Teaching Students Software Architecture Decision Making

Rafael Capilla[1(✉)], Olaf Zimmermann[2(✉)], Carlos Carrillo[3(✉)], and Hernán Astudillo[4(✉)]

[1] Rey Juan Carlos University, Madrid, Spain
rafael.capilla@urjc.es
[2] HSR Hochschule für Technik, Rapperswil, Switzerland
olaf.zimmermann@hsr.ch
[3] Technical University of Madrid, Madrid, Spain
carlos.carrillo@upm.es
[4] Universidad Técnica Federico Santa María, Santiago de Chile, Chile
hernan@inf.utfsm.cl

Abstract. Making the right decisions is challenging for architects on all levels of seniority. Less experienced architects in particular perceive the transition from design problems to their solutions as hard; it is not always clear how to find suitable concepts and technologies, how to compare alternatives, and how to build consensus. Lack of experience makes it difficult to train software engineering students in the identification, selection, and collective evaluation of design alternatives. Moreover, human factors such as cognitive bias make "soft" topics like architecture decisions rather hard to teach. To overcome these issues and let students gain the required experience, a Spanish University ran two experiments. Undergraduate computer science students assumed different roles in collaborative decision-making tasks and design activities. They used a novel decision-modeling tool to capture and challenge the relevant design decisions. This paper describes this new teaching setup and reports on lessons learned.

Keywords: Architectural knowledge · Collaborative decision making · Design decision · Design thinking · Reflection · Teaching software architecture

1 Introduction

The creativity of software architects may lead to different designs or solution architectures serving for the same purpose. Diagrams illustrating proven designs may be worth a thousand words [15]. Nevertheless, teaching novice architects and undergrad students in the practice of software architecture is not easy because of the plethora of design alternatives and possibilities to compose them. Thus, it is challenging to select adequate design solutions promoting well-known design principles; years of apprentice are required. The ISO/IEC 42010 standard[1] considers a software architecture as the result of a set of

[1] ISO/IEC/IEEE 42010 – Systems and software engineering – architecture description. Available at: https://www.iso.org/standard/50508.html.

A. Jansen et al. (Eds.): ECSA 2020, LNCS 12292, pp. 231–246, 2020.
https://doi.org/10.1007/978-3-030-58923-3_16

design decisions and first-class artifacts. An *Architecture Design Decision (ADD)* can be understood as a decision that addresses architecturally significant requirements and design problems. The tacit knowledge and expertise of software architects should be captured explicitly via their significant ADDs [35]; this is costly and challenging when adequate tools are missing [12]. Consequently, teaching novice designers on making and selecting the right decisions, evaluating the consequences of such selections and challenging the decisions made by others is still in its infancy.[2]

An additional challenge is that development teams tend to be decentralized and distributed more and more. This has raised new forms of collaboration, where several stakeholders take on different roles and often use collaborative modeling tools to arrive at adequate and accepted design solutions. This research explores behavior of software engineering students as novice software architects in different roles: how do they make, challenge and capture a set of ADDs and architectures in a collaborative way? The set of decisions captured serves as training material as reusable knowledge; the roles assumed by the students is supposed to enable and promote critical design thinking to produce decisions with better quality and architectures.

The remainder of the paper is structured as follows. Section 2 discusses related work and provides background information. Section 3 outlines the design of two experiments we run in a Spanish university to teach a collaborative decision-making practice. Section 4 outlines the results we obtained. Section 5 presents lessons learned and resulting insights for researchers, educators, and practitioners. Finally, Sect. 6 summarizes our conclusions and provides an outlook.

2 Background and Related Work

This section describes related background and similar studies used in our experience and around three different topics.

2.1 Teaching Design and Collaborative Decision-Making

Collaborative software engineering [18] gains more and more significant attention by software development teams, especially in cases of delocalization in distributed teams. One important task in the software development process is software modelling, essentially seen as a creative task used to yield the software architecture of a system. In any architecting process, the chief architect must discuss, evaluate and deliberate with solution architects, senior and junior designers about the resultant design.

This deliberative task, adapted to different project contexts (e.g. open source, global development, agile projects, etc.) should capture the significant architecture design decisions. Bang et al. [3] report the insights related to six different roles of software architects during collaborative software design tasks in a distributed environment. Some teaching experiences about software architecture [17, 24] reflect the importance of architecture

[2] The IBM e-business reference architecture taught about the importance of architectural decisions; decision capturing was a recommended practice at IBM since, at least, 1998.

design decisions as part as the architect's soft skills and how to explain the technical decisions to non-technical staff and also how students have to wrestle with the complexity of design decisions.

Collaborative approaches teaching software architecture [8] evaluate the effect of design decisions in a collaborative software architecture course in open source projects, or using ASQ, a collaborative web-based platform [36] which allows tracking and recording real-time students' interactions and thinking. Other experiences report about *Group Decision Making (GDM)* practices in software architecture [28] showing that brainstorming is a preferred activity for team members. The authors of [20] describe the DVIA approach to provide design verbal intervention analysis to capture GDM activities; they also categorized the variety of decision topics. However, the lack of tool support is an important barrier to assist participants in the decision-making process [34].

