Using Automated Prompts for Student Reflection on Computer Security Concepts

Hui Chen
CUNY Brooklyn College
Brooklyn, NY, U.S.A.
hui.chen@brooklyn.cuny.edu

Agnieszka Ciborowska
Virginia Commonwealth University
Richmond, VA, U.S.A.
ciborowskaa@vcu.edu

Kostadin Damevski
Virginia Commonwealth University
Richmond, VA, U.S.A.
kdamevski@vcu.edu

ABSTRACT

Reflection is known to be an effective means to improve students' learning. In this paper, we aim to foster meaningful reflection via prompts in computer science courses with a significant practical, software development component. To this end we develop an instructional strategy and system that automatically delivers prompts to students based on their commits in a source code repository. The system allows for prompts that instigate reflection in students to be timely with respect to students' work, and delivered automatically, thus easily scaling up the strategy.

In this paper, we describe the design of a rule-based prompt delivery system, including a list of security related reflection prompts. We collect preliminary evidence for the reflection strategy in a course targeting mobile development. The evaluation provides evidence that such a system can help realize a reflection-in-action instructional strategy at scale and improve students' learning.

CCS CONCEPTS

• **Applied computing** → **Education**; **Computer-assisted instruction**; • **Computing methodologies** → *Artificial intelligence*;

KEYWORDS

reflection, reflection prompt, automated reflection

ACM Reference Format:
Hui Chen, Agnieszka Ciborowska, and Kostadin Damevski. 2019. Using Automated Prompts for Student Reflection on Computer Security Concepts. In *Proceedings of ACM ITiCSE conference (ITiCSE'19).* ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Drawn from experiential and constructivist view of learning, reflection is a process where a student contemplates and learns from current or past experiences. Instructors have strived to instigate reflection in various fields of professional practice and education for multiple decades [18], with numerous studies showing that reflection is beneficial to learning in many areas and disciplines [8]. In particular are its benefits to system design [2], including computer programming and software engineering [4–6, 22, 26]. While reflection is a diverse area of study, the focus of this paper is prompt-based reflection, where students are asked to reflect based on questions or directions provided by the instructor. The literature states that the learning outcomes of prompt-based reflection are strongly dependent on: 1) the quality of the prompts; 2) the timing of their delivery [7, 12]. Both of these characteristics are contingent on connecting the reflective prompts to individual student progress. However, *instructors closely monitoring each student's progress on numerous projects and assignments does not scale and is not realistic in most computer science programs.*

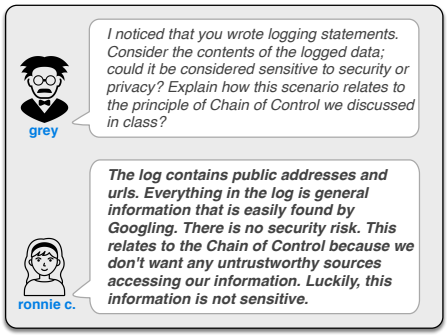

Figure 1: An example of automated reflection-in-action where a student is encouraged to learn about the security principle of Chain of Control, as they are working on developing a secure mobile app.

To address the challenge of scaling up prompt based reflection, in this paper, we propose and provide preliminary evidence for *automated prompt-based reflection*, an instructional strategy that applies principles from recommendation systems to suggest prompts to students based on their current activity. The recommendation system necessary to enable this instructional strategy uses as input students' ongoing work represented as changesets (or diffs) in source code management system (e.g. git), which are increasingly stored online via Web services like GitHub and BitBucket.

We evaluate the instructional strategy on reflection prompts targeting secure mobile application development. The prompts provide an opportunity to reinforce abstract security concepts introduced in lecture, by applying them to each individual student's progress on a practical project of implementing an Android app. Our results indicate that the instructional strategy is well received by students, leads to improvements in learning of security concepts, and even encourages students to revise and improve the design and implementation of the software project itself. Figure 1 shows an actual interaction