# A Collaborative Classroom Activity for Teaching Cloud Architecting

Steven Santillan
*Escuela Superior Politécnica del Litoral, ESPOL*
Guayaquil, Ecuador
steisant@fiec.espol.edu.ec

Cristina L. Abad
*Escuela Superior Politécnica del Litoral, ESPOL*
Guayaquil, Ecuador
cabad@fiec.espol.edu.ec

***Abstract*—We present a collaborative classroom activity aimed at helping students improve their cloud architecting skills. The activity is a collaborative design exercise where students are given a (synthetically generated, realistic) set of cloud components and are asked to think about how they could be integrated into a single architecture while considering possible real-world use cases, how the components could be connected, and whether additional cloud components are missing from the design. The lists of components provided to the students are synthetically generated to mimic real cloud architectures from a recent public dataset. Instructors can seed the generator with specific components to generate a list suitable for a specific type of architecture; e.g., FSx (Lustre) for HPC, Elastic MapReduce (Hadoop/Spark/...) for Big Data, or Lambda@Edge for edge computing. We present observational and questionnaire-based results from two runs across two semesters (18 students in semester 1 and 49 students in semester 2) in an undergraduate *Distributed Systems and Cloud Computing* class at ESPOL university in Ecuador. Student responses highlighted the positive value of the activity and showed that they were highly engaged and applied critical thinking and teamwork during the exercise. Our synthetic generator tool has been released on GitHub so that others can use it or adapt it.**

***Index Terms*—Education; cloud computing; distributed, parallel, and cluster computing; software performance engineering**

## I. Introduction

The ability to design reliable, scalable, performant, and cost-efficient cloud architectures is an important software practice skill that comes with years of experience. However, as it has been reported in recent years, there is a *cloud talent drought* or *cloud skills gap* [1]–[3] in part because it is a skill known to be hard to teach in the classroom [4].

Architecting for the cloud, or cloud architecting as we call it herein, is a key knowledge area in Cloud Computing education [5]. In the ACM/IEEE-CS Computer Science curricula, these skills fall within the *Software Engineering - Design* and *Systems Fundamentals - Reliability* knowledge areas [6]. The ITiCSE Working Group on Cloud Computing [7]–[11] has defined a set of Cloud Computing learning objectives, including several related to architectural design.[1] In addition to cloud architecting itself, designing these architectures can help students think deeply about non-functional properties like performance and reliability, as well as pervasive concepts like locality, latency, load balancing and scaling, that have been highlighted by the NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing (PDC) [12] and are frequently present in Distributed Systems syllabi [13], [14].

Skills like cloud architecting can be developed via course or capstone projects [6], [8]; however, with the rise of active learning and flipped classroom approaches [15] and the desire to engage students while avoiding the abuse of generative AI tools [16], activities that can be done during a class session are increasingly in demand. Further, when teaching new skills, unplugged activities can provide a high-level overview of a topic before getting into technical details [17], and have proved to be liked by students and instructors [17], [18].

To address these issues, we propose a collaborative activity to help teach cloud architecting. The students are organized in teams and provided with a set of cloud components which they are asked to integrate into a single cloud architecture that can be used to solve a real industry problem of their choice. The students provide written and oral justifications of their designs, allowing for instructor and peer feedback.

We ran two tests of the activity in 2025 and administered a questionnaire to gather information on the clarity of instructions, usefulness of support materials, and relevance of the discussion and feedback, helping us make adjustments for future implementations. Results show that students liked the exercise and think it was a positive learning experience.

In this work we make the following contributions:

1) Present a tool for synthetic generation of cloud architecture components based on real cloud architectures,
2) describe a classroom activity based on this tool that can be used to improve students' cloud architecting skills,
3) analyze two small-scale runs of the activity to show its usefulness in a real classroom setting, and
4) the data, code and instructional material has been released for others to easily build upon our work.[2]

*Applicability outside our own local context:* We developed the activity to be used in our Distributed Systems and Cloud Computing class, which has weekly hands-on labs on AWS. As a result, our experience report is AWS centric. However,

[1] E.g. , Design a cloud service solution that is fault tolerant and meets the minimum SLA requirements; Design a secure architecture model for a cloud-enabled computing system using security design principles

[2] See: https://github.com/disel-espol/cloud-arch-teaching-synth

we have released three alternate versions with components for Google Cloud Platform, Microsoft Azure, and a generic version suitable for in-house designs for a company's datacenter. Further, the activity can be used in other courses by manually selecting a seed component from emerging areas like Big Data, IoT or edge computing, which the TCPP PDC initiative [12] describes as "topics of significant [...] interest that can serve as suitable backdrops to explore several PDC ideas."

The rest of this paper is organized as follows: Section II describes the design of the cloud component generator and our collaborative activity. In Section III we present our experience report (results). Section IV discusses the limitations of our work and possible improvements and Section V discusses the related work. In Section VI we conclude.