2.2 Architectural Knowledge Research

The recent evolution in AKM tools has led to more collaborative approaches that let architects not only capture and share ADDs, but also discuss them collaboratively – supported by extended capabilities aimed to facilitate collaborative decision-making [5]. Research tools like RAT, AREL, SAW, and ADvISE (among others) offer sharing and voting mechanisms to deliberate around the design alternatives and facilitate the selection of the best design choices, many times complicated by the fact to achieve a consensus [35]. However, the dichotomy between AKM and software modeling tools still does not help to embed the design rationale and their decisions in software architecture modeling tasks [4]. ADMentor [39] helps novice architects to discriminate between the problem and solution spaces capturing the key design decisions and providing a design solution for the decisions chosen. A Markdown-based template and supporting tools are presented in [14].

2.3 Design Thinking, Reflection and Reasoning

Critical thinking can enhance problem-solving abilities, not only for learning programming concepts [13, 37], but also to acquire design skills and achieve a clear understanding of the problem and solution spaces. Producing to high quality decisions is not easy; sometimes reflective approaches [23] such as the Mind1-2 model are necessary [22] to challenge (Mind 2) the decisions made by others (Mind 1). This reflective thinking (i.e. reflection in action theory [24, 25]) and reasoning approach have also been highlighted in [28], considered a learning process to bring the tacit knowledge explicit [9]. As during the design activity design problems must be solved, designers must consider different options as design alternatives when they explore the solution space [2, 11] and reduce the cognitive bias.

Users can experience the reflective activity in different development context. Reflection, as a conversation during the design thinking activity, can be integrated with agile practices [29] during software constructions, such as stated in [2]. In [19] the authors explore how students increase awareness and reflective practices during learning activities. The study reports that the Critical Incident Technique (CIT) used in a software development course promotes critical reflection and communication skills. Other approaches

describe how developers discuss rationale in open source software (OSS) projects [1] to understand how rationale is discussed using IRC channels. One recent experience [10] reports the results of teaching reflection in software engineering students as a computer-supported collaborative learning (CSCL) activity; software engineering teachers and students reflected during UML modeling sessions.

During the reflective modeling tasks, the participants increased their awareness on modeling assumptions and sharing knowledge to confront multiple perspectives using video-taped sessions. Moreover, as software design is recognized as an outcome of design reasoning, Tang et al. [33] argue that decisions made collectively are based on some rationale, but the reasoning and argumentation of such decisions may not be explicitly stated. They suggest an approach to identify the design issues [31] in an experiment with students and professionals using reminder cards to investigate the use of design reasoning explicitly. In an experiment with students and professionals reported in [32], the authors studied the differences in design reasoning between both groups and the effort spent in decision-making activities. Another study reports on decision making practices and the authors identify their gaps and mention a lack of improvement and no new practices [21].

Finally, one recent study investigates a rationale management case study in an agile context to uncover what matters to students when they need to integrate decision documentation into Scrum projects [27]. The authors run an experiment with 400 participants grouped in 82 teams to analyze the effects of a lightweight architectural decision capture and understand how students capture decision alternatives and their rationale and which types of decisions are more important when teaching agile methods. Part of the results show that the majority of the students do capture multiple design *alternatives*, but struggle to capture *rationale*. However, implementing continuous reflection in software tools is still challenging but feasible at least to a certain extent, such as described in the prototype discussed in [26].

3 Experimental Design

As there still not many students showing the impact in teaching architecture with the capture of design decisions and rationale and also how students reflect and think about different design choices, this study investigates the quality and effectiveness of teaching design thinking [30] in a software architecture course. We were interested in evaluation the positive impact of reflections in design thinking, were students challenge the design decisions made by others, and to what extent students acting as software modelers perceive value in having design decisions captured before the modeling tasks in order to ease the design activity.

The research questions that guided this study are:

RQ1: *How does consensus influence the decision-making process in collaborative environments?*
RQ2: *Do reflective practices like the Mind1-2 model [Razavian2016] have a positive influence on design thinking?*
RQ3: *Do the decisions captured have any positive effect on software architecture modeling tasks?*

3.1 Course Description

The learning objectives of the target course on software architecture (third year of undergraduate studies on Software Engineering) at the Rey Juan Carlos University of Madrid (Spain) encompasses common topics like design patterns, architecture styles, design principles and architecture evaluation methods like Architecture Trade-off Analysis Method (ATAM). Furthermore, we include a lecture on ADDs to train students on their importance of and the difficulty to capture the significant decisions and evaluate the design alternatives. Finally, we highlight the importance of reflection as a technique to challenge design alternatives when dealing with incomplete information and uncertainty (for instance, design patterns and/or technology selection). The teaching method follows a classical approach using slides to teach the lessons and minutes for discussion at the end of each lesson. The students need to pass two practical assignments and form groups of five to six persons, which is easy to achieve because the average number of students enrolled in the course is often more than 50.

