision-making and trade-off analysis, which are crucial aspects of the architectural design process. Besides that, student teams were also thinking of designing or attempting to design a comprehensive architectural design that addresses all non-functional requirements.

The workshop facilitated students to learn how to make decisions and do trade-offs in situations where business needs and quality requirements compete. During the workshop, student teams were prompted to analyse, do trade-offs, and prioritise non-functional requirements.

Workshop also helped students to learn that architecture design requires identification and comprehension of the architectural drivers before the commencement of the design. Some student teams learned this fundamental lesson the hard way once their peers identified that they had ignored some contextual information, rendering their assumptions and technical choice unfit.

4.3 Recommendations for Teachers

The only pre-requisite for conducting the Architectural Kata workshop is the selection of kata problems and teacher's preparation to act as a moderator. The moderator's role is crucial in student engagement during the workshop's discussion and design phase. Teachers can gain knowledge of kata problems and solutions from software architecture textbook [10] and solutions available on GitHub [9].

When selecting kata problems for the Architectural Kata workshop, instructors should consider the appropriate level of difficulty and relevance to their context. It is important to note that the kata problems are designed for seasoned software development professionals rather than students. Thus, teachers may adapt the selected kata problems by removing or altering requirements if required.

More time should be allocated to the discussion and design phase for the Architectural Kata workshop. The moderator should prompt students to focus on the problem, prioritise non-functional requirements and identify architectural drivers before commencing the design work.

5 CONCLUSION

As per students' feedback, the Architectural Kata workshop facilitated collaborative work and resulted in the improved acquisition of conceptual knowledge and skills. However, the validity of the feedback may be limited as it was collected from only one workshop that was conducted during a single course. In the feedback regarding the class activities, student bias is a risk. However, since the survey was carried out after the completion of the course where the participation was voluntary; the risk of bias in responses was likely mitigated because the respondents were aware that their feedback would not impact their grades.

To enhance the overall student experience and learning, future workshop iterations are planned and would include improvements such as allocating more time for the discussion and design phase.

Software Architecture instructors are encouraged to incorporate the Architectural Kata workshop into their course curriculum as an instructional activity.

ACKNOWLEDGMENTS

I would thank Muhammad Usman for reviewing an earlier version of the manuscript. This work has been partially funded by Education Development Unit at Blekinge Institute of Technology, Sweden.

REFERENCES

[1] Paolo Ciancarini. 2020. *Lecture: Architectural Kata.* Retrieved March 08, 2023 from https://www.cs.unibo.it/~cianca/wwwpages/as/7.pdf
[2] Matthias Galster and Samuil Angelov. 2016. What Makes Teaching Software Architecture Difficult?. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C).* 356–359.
[3] Anderson Cavalcante Goncalves, Valdemar Vicente Graciano Neto, Deller James Ferreira, and Uyara Ferreira Silva. 2020. Flipped Classroom Applied to Software Architecture Teaching. In *2020 IEEE Frontiers in Education Conference (FIE).* 1–8. https://doi.org/10.1109/FIE44824.2020.9274255
[4] O'Reilly Media, Inc. 2022. *Architectural Katas (Live Event).* Retrieved March 8, 2023 from https://www.oreilly.com/live-events/architectural-katas/0636920458487/0636920458463/
[5] Andy Marks. 2020. *Learning with Architecture Kata: Workshops & Recruitment.* Retrieved March 08, 2023 from https://vampwillow.wordpress.com/2020/04/01/learning-with-architecture-kata-part-8/
[6] Utathya Nag. 2010. Everything you need to know to about karate kata. Retrieved March 08, 2023 from https://olympics.com/en/news/karate-kata-martial-arts-history-how-many-forms-meaning-rules-scoring
[7] Ted Neward. 2014. *Architecture Katas.* Retrieved March 8, 2023 from https://www.architecturalkatas.com/
[8] Brauner R. N. Oliveira, Lina Garcés, Kamila T. Lyra, Daniel S. Santos, Seiji Isotani, and Elisa Y. Nakagawa. 2022. An Overview of Software Architecture Education. In *Proceedings of the 25th Ibero-American Conference on Software Engineering* (Natal, Brazil) *(CIbSE 2022).* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.5753/cibse.2022.20964
[9] PegasuZ. 2022. *Spotlight Platform - Spring 2022.* Retrieved March 12, 2023 from https://github.com/z-katas/arch-katas-dcc
[10] Mark Richards and Neal Ford. 2020. *Fundamentals of Software Architecture: An Engineering Approach.* O'Reilly Media.
[11] Fabio Gomes Rocha, Rosimeri Ferraz Sabino, and Guillermo Rodriguez. 2018. Using Dojo as a Pedagogical Practice to Introduce Undergraduate Students to Programming. In *2018 XIII Latin American Conference on Learning Technologies (LACLO).* 13–16. https://doi.org/10.1109/LACLO.2018.00012
[12] Dave Thomas. 2014. *CodeKata.* Retrieved February 10, 2023 from http://codekata.com/kata/codekata-intro/