## II. Methodology

In this section, we describe the design of the classroom activity and the tool (synthetic generator) that is used to generate the preparatory material. In short, the activity consists of having small teams of students design a cloud architecture using a random set of cloud components. Students should think of a use case that would combine all the provided components, indicate the relationships between these, and suggest adding components if needed. In § II-A, we describe how we randomly generate the components. In § II-B, we describe the design of the classroom activity.

### A. Randomly selecting the cloud components

We propose two ways of selecting cloud components for each group to work with: (1) selecting all the components from a random cloud architecture from Cloudscape, a recently published dataset of real cloud architectures on AWS [19]; or (2) synthetic generation: randomly generating a list of components that follow patterns mined from the Cloudscape dataset. The advantage of (1) is that after submitting their designs, students can "validate" them with the original architecture. However, this check may discourage students if their proposals—even if valid—differ significantly from the source. For this reason, we have implemented a tool that generates lists of random cloud components that are based on architectural patterns mined from the Cloudscape dataset. In the remainder of this subsection, we describe our synthetic generation process.

To create a tool (synthetic generator) that randomly generates a set of components that can likely be integrated into a useful cloud architecture, we use **frequent itemset mining** to generate **association rules**, a data mining approach used to discover relationships between items in a dataset.[3]

We model each architecture $t$ as a transaction containing a subset of cloud services $S_t \subseteq U$, where $U$ is the universe of available cloud services. We consider a subset $I \subseteq U$ a frequent itemset if its support is above a pre-defined threshold:

$$\text{supp}(I) = \frac{|\{t : I \subseteq S_t\}|}{N},$$

[3]These techniques were originally proposed to discover common patterns in consumer transactions; e.g., cereal is usually bought together with milk.

where $N$ is the total number of architectures. A subset $I$ is *frequent* if $\text{supp}(I) \geq$ `min_support`. To balance realism and diversity we established two support thresholds:

- `min_support` $= 0.06$: Favors identifying robust patterns by capturing recurrent dependencies.
- `min_support` $= 0.03$: Allows us to recognize extended patterns, incorporating less frequent combinations and fostering the diversity of configurations.

From the frequent *itemsets* we derive association rules:

$$X \Rightarrow Y \quad \text{with} \quad X \cap Y = \emptyset,$$

where $X$ is called the antecedent and $Y$ the consequent; if $X$ occurs in a dataset, then $Y$ will as well. The association rules let us find components that occur in a dataset given that another component is already present.

We evaluate these rules using three classic metrics, support, confidence and lift, each described next.

$$\text{supp}(X \cup Y) = \frac{|\{t : X \cup Y \subseteq S_t\}|}{N},$$
$$\text{conf}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} = P(Y|X),$$
$$\text{lift}(X \Rightarrow Y) = \frac{\text{conf}(X \Rightarrow Y)}{\text{supp}(Y)} = \frac{P(X \cup Y)}{P(X)P(Y)},$$

Support measures the proportion of architectures that contain all the services involved in the rule. Confidence estimates the conditional probability of observing $Y$ given that $X$ is already present. Lift quantifies the strength of the association, indicating how many more times we observe $Y$ together with $X$ compared to what we expect under statistical independence. We filter the rules considering the following thresholds: $\text{conf} \geq 0.60$ and $\text{lift} > 1.05$, so that we only retain reliable, non-trivial, statistically significant associations.

We organize the frequent itemsets and association rules into two complementary catalogs: (i) the core catalog, $\text{supp} = 0.06$, contains reliable patterns that guarantee realism; (ii) the extended catalog, $\text{supp} = 0.03$, has patterns that contribute to diversity. During the generation phase, we follow these steps: (1) We randomly or manually select an initial seed, corresponding to a (single-component) partial architecture; (2) apply association rules whose antecedents are already present; (3) incorporate the consequents stochastically, weighted by $\text{score}(r) = \text{conf}(r) \cdot \log_2(\text{lift}(r)) \cdot \text{supp}(r)$, which privileges solid rules and reduces the probability of spurious expansions; and (4) if no applicable rules exist, we use conditional pairwise co-occurrence as a fallback mechanism. In step (1), a manual seed is useful if the goal is for students to work on a specific type of cloud architecture. For example, choosing AWS Lambda as seed when teaching about serverless architectures.

We developed the synthetic generator in Python using the `mlxtend` package and the functions `fpgrowth` and `association_rules` to extract frequent itemsets and generate the association rules, respectively. Each rule has the form $X \Rightarrow Y$, with associated metadata for confidence, $\text{lift}(X \Rightarrow Y)$ and $\text{supp}(X \cup Y)$. Using the two support

thresholds independently, we obtain the two catalogs that emphasize realism (0.06) and controlled variety (0.03). As a fallback, we use frequent subsets $I \subseteq S$ with support greater than or equal to the corresponding threshold, guaranteeing that expansions come from empirically observed co-occurrences.