3.2 Course Selection and Participants

From the course, we selected the following participants from two different years. We ran two experiments in 2016/17 and 2017/18, with 65 and 58 students respectively. The age of the participants enrolled in the course 2016/17 was between 19 and 29 years (plus one student over 30), while the age of the subjects in the course 2017/18 was between 19 and 29. However, most participants were between 19 and 24 years old. Only some of them have a few months of professional experience in software companies.

3.3 Training and Laboratory Activities

Before the experiment, students had around three weeks of software architecture lectures on concepts, design patterns and architectural design decisions; we used several examples to train them in modern software architecture practice and emphasized the importance of capturing the significant architecturally design decisions. The laboratory sessions for each of the two experiments took 4 weeks in sessions of 4 h per week. Additionally, the students spent additional time outside the laboratory to complete the weekly tasks. We established that each week students had to produce one iteration of the software architecture for the target system, including smaller sub-iterations according to the number and complexity of the software requirements and design problems.

3.4 Groups and Tasks

We organized the students in 10 teams composed by five to six participants each, with the following roles (see Fig. 1).

- *Two senior architects* are in charge of making architectural design decisions and capturing them, identify design problems, make and capture architecture design decisions;

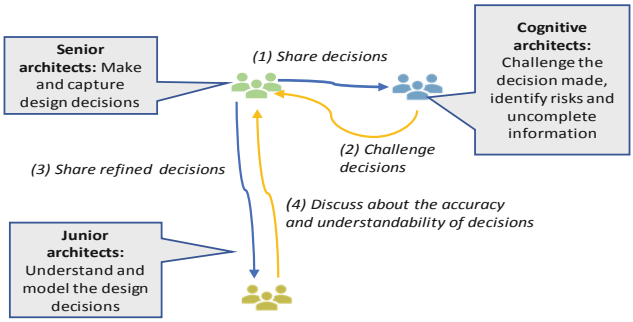

Fig. 1. Relationships between participants in collaborative decision making and design thinking

- *Two cognitive architects* challenge the decisions captured by senior architects and raise any concern when risks or incomplete information appear;
- *One or two junior architects* are in charge of understand and model the decisions captured providing a solution architecture.

Each team was composed of two senior architects, two cognitive architects, and one to two junior architects. We selected the most experienced students (i.e., those who had industry experience) as cognitive and senior architects while those students with no experience acted as junior architects. Because it was difficult to find enough cognitive and senior architects, we balanced the groups to have at least one senior and one cognitive architect. The students using the different roles interacted during 2 + 2 laboratory hours weekly and via Skype to discuss and annotate the changes in the decisions and architecture documentation using Google drive docs.

In addition, we let the groups to use the ADMentor[3] tool [39] for modeling the problem space and solution space and we limited the granularity of the decisions up to UML classes to avoid an excessive number of small decisions. Finally, all groups captured the decisions using any of the three different templates provided: a minimalistic template including five attributes (i.e., useful for decisions captured in agile projects), a medium template comprising seven attributes and a longer template comprising eleven items.

4 Results

During the first experiment, we got responses from ten groups and we discarded one group because they did not provide significant results that could be used to evaluate the impact of design decisions in architecture modeling tasks. In the replica experiment, we received valid responses from ten groups, so in both experiments we had almost the same number of participants. The collaborative decision-making activities happened along four weeks, and we collected the following results.

[3] https://www.ifs.hsr.ch/ADMentor-Tool.13201.0.html?&L=4.

4.1 Collaborative Decision-Making

Once the senior architects made and captured the key design decisions during each iteration, they interacted collaboratively with the cognitive architects to produce better quality decisions. As we show in Fig. 2, cognitive architects from all groups challenged a number of 110 decisions during the deliberative tasks with senior architects. RQ1: *How does consensus influence the decision-making process in collaborative environments?*

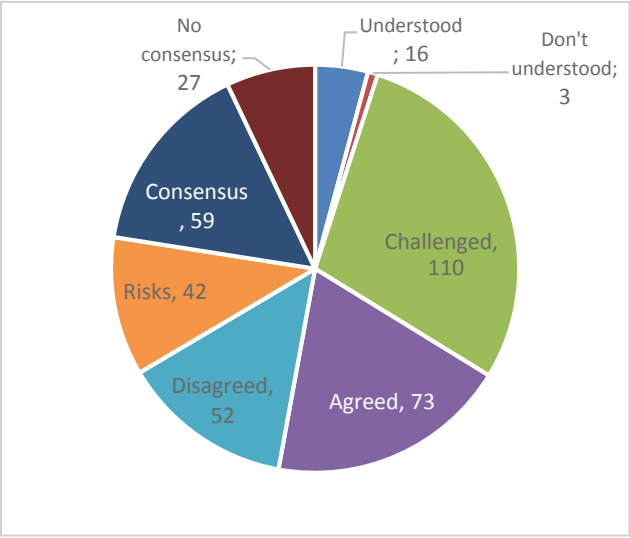

Fig. 2. Activity of senior and cognitive architects (Mind1-2 tasks) regarding the decision challenges, risks stated and consensus between decisions.