To avoid generating the same architecture every time, we select the consequents using a weighted random generator with the following weight for each rule:

$$w(X \Rightarrow Y) = \text{conf}(X \Rightarrow Y) \cdot \text{supp}(X \cup Y) \cdot \log_2 \max(\text{lift}(X \Rightarrow Y), 1 + \varepsilon),$$

with $\varepsilon \approx 10^{-12}$ for numerical stability.

*Summary of the synthetic generation process:* The generator is configured with the two rule catalogs. Given a partial state $S_t$, it performs weighted random selection to choose a $y \in Y$ (if such rules exist). In the absence of rules, it selects items from the frequent subset catalogs (fallback). The process repeats until it reaches `max_size` or exhausts the candidates, producing plausible and varied architectures consistent with the mined architectural patterns.

*Validation of the synthetically generated sets:* To evaluate the generator's ability to produce realistic sets of cloud components that could be integrated into a real application, we used a 10-fold stratified cross-validation test with `max_size` = 10, `n_candidates` = 100) and a mixed seed policy (50% real, 50% random architectures). Results confirm that the generator respects the typical scaling of architectures ($\approx$ 8-9 services), maintains close marginal distributions (JSD $\approx 0.18$-$0.19$), preserves the co-occurrence structure with moderate discrepancies (Frobenius $\approx$ 3.6-3.9), and ensures local realism (NN-Jaccard $\approx 0.33$-$0.34$). Furthermore, it achieves significant diversity (intra-Jaccard $\approx$ 0.24-0.25, coverage $\approx$ 0.27-0.28, entropy $\approx 0.67$). Overall, the 10-Fold validation supports the generator's validity: The synthetic architectures are plausible, diverse, and consistent with empirical evidence. Support at 0.06 offers greater stability, while 0.03 increases coverage and diversity. Detailed results (incl. .632 bootstrap) in our repo.

### B. Design of the classroom activity

We designed an activity aimed at a senior-level undergraduate course as summarized in Table I. Our goal was to promote active learning on the topic of designing cloud architectures, targeting four ITiCSE Cloud Computing WG learning objectives (see Table II). At the core of the activity is a synthetic generator that outputs a list of components that can likely be integrated into a cohesive cloud architecture, following architectural patterns mined from a real cloud architectures. We assign these lists to small teams who then propose a design that can solve a real industry problem and justify their decisions in written and oral form, as detailed next.

*Material provided to the students:* Each group receives a different list of cloud components that has been synthetically generated using our tool. We also include a printed cloud service cheat sheet with high-level descriptions of the functionalities of the cloud services included in the assignment, as well as other useful services. For example, if students receive an architecture that contains edge services, the cheat sheet may include a description of the following services: Greengrass, IoT Core, SageMaker Edge Manager, Storage Gateway, AWS Lambda, S3, EC2 and DynamoDB.

TABLE I
ACTIVITY AT A GLANCE (INCL. RELEASED ARTIFACTS). *Qn*: QUESTION *n*.

| Item | Description |
|---|---|
| Target skill | Cloud architecting (cloud infrastructure design) |
| Setting | Collaborative design; active learning; unplugged |
| Group size | 2–4 students per team (pairs in our runs) |
| Inputs | Synthetically generated set of cloud components; service description cheat sheet; student worksheet |
| Duration | 2+ hours (with checkpoints) |
| Core tasks | (i) choose a real-world use case; (ii) draw a service-flow diagram; (iii) add/remove up to 3 services with justification; (iv) state design assumptions; (v) propose validations/metrics |
| Suggested schedule | 20 min: Instructions; 30 min: Architecture diagram; 20 min: Q1 (added/removed services); 20 min: Q2 (design assumptions); 20 min: Q3 (validations and metrics); 10 min: Final review + submission |
| Deliverables | Diagram; written justification; short oral explanation |
| Evidence collected | Post-activity questionnaire; written report; instructor observations; (two runs across two semesters) |
| Released artifacts | Generator notebook; activity handout; schedule; evaluation rubric; survey instrument; raw survey data; analysis notebook; optional statistical test notebook; supporting datasets/metadata for generation |

*Group work:* Students work in randomly assigned teams to propose a design (cloud architecture) that integrates the components and describe a use case for which their design would be reasonable. Students can add/remove 0-3 components if the provided list can't be integrated into a single architecture as-is.

*Written deliverables:* Each group documents their design in writing on a provided sheet of paper, including a diagram with directed edges showing the interactions between components, and a justification of their design choices, assumptions, and proposed validations and metrics that target: performance (latency or throughput), scalability or fault tolerance.

*Discussion and feedback:* After documenting their designs, students must orally present their proposals and get feedback from the instructor and peers. The instructor provides immediate feedback and follow-up questions focused on the coherence of the data flow, the consistency of the design, and the technical relevance of the proposed changes. Based on these exchanges, and if time allows, the teams can iterate on their deliverables making adjustments to the diagram and textual answers, indicating which changes have been made in response to the feedback received.

*Assessment and grading:* We devised a rubric with six criteria:[4] (C1) Active Participation, 15 pts, evaluates student initiative in proposing ideas and contributing to the design; (C2) Collaboration and Exchange of Ideas, 20 pts, considers the ability to communicate within the team and with other

[4]The detailed rubric is available in the GitHub repository.

TABLE II
MAPPING ITICSE CLOUD WORKING GROUP LEARNING OBJECTIVES (LOS) TO ACTIVITY TASKS.

| **WG learning objective (LO)** | **Where it appears in our activity** |
|---|---|
| Design network architecture as part of utilizing cloud resources | Students connect the given components into a service-flow diagram (data paths, request flow, integration points) and justify connectivity choices |
| Describe cloud performance models and discuss common design patterns | In the justification, teams discuss latency/cost/scalability trade-offs (e.g., caching, async messaging, autoscaling) and why the architecture fits the chosen use case |
| Design a cloud service solution that is fault tolerant and meets minimum SLA requirements | Teams state reliability assumptions and propose validations/metrics (e.g., redundancy, failure modes, availability targets) |
| Design a secure architecture model using security design principles | Teams identify trust boundaries and add missing security-related components (e.g., IAM, encryption, network controls) as part of the "up to 3 changes" rule |

groups; (C3) Justification of Changes to the Architecture, 20 pts, assesses the clarity of the explanation of added/removed services and the understanding of their role in a solution; (C4) Design Assumptions, 20 pts, weighs the technical soundness of the assumptions made to produce the design; (C5) Proposed Validations and Metrics, 15 pts, examines the ability to translate theoretical concepts into non-functional indicators (e.g., scalability, fault tolerance); and, (C6) Clarity and organization of the report, 10 pts, evaluates the written report.

## III. EXPERIENCE REPORT: RESULTS

We piloted the activity during two 2-hour runs with students from a *Distributed Systems and Cloud Computing* class at our university. We administered a questionnaire immediately after the classroom activity, ensuring students responded with the experience fresh in their minds. For the semester 1 run (18 students, G1) the participation was completely voluntary and the activity was scheduled at a day/time where all the students were available. For the semester 2 run (49 students, G2) the activity was scheduled during one of the regular semester sessions, two weeks before the final exam, with an extra 30 minutes as our sessions are 1.5 hours long. For this second run, the participation was voluntary in the sense that the grade obtained in the activity did not count towards their final grades nor was their presence mandatory. However, as the activity was scheduled during a regular class session most of the students showed up to class. Participants completed the survey anonymously and we properly disclosed that the answers would be collected and used solely for academic purposes; all students consented to research participation.

We designed and applied an anonymous post-activity questionnaire to evaluate students' perception and satisfaction. The questionnaire had two main goals: (i) to collect information on participants' experience regarding motivation, learning, and methodology, and (ii) to generate input for continuously improving the pedagogical approach. We structured the questionnaire into six sections with thirteen close-ended 5-point Likert-scale questions and two open-ended questions as shown in Table III. The results, in Figure 1, show that the students overall liked the activity and thought it would be a positive addition to the class in future semesters.

### A. Likert-scale questionnaire results

*Student satisfaction results:* Questions P1–P3 had high mean values for both groups (P1-G1=4.78, P1-G2=4.51; P2-G1=4.83, P2-G2=4.55; P3-G1=4.78, P3-G2=4.55), indicating high satisfaction with the activity in terms of enjoyment and motivation. The narrow confidence intervals suggest consistency in student perceptions.

TABLE III
POST-ACTIVITY QUESTIONNAIRE. ANSWERS FOR P1–P13 IN LIKERT-SCALE (1:STRONGLY DISAGREE AND 5:STRONGLY AGREE).

| |
|---|
| Section 1. Satisfaction and motivation |
| P1: I enjoyed participating in the classroom activity. |
| P2: The classroom activity was interesting and held my attention. |
| P3: I felt motivated to participate actively in my group. |
| Section 2. Utility for learning (I) |
| P4: The activity helped me better understand the concepts of distributed architectures. |
| P5: The practical exercise was more useful than just reviewing theory. |
| P6: I would like this type of activity to be included regularly throughout the semester. |
| Section 3. Utility for learning (II) |
| P7: I can apply what I learned in this activity in academic or professional projects. |
| P8: Designing the architecture helped me connect theory with real-world use cases. |
| Section 4. Classroom activity methodology |
| P9: The instructions were clear and easy to follow. |
| P10: The support material (cheat sheet, guide) was useful for completing the activity. |
| P11: The time we allotted for the classroom activity was adequate. |
| Section 5. Collaboration and discussion |
| P12: The group work helped me learn from my classmates. |
| P13: The exchange of ideas between teams was valuable for enriching my proposal. |
| Section 6. Open-ended |
| P14: What did you like most about the classroom activity? |
| P15: What would you improve for future editions? |