From the data obtained from the discussions between cognitive and senior architects, they agreed in 73 decisions and disagreed in 52 (please note that we included here decisions that were challenged and non-challenged). Finally, they achieved a consensus in 59 cases and no consensus in 27 times. In addition, the cognitive architects stated the appearance of risks in 42 decisions, as potentially critical decisions or caused by uncompleted information during the decision-making activity.

The distribution of the percentages during the interactions between senior and cognitive architects prove the utility of the Mind1-2 model and the importance for challenging the decisions made by others and the intention to arrive to a consensus during discussions and agree on as much decisions as possible in order to arrive to a solution architecture. Therefore, building a consensus in architecture decision-making seems to influence positively the speed to arrive to agreements, particularly for critical decisions.

RQ2: *Do reflective practices like the Mind1-2 model have a positive influence on design thinking?* Regarding the second research question, we can strongly affirm reflection has a positive impact in architecting practices and more specifically in design thinking activities. Figure 3 shows the results of the cognitive activity and discussions, which are quite similar in both experiments. In Fig. 3, the X axis represents the different states

about the decisions as results of the discussion between senior and cognitive architects while in the Y axis we show how many decisions in both experiments where challenged, agreed, understood or needed a consensus.

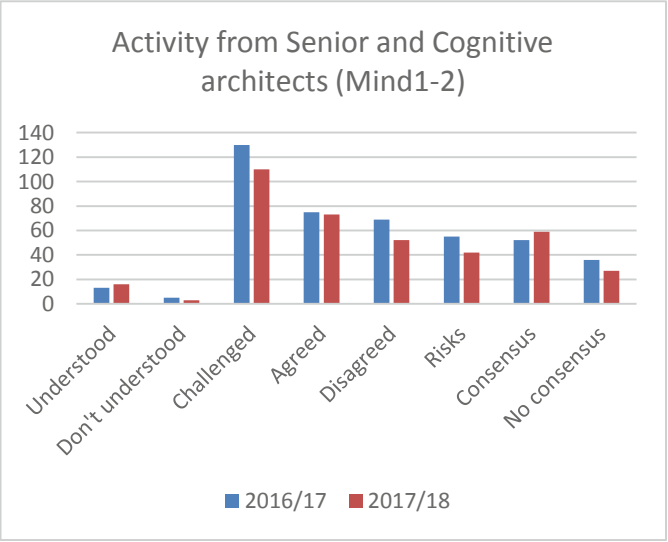

Fig. 3. Activity of senior and cognitive architects (Mind1-2 tasks) regarding the decision challenges, risks stated and consensus.

The results from the collaborative decision-making activity between both roles was very fruitful, and based on the number of decision challenged we can say that the Mind1-2 model [22] was proven better in yielding higher quality decisions than if senior architects make decisions alone without the criticism of the cognitive architects. In the replica experiment we got a bit better results in most cases than in the first experiment, maybe because we trained the students better in the second year, but the differences in general terms are small and the replica confirm our initial results.

4.2 Collaborative Analysis and Modelling Tasks

Figure 4 compares the activity of interactions between senior and junior architects in order to answer research question RQ3.

RQ3: *Do the decisions captured have any positive effect on software architecture modeling tasks?* One of the most interesting outcomes derived from both experiments is the satisfaction of junior architects as software modelers using the decisions made by others to depict the software architecture. During the interactions between them, junior architects widely used design patterns and the decisions shared by the senior architects were useful to model the software architecture. Figure 4 describes the activity of the interactions performed by junior architects. For instance, we can observe from the first experiment that junior architects believed decisions were useful to model the software

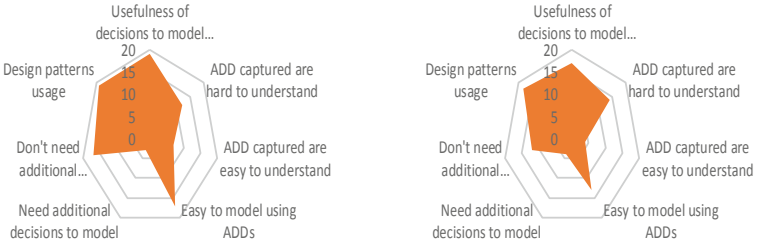

Fig. 4. Activities performed by junior architects in the experiment 2016/17 (left side) and in 2017/18 (right side).

architecture and, in most cases, they did not need an additional explanation. Also, in many cases junior architects didn't need additional decisions to model the solution. In addition, the subjects used design patterns and they experienced that software architectures were easier to model using ADD descriptions.