*Perception about the activity's utility:* Questions P4-P8 focus on the understanding of key concepts, the perceived value of the classroom activity, and the applicability of the concepts in academic and professional contexts. Answers for all four questions had high averages P4 (G1=4.78, G2=4.35), P5 (G1=4.67, G2=4.39), P6 (G1=4.94, G2=4.45), P7 (G1=4.72, G2=4.55) and P8 (G1=4.61, G2=4.33). G1:P6 stands out with the highest average score across the entire questionnaire (4.94) and while G2:P6 is lower (4.45), it's still a high score, indicating that students want this type of activity to be integrated into the course. The perceived utility for understanding complex concepts and the value of the practical exercise as a complement to concepts studied in class support the idea that collaborative active learning can generate a deeper formative impact than

a merely expository approach. However, the mean scores for P7 and P8 are slightly lower than for other questions, hinting that students are slightly less confident that the activity will be useful to prepare them for industry.

*Evaluation of the activity design:* Questions P9 (G1=4.72, G2=4.29) and P10 (G1=4.78, G2=4.53) show a positive evaluation of the methodological aspects and that the instructions and support material were clear and relevant. Conversely, P11 (G1=4.67, G2=3.31), which corresponds to the allotted time, although well-rated for G1, shows slightly greater dispersion and, for G2, a much lower mean score. We believe the significantly lower score for G2:P11 is due to the larger number of participants: More groups had to present their designs to the class and this takes more time overall. As planning for an activity that takes more than two hours may not be desirable, an alternative could be to have the students read about the dynamics of the activity offline, prior to the in-person session, or to have only a subset of the groups present their designs to the whole class. In sum, the design of the activity was well received among the students but it could be improved with adjustments to the time allocation or by reducing the number of groups that present their designs (which also helps in limiting how long the activity takes).

*Collaboration and discussion:* This section shows a high score for both P12 (G1=4.67, G2=4.24) and P13 (G1=4.83, G2=4.24), showing that collaboration and exchange of ideas was viewed in a positive light by the students.

*Commentary on the lower scores for G2 than G1:* For most of the questions the mean score for the responses from G2 are slightly (0.17 - 0.43) lower than those of G1. We believe this may be a result of the selection of students participating: G1 had only students that made an effort to participate in the activity, thus were already highly motivated to begin with, while G2 included most students in the class as the activity took place during a regular class session. To better understand the differences in opinions between these two groups, we further check which are the three questions with lower mean scores for G2 than G1: P6 (0.49 lower), P13 (0.59 lower) and P11 (1.36 lower, already discussed). A lower score for P6 (*I would like this type of activity to be included regularly throughout the semester*) could be a result of a lower internal motivation to participate or a result of the timing problem with this second group. A lower score for P13 (*The exchange of ideas between teams was valuable for enriching my proposal*) could be a result of the extra time that the group presentations took (which they thought was excessive).

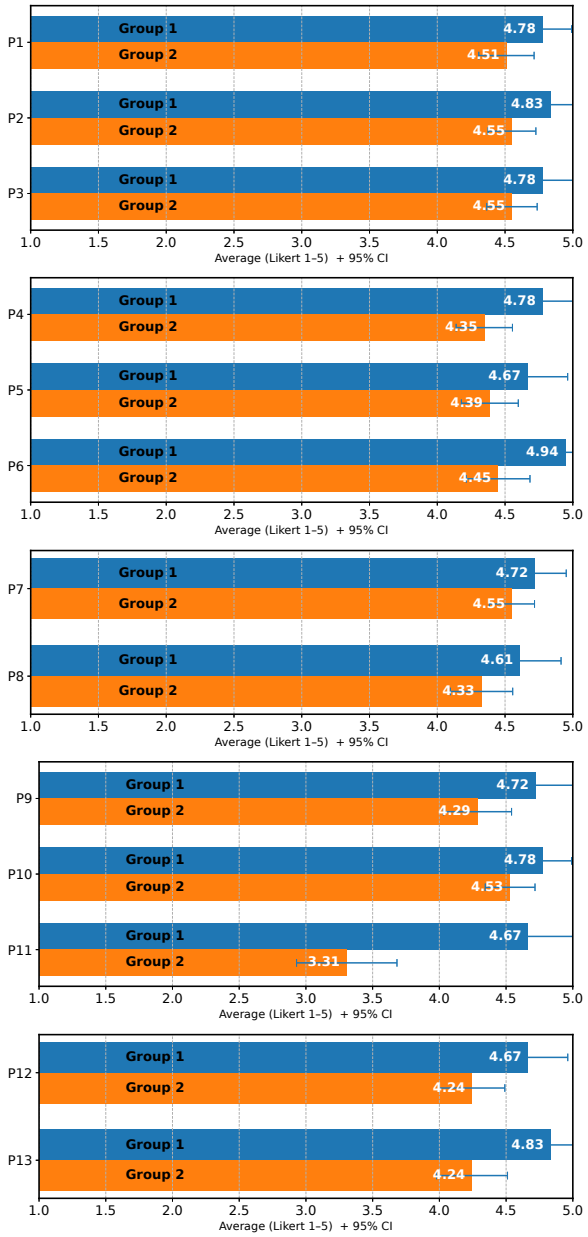


Fig. 1. P1-P3: Satisfaction and motivation; P4-P8: Utility for learning; P9-P11: Activity methodology; P12-P13: Collaboration and discussion.