Similar results we obtained from the replica experiment run during 2017/18, as show in Fig. 4. In the replica, more junior architects than in the first experiment thought they did not need additional decisions to model the software architecture. In some cases, some of the subjects perceived the design were not so easy to model using the ADDs because some of the decisions had been not described or captured properly and hence were hard to use during the modeling tasks.

However, only two more subjects in the replica experiment thought decisions were hard to use (14 subjects in the replica versus twelve in the first experiment). Related to this, only three subjects in the first experiment thought more decisions were required, and four subjects in the replica thought the same. Although some decisions should be captured better, most of them were understood by the software modelers, and in only a few cases additional decisions were needed.

Moreover, during the sessions with the students we observed their activity and discussions but we did not participate in any form of action research as we only had minimal interventions about questions in order to avoid bias during the decision-making activity, as any kind of help could interfere in the decisions chosen by the senior architects. In addition, we run a questionnaire asking all participants from the three different roles a set of questions regarding about their activities performed, and with respect to the activity of junior architects and the interactions with senior architects. Table 1 shows the results from both experiments.

We got a maximum of 19 and 18 responses out from 20 and 19 junior architects belonging to the experiments 2016/17 and 2017/18 respectively. As we can see in Table 1, the results from the second experiment confirm the results from the initial one. Most subjects thought that decisions are useful to model the software architecture. However, 12–14 of junior architects thought also that the decisions descriptions provided by the senior architects were hard to understand for modeling an architectural solution, while only 4–7 thought the opposite. In general, most junior architects felt architectures are easier to model if they have design decisions, but the results of the replica show a higher disagreement at this point. In addition, in both experiments only three and four students respectively thought they would need additional decisions to model the solution

Table 1. Results from the questionnaire evaluating the activity and interactions of junior architects.

Activity of junior architects	Experiment 2016/17	Experiment 2017/18
Decisions are useful to model the SA	19	17
ADD descriptions are hard to model the SA	12	14
ADD descriptions are not hard to model the SA	7	4
Architecture is easy to model using ADDs	17	13
Need of additional decisions to model the solution architecture	3	4
Don't need additional decisions to model the architecture	17	12
Use of design patterns	19	18

architecture, while 17 subjects in 2016/17 thought they did not need additional decisions. Again, there are some differences with junior architects in the replica experiment as only twelve thought they do not need additional decisions. Finally, most students in both experiments agreed that using design patterns to define a solution architecture was quite useful.

4.3 Effort Spent in Cognitive and Modelling Tasks

In order to compare the effort spent in hours by the three different roles in the experiment, we illustrate in Fig. 5 the results (expressed in hours) from the replica. We observed that most groups with higher decision-making (including the capturing effort of the ADDs) and reflective activities required less modeling effort (i.e. groups G1, G2, G3, G6, G8, G9, G10) while only groups G4, G5 and G7 spent more time during modeling activities and maybe caused because the design thinking and reflective effort was low.

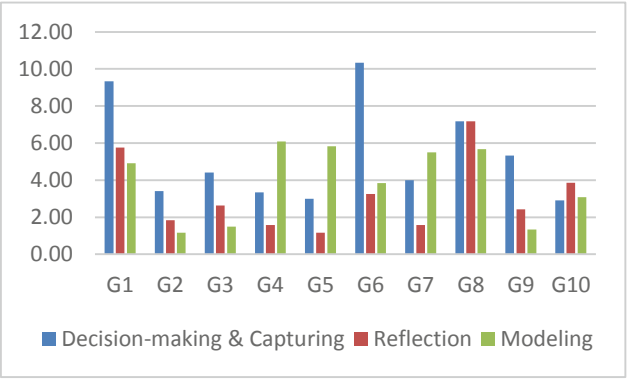

Fig. 5. Comparison of cognitive and modeling effort of groups from the replica experiment.

Some distorting results belong to group G6, which exhibits a high decision-making effort probably caused because they captured more design alternatives than the rest of the groups, and group G8, where team members spent the same time in decision-making and reflective activities. One reason for this (i.e. assuming they measured the effort correctly) might be that a significant number of discussions were devoted because unclear decisions, disagreements or high number of risks identified by the cognitive architects.

Finally, both experiments exhibited a similar pattern where junior architects classified more decisions as "hard to understand" easy to understand and model". Although junior architects perceived that having decisions eases the modeling tasks, the description of many decisions require further explanation, which added additional discussion cycles during the interactions with senior architects. As the subjects stated that in majority of the interactions, they did not need additional decisions to model the solution, it seems that it was just a matter of improving the descriptions captured in the templates.

Moreover, the interactions between the junior architects of each group to depict the architectural sketches [16] revealed that in many cases they performed short iterations until they model a bunch of design decisions. Fifteen groups out of 20 used Google drive to upload architectural sketches and documents with junior and senior architects annotating comments collaboratively to deliberate about the resultant architectures, whereas only five groups used a collaborative modeling tool.