## B. Open-ended questions

For P14, students in G1 and G2 emphasized the **hands-on, applied nature** of the activity and the opportunity to practice cloud architecture design in realistic contexts (e.g., "The ability to practice building our own architecture," "Designing an architecture that could be applied to a real-life system"). Across both groups, students also highlighted the clarity of the explanations and the instructor's guidance as key factors supporting a positive experience (e.g., "The professor explained [everything] very well," "The freedom of the work, but at the same time the professor's assistance"). Likewise, many responses valued the **collaborative dynamic** within teams, idea sharing, and learning from peers, reinforcing the quantitative results on Collaboration and Discussion (e.g., "Sharing ideas with my partner," "Working in pairs made it easier to exchange ideas and learn things I did not know"). In G2, students additionally noted the usefulness of **support materials** (examples, cheat sheet) as scaffolding to guide their design decisions.

TABLE IV
EVALUATION RESULTS BY CRITERIA.

| # | Criterion | G1 | G2 |
|---|---|---|---|
| C1 | Active individual participation | 95.53% | 97.16% |
| C2 | Collaboration and exchange of ideas | 95.55% | 96.21% |
| C3 | Justification of architectural changes | 87.20% | 88.24% |
| C4 | Design assumptions | 91.65% | 91.80% |
| C5 | Proposal of validations/metrics | 88.13% | 88.80% |
| C6 | Clarity and organization of the report | 92.20% | 93.00% |
| **Total score** | | **91.67%** | **92.44%** |

The analysis of the open-ended responses also identified **areas for improvement**. In both groups, the most frequent request was **more time** to complete the activity and reflect on service choices, consistent with the quantitative feedback related to time allocation (e.g., "More class time," "Only 20 minutes [to document their assumptions] seemed too little"). Students in both groups suggested **additional examples or prior use cases** to reduce initial uncertainty before starting the design work. Several responses proposed extending the activity beyond the classroom through **implementation in AWS** (e.g., "Using the architecture directly in AWS"). Finally, some comments suggested enhancing the learning closure via **more feedback** (e.g., more peer discussion), and, less frequently, adjusting the collaboration format (e.g., moving from pairs to larger teams) to enrich peer interaction.

### C. Grading rubric results

During the activity, the instructor and teaching assistant moved around the classroom, observing participants and applying the rubric while students worked. When each team presented their proposals the instructor observed and applied the rubric corresponding to that component. The report was graded post-activity. Thus, the evaluation was not limited to a final product, but captured both the learning process and the quality of the proposal. Table IV shows the evaluation results for both groups (per criteria averages, expressed as a percentage of the maximum grade for each criteria). Figure 2 shows one of the designs proposed by one team in G2.

A very high C1 score evidences students' commitment and initiative during the design process. The C2 component has the highest average score of all, suggesting that the teamwork format fostered healthy discussions and co-construction of solutions. Component C3 has the lower score average; although student performance in justifying their architectural modifications can be considered "good," there is room to strengthen their technical arguments behind the modified services and their relationship to non-functional requirements (e.g., performance, availability, cost). The teams formulated consistent assumptions, C4, suggesting that they understood the domain constraints and specifically incorporated them into the design. Metrics were mostly correctly identified (latency, throughput, etc.), C5, although in several cases they lacked operationalization (how to measure, with what tools, under what loads). The diagrams and reports were generally clear and well-structured, C6, suggesting that the students were able to adequately communicate their proposals.

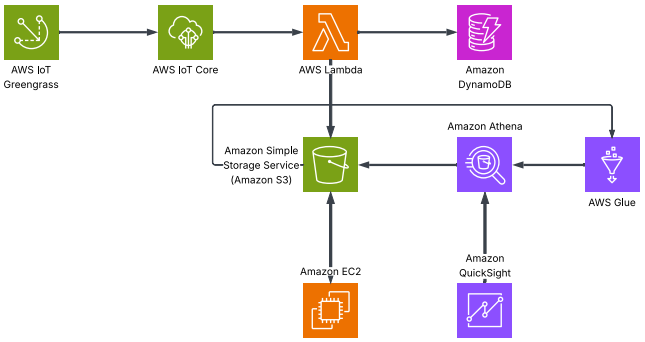


Fig. 2. One resulting design for an IoT architecture with monitoring and scalable data analysis capabilities.

## IV. DISCUSSION

*Future improvements:* Based on our results we propose some changes to the activity that could improve the student experience and learning process. In future runs, we can include a brief illustrative example where the instructor shows how a specific change in the architecture (for example, adding or removing a service) is justified based on non-functional goals. In addition, the dynamic of the activity can be explained to students prior to the session, possibly via an asynchronous offline reading task. It would also be helpful to include a short guide with key metrics (latency, throughput, etc.) and a brief explanation on their importance and collection. Allowing for larger groups with up to four students could improve the experience and reduce the presentation time. When possible, we should extend the activity by an additional 30 minutes.

*Limitations:* While our results point to our activity being useful for students, they come from two small runs, in the first of which the students voluntarily made an effort to participate, biasing the results. The small number (G1 & G2) and volunteer status (G1) limit generalizability and our results only serve as anecdotal evidence of the activity's usefulness. Regarding the synthetically generated sets of cloud components, we did not thoroughly validate their soundness. However, we did do several statistical validation tests that together with the qualitative evaluation of the designs submitted by the students suggest that the generated sets of components are at least realistic enough for the purposes of this activity. Finally, given that the sets of components are generated synthetically and assigned randomly to groups of students, it is possible for a group to receive a set that is much easier to integrate than another group. This means that using the activity as a graded (evaluative) course activity may not be a good idea as students may perceive an unfairness in the assignments.

## V. RELATED WORK

*Collaborative learning in software design and architecture:* In the past, the community has highlighted the importance of

adding collaborative learning experiences in Software Engineering education [20]. The DBCollab [21] tool was proposed to aid in collaborative database design. The *Collaborative Design and Build (CDB)* activity framework [22] spans the analysis, modeling and implementation phases of designing a software solution while fostering soft skills: collaboration, communication, problem-solving and critical thinking. While the modeling phase includes creating block diagrams, these are diagrams of the internal architecture (modules) and not of a multi-component distributed application. Capilla et al. [23] describe a decision-modeling tool that can be used in collaborative decision-making tasks and software design activities. In the future, it would be valuable to integrate frameworks and tools like these with our collaborative architecture design activity to help instructors when using it in the classroom.

*Teaching cloud computing:* As has been stressed by the ITiCSE WG on Cloud Computing [7]–[11], architectural design is an important component of Cloud Computing education. For this reason, several recent papers have targeted this computing subdomain [24]–[28]. Saligrama et al. [26] designed an undergraduate-level course to teach cloud infrastructure deployment as an applied software engineering practice. The course includes hands-on assignments using infrastructure-as-code tools so that students can learn to build cloud-native applications while reasoning about their elasticity, cost, and security models. Casale [27] described two hands-on activities to teach Performance Engineering using distributed/cloud architectures. Our proposed activity can fit well in courses like these, to help students learn how to properly justify their design choices when designing cloud (infrastructure) architectures while focusing on non-functional requirements such as performance and scalability.

*Unplugged classroom activities:* Unplugged classroom activities have been proposed as they encourage collaboration, promote equity in computing and aid in student understanding [17], [18]; e.g., a card game for teaching PDC concepts [29]. While the approach is not new [30], [31], these activities are not generalizable and need to be designed for specific computing topics. Our proposed activity expands the list of CS unplugged activities with a new proposal suitable for classes like Distributed Systems, Cloud Computing, Big Data, Software Engineering or elective topics in emerging areas like Edge computing.

## VI. Conclusion

We presented a collaborative activity to improve students cloud architecting skills. The activity is a collaborative design exercise where students work in teams to integrate a given set of cloud components into a single design that could solve a real use case. The lists of components given to the students are synthetically generated to mimic real cloud architectures from a recent public dataset. Our results (observational, questionnaire, assessment) from two small runs in an undergraduate *Distributed Systems and Cloud Computing* class show that students liked the activity, were highly engaged and applied critical thinking, collaboration and communication skills. We are releasing our tool, grading rubric and support material so that others can replicate the activity at their institutions.

## References


[1] E. Sayegh, "The Cloud Talent Drought Continues (And Is Even Larger Than You Thought) — forbes.com," https://www.forbes.com/sites/emilsayegh/2020/03/02/the-2020-cloud-talent-drought-is-even-larger-than-you-thought/, 2020, [Accessed 28-09-2025].

[2] V. Vladimirskiy, "Closing The Cloud Skills Gap Requires A Culture Of Continuous Upskilling — forbes.com," https://www.forbes.com/councils/forbestechcouncil/2024/05/28/closing-the-cloud-skills-gap-requires-a-culture-of-continuous-upskilling/, 2024, [Accessed 28-09-2025].

[3] D. Garn, "Is there still a cloud skills gap in 2025? — TechTarget — techtarget.com," https://www.techtarget.com/searchcloudcomputing/tip/Is-there-still-a-cloud-skills-gap, 2025, [Accessed 28-09-2025].