4.4 Effort in Reflective Tasks

The importance of reflective effort compared to the time spent in decision-making and capturing the decisions is highly relevant to the quality of the decisions made and of the solution architecture as well. We compared this reflective effort in both experiments and we got the following results. Figure 6 (left) shows the effort (in hours) spent by the different teams in making and capturing decisions as well as the extra effort when they reflected about the decisions.

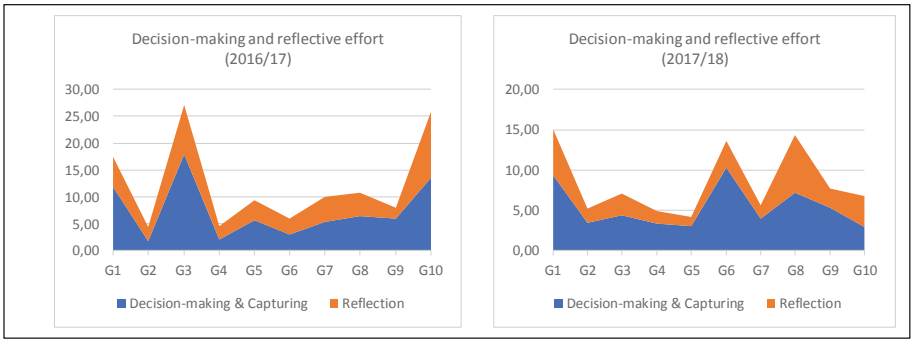

Fig. 6. Effort spent in making decisions and during reflective tasks in both experiments.

As shown in Fig. 6 (left), groups G2 and G4 spent little effort in making and capturing the decisions while G1, G3 and G10 spent much effort than the others. The rest of the

groups spent similar time in this activity. The reflective activity indicates the extra effort required for producing better quality decisions. Apart from groups G3 and G10, which present distorting numbers, the rest of the groups employed a reasonable time in reflection compared to the previous decision-making activity. Only G2 and G4 groups required more time in reflection than making decisions but the differences are rather small.

As confirmation in the replica shown in Fig. 6 (right), reflective effort is an extra effort for all groups, not just in addition to making decisions but because the students were better trained in decision-making than in the first experiment, and they need significantly less effort for making and reflecting decisions (see effort the Y axis). In this case only groups G1, G6 and G8 required much more effort than the others and only G10 spent bit more effort in reflecting than in making decisions.

5 Discussion

The major results derived from both experiences are the following:

(i) The groups using the ADMentor tool understood the distinction of the problem and solution space much better, while senior architects identified design problems and design alternatives more accurately.
(ii) The quality of the architectures backed by high-quality design decisions led to an adequate solution architecture with the desired qualities.
(iii) Groups using ADMentor arrived at "better" architectures as they identified the design problems more clearly.
(iv) The investment in reflective tasks paid off and produced "good" quality decisions as outcome of the discussions between senior and cognitive architects.
(v) In most cases, the decisions were useful to model the design solutions; in some other cases, the description of the decisions was hard to understand by the junior architects.
(vi) Sometimes a decision might not be fully clear to the junior architects and require additional clarification; hence, more interactions between junior and senior architects happened and helped to clarify the design issues; this stresses the importance of collaborative group decision-making.

It is difficult to say how results would look like if we replicated the experiment with more experience practitioners from industry: In our study, we assigned multiple roles to the subjects; in companies, this might not be realistic as the role depends on the years of experience and the project contexts. However, the effort spent in decision-making and reflective tasks might be more accurate (as well as the quality of the architectures produced). Moreover, the experience seems difficult to replicate in multiple real projects as the timelines and architecture iterations on these projects typically differ a lot [15]. Another factor we did not study is the granularity of the design decisions as we focused on the decision-making as well as reflective and collaborative tasks. Nevertheless, we limited the size of the decisions to the class level (i.e. in the sense of object-oriented classes) and dependencies between such classes in object-oriented programs; we did not capture smaller decisions such as the creation of an attribute or a method. In addition, we did not

discuss the relevance of the decisions and the quality of the resulting architectures; the groups that spent more time in reflective tasks and discussions produced decisions with better quality (i.e. better described and argued). The quality of the final architectures was difficult to observe as we did not implement any system or evaluate quality properties to test how good the architecture is. However, we carried out interviews with the teams every week to mitigate the risk for having architectures not addressing the design problems. Our approach complements architecture design methods and techniques such as ADD [6] and the Architecture Tradeoff Analysis Method (ATAM) [7]. ADD in its latest version is also decision-centric but does not model these decisions explicitly; it rather collects related guidance informally. ATAM is an evaluation and review technique centered on desired quality attributes; it also can play a role in forward engineering. ATAM investigates architectural decisions made when exploring risks, sensitivity and tradeoff points. Hence, ATAM can also benefit from the explicit modeling capabilities of our approach and the ADMentor tool. In summary, one can view the concepts and their implementation presented in this paper as a sub-step or micro-methodology [38] for ADD and ATAM. Compared to previous related works and regarding the collaborative aspects, we investigated the interactions of the team members for the between different roles and for the same role. For instance, we studied how the students in the role as senior software architects reason to make decisions and how students acting as cognitive architects challenge and reflect on the decisions captured. However, we did not include such qualitative analysis in this paper due to space constraints. However, we did show the benefits of reflective practices in architecture decision-making using the Mind1-2 model. Nevertheless, a deeper analysis to examine the quality of the decisions after reflection is required. Moreover, we noticed that the students took some time until they learned how to reflect and challenge the decisions made; in the second and third iterations of the architecture development process, team members applied these reflective practices more commonly. We did not continue with the same experiment after 2018 as during 2019 and 2020 we moved and compared other architectural knowledge capturing approaches.

6 Conclusion

In this paper, we reported our experiences with teaching software architecture decision making, and we highlighted the importance of collaborative decision-making to produce more accurate and complete design decisions, by reducing the cognitive bias and anchoring through reflections using the Mind1-2 model. We summarize the main outcomes of our work as: (i) Adopting different roles when performing collaborative tasks such as challenging decisions improved the quality of the architectures. (ii) A significant number of decisions resulted from the agreement between the different stakeholders during the interactive decision-making process. (iii) The decisions captured using the Mind1-2 model exhibit better quality than those produced without reflection. To mitigate the threat to integrity that arises of not having test and control groups to evaluate the experimental design, we compared the resulting architectures with models produced in previous instances of the course, where subjects did not have access to captured decisions. Stating the risks of the decisions reduces the bias for making suboptimal decisions. In the future, we plan to explore voting systems to arrive to a faster consensus, and to explore cognitive

activity in agile projects where the time for decision-making is limited and very lean documentation templates are applied. Furthermore, we will investigate a better form of representing decisions to avoid the problem that decisions are hard to understand or that additional decisions are required to model a design solution.

Acknowledgements. H. Astudillo's work was partially funded by grant ANID PIA/APOYO AFB180002. O. Zimmermann's work was partially funded by the Hasler Foundation (project number: 19083).

References

1. Alkadhi, R., Nonnenmacher, M., Guzman, E., Bruegge, B.: How do developers discuss rationale? In: International Conference on Software Analysis, Evolution and Reengineering (SANER 2018). IEEE DL (2018)
2. Babb, J., Hoda, R., Norbjerg, J.: Embedding Reflection and Learning into Agile Software Development (2014)
3. Bang, J.Y., Krka, I., Medvidovic, N., Kulkarni, N.N., Padmanabhuni, S.: How software architects collaborate: insights from collaborative software design in practice. In: 6th International Workshop on Cooperative and Human Aspects of Software Engineering, pp. 41–48. IEEE (2013)
4. Capilla, R.: Embedded design rationale in software architecture. In: Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, pp. 305–308. IEEE DL (2009)
5. Capilla, R., Jansen, A., Tang, A., Avgeriou, P., Ali Babar, M.: A 10 years of software architecture knowledge management: practice and future. J. Syst. Softw. **116**, 191–205 (2016)
6. Cervantes, H., Kazman, R.: Designing Software Architectures: A Practical Approach. SEI Series in Software Engineering. Addison-Wesley, Boston (2016)
7. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. SEI Series in Software Engineering. Addison-Wesley, Boston (2001)
8. van Deursen, A., et al.: A collaborative approach to teaching software architecture. In: ACM SIGCSE Technical Symposium on Computer Science Education SIGCSE, pp. 591–596. ACM (2017)
9. Dingsoyr, T., Lago, P., van Vliet, H.: Rationale promotes learning about architectural knowledge. In: 8th International Workshop on Learning Software Organizations (LSO), Rio de Janeiro, Brazil. ACM (2006)
10. Dittmar, A., Forbrig, P.: A case study on supporting teachers' collective reflection in higher education. In: 36th European Conference on Cognitive Ergonomics ECCE 2018, pp. 4:1–4:8. ACM DL (2018)
11. Dorst, K.: Design problems and design paradoxes. Des. Issues **22**, 4–17 (2006)
12. Hohpe, G., Ozkaya, I., Zdun, U., Zimmermann, O.: The software architect's role in the digital age. IEEE Softw. **33**(6), 30–39 (2016)
13. Hwang, W.-Y., Shadiev, R., Wang, C.-Y., Huang, Z.-H.: A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. Comput. Educ. **58**(4), 1267–1281 (2012)
14. Kopp, O., Armbruster, A., Zimmermann, O.: Markdown architectural records: format and tool support. In: 10th Central European Workshop on Services and their Composition, pp. 55–62 (2018)