[4] A. Van Deursen, M. Aniche, J. Aué, R. Slag, M. De Jong, A. Nederlof, and E. Bouwers, "A collaborative approach to teaching software architecture," in *ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2017.

[5] H. P. Breivold and I. Crnkovic, "Cloud Computing education strategies," in *IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*, 2014.

[6] A. Kumar, R. Raj, S. Aly *et al.*, *Computer Science Curricula 2023*. Association for Computing Machinery, 2024.

[7] D. Foster, L. White, J. Adams *et al.*, "Cloud computing: Developing contemporary computer science curriculum for a cloud-first future," in *ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE) Companion*, 2018.

[8] J. Adams, B. Hainey, L. White *et al.*, "Cloud computing curriculum: Developing exemplar modules for general course inclusion," in *ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, 2020.

[9] J. Paterson, J. Adams, B. Hainey, and S. Nazir, "A repository of resources and exemplars for the cloud curriculum," in *Conference on Computing Education Practice (CEP)*, 2021.

[10] J. Paterson, J. Adams, L. White, A. Csizmadia *et al.*, "Designing dissemination and validation of a framework for teaching cloud fundamentals," in *ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, 2021.

[11] J. Paterson, J. Adams, D. Foster *et al.*, "Motivation and strategies for effective inclusion of cloud solution provider certifications in computing curricula," in *ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, 2022.

[12] S. Prasad, T. Estrada, S. Ghafoor *et al.*, "2020 NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing - Core topics for undergraduates, Version II-beta," 2020. [Online]. Available: http://tcpp.cs.gsu.edu/curriculum/

[13] C. Abad, E. Ortiz-Holguin, and E. Boza, "Have we reached consensus? An analysis of distributed systems syllabi," in *ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2021.

[14] C. Abad, A. Iosup, E. Boza, and E. Ortiz-Holguin, "An analysis of distributed systems syllabi with a focus on performance-related topics," in *ACM/SPEC International Conference on Performance Engineering (ICPE) Companion*, 2021.

[15] R. Caceffo, G. Gama, and R. Azevedo, "Exploring active learning approaches to computer science classes," in *ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2018.

[16] J. Alonso, "Amid AI Plagiarism, More Professors Turn to Handwritten Work — insidehighered.com," https://www.insidehighered.com/news/faculty-issues/curriculum/2025/06/17/amid-ai-plagiarism-more-professors-turn-handwritten-work, 2025, [Accessed 28-09-2025].

[17] S. Matthews, "PDCunplugged: A free repository of unplugged parallel distributed computing activities," in *International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020.

[18] M. Smith, S. Srivastava, D. Bunde *et al.*, "A visual unplugged activity to introduce PDC," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2025.

[19] S. Satija, C. Ye, R. Kosgi, A. Jain, R. Kankaria, Y. Chen, A. Arpaci-Dusseau, R. Arpaci-Dusseau, and Srinivasan, "Cloudscape: A study of storage services in modern cloud architectures," in *USENIX FAST*, 2025.

[20] R. Garcia, C. Treude, and A. Valentine, "Application of collaborative learning paradigms within software engineering education: A systematic mapping study," in *ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2024.

[21] V. Echeverria, R. Martinez-Maldonado, K. Chiluiza, and S. Buckingham Shum, "DBCollab: Automated feedback for face-to-face group database design," in *International Conference on Computers in Education (ICCE)*, 2017.

[22] G. Brieven, M. Moraes, D. Pawelczak, S. Vasilache, and B. Donnet, "Integrating soft skills training into your course through a collaborative activity," in *ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2025.

[23] R. Capilla, O. Zimmermann, C. Carrillo, and H. Astudillo, "Teaching students software architecture decision making," in *European Conference on Software Architecture*. Springer, 2020, pp. 231–246.

[24] C. Anglano, M. Canonico, and M. Guazzone, "Teaching cloud computing: Motivations, challenges and tools," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020.

[25] ——, "An educational toolkit for teaching cloud computing," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 4, 2021.

[26] A. Saligrama, C. Ho, B. Tripp, M. Abbott, and C. Kozyrakis, "Teaching cloud infrastructure and scalable application deployment in an undergraduate computer science program," in *ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2025.

[27] G. Casale, "Performance evaluation teaching in the age of cloud computing," *SIGMETRICS Perf. Eval. Rev.*, vol. 51, no. 2, 2023.

[28] A. Mokhtari, D. Rawls, T. Huynh *et al.*, "E2C: A visual simulator to reinforce education of heterogeneous computing systems," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2023.

[29] S. Srivastava and M. L. Smith, "Assessing parallel and distributed computing knowledge through a card game," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2025.

[30] W. Lloyd, "Exploring the byzantine generals problem with beginning computer science students," *ACM SIGCSE Bulletin*, vol. 26, no. 4, 1994.

[31] S. Andrianoff and D. Levine, "Role playing in an object-oriented world," in *ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2002.