15. Larkin, J.H., Simon, H.A.: Why a diagram is (sometimes) worth ten thousand words. Cogn. Sci. **11**(1), 65–99 (1987)
16. Mangano, N., LaToza, T.D., Petre, M., van der Hoek, A.: How software designers interact with sketches at the whiteboard. IEEE Trans. Softw. Eng. **41**(2), 135–156 (2015)
17. Männistö, T., Savolainen, J., Myllärniemi, V.: Teaching software architecture design. In: Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), pp. 117–124. IEEE DL (2008)
18. Mistrík, I., van der Hoek, A., Grundy, J., Whitehead, J.: Collaborative Software Engineering. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-10294-3
19. Nylén, A., Isomöttönen, V.: Exploring the critical incident technique to encourage reflection during project-based learning. In: Koli Calling, pp. 88–97. ACM DL (2017)
20. Pedraza-Garcia, G., Astudillo, H., Correal, D.: DVIA: understanding how software architects make decisions in design meetings. In: ECSA Workshops 2015, pp. 51:1–51:7. ACM DL (2015)
21. Razavian, M., Paech, V., Tang, A.: Empirical research for software architecture decision making. J. Syst. Softw. **149**(3), 360–381 (2018)
22. Razavian, M., Tang, A., Capilla, R., Lago, P.: In two minds: how reflections influence software design thinking. J. Softw. Evol. Process **28**(6), 394–426 (2016)
23. Razavian, M., Tang, A., Capilla, R., Lago, P.: Reflective approach for software design decision making. In: Qualitative Reasoning About Software Architectures, pp. 19–26. IEEE DL (2016)
24. Rupakheti, C.R., Chenoweth, S.V.: Teaching software architecture to undergraduate students: an experience report. In: 37th IEEE/ACM International Conference on Software Engineering, vol. 2, pp. 445–454. IEEE CS (2015)
25. Schön, D.A.: The Reflective Practitioner: How Professionals Think in Action. Basic Books, Nueva York (1983)
26. Schoormann, T., Hofer, J., Knackstedt, R.: Software tools for supporting reflection in design thinking projects. In: 53rd Hawaii International Conference on System Sciences (HICSS 2020), ScholarSpace, pp. 1–10 (2020)
27. Schubanz, M., Lewerentz, C.: What matters to students – a rationale management case study in agile software development. In: 17. Workshops "Software Engineering im Unterricht der Hochschulen (SEUH 2020)", CEUR Workshop Proceedings, pp. 17–26 (2020)
28. Smrithi Rekha, V., Muccini, H.: Group decision-making in software architecture: a study on industrial practices. Inf. Softw. Technol. **101**, 51–63 (2018)
29. Talby, D., Hazzan, O., Dubinsky, Y., Keren, A.: Reflections on reflection in agile software development. In: Agile Conference, pp. 11–112. IEEE (2006)
30. Tang, A., Aleti, A., Burge, J., van Vliet, H.: What makes software design effective? Des. Stud. **31**, 614–640 (2010)
31. Tang, A., Lau, M.F.: Software architecture review by association. J. Syst. Softw. **88**(2), 87–101 (2014)
32. Tang, A., van Vliet, H.: Software designers satisfice. In: Weyns, D., Mirandola, R., Crnkovic, I. (eds.) ECSA 2015. LNCS, vol. 9278, pp. 105–120. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23727-5_9
33. Tang, A., Bex, F., Schriek, C., van der Werf, J.M.E.M.: Improving software design reasoning - a reminder card approach. J. Syst. Softw. **144**, 22–40 (2018)
34. Tofan, D., Galster, M., Avgeriou, P., Schuitema, W.: Past and future of software architectural decisions - a systematic mapping study. Inf. Softw. Technol. **56**(8), 850–872 (2014)
35. Tofan, D., Galster, M., Lytra, I., Avgeriou, P., Zdun, U., Fouche, M.-A., de Boer, R.C., Solms, F.: Empirical evaluation of a process to increase consensus in group architectural decision making. Inf. Softw. Technol. **72**, 31–47 (2016)

36. Triglianos, V., Pautasso, C., Bozzon, A., Hauff, C.: Inferring student attention with ASQ. In: Verbert, K., Sharples, M., Klobučar, T. (eds.) EC-TEL 2016. LNCS, vol. 9891, pp. 306–320. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45153-4_23
37. Wachenchauzer, R.: Work in progress – promoting critical thinking while learning programming language concepts and paradigms. In: Proceedings of IEEE International Conference on Frontiers in Education, Savannah, GA, USA, pp. 13–14 (2004)
38. Zimmermann, O., Koehler, J., Leymann, F.: Architectural decision models as micro-methodology for service-oriented analysis and design. In: Lübke, D. (ed.) Proceedings of the Workshop on Software Engineering Methods for Service-oriented Architecture 2007 (SEMSOA 2007), Hannover, Germany, vol. 244. CEUR-WS.org (2007)
39. Zimmermann, O., Wegmann, L., Koziolek, H., Goldschmidt, T.: Architectural decision guidance across projects. In: Proceedings of IEEE/IFIP WICSA (2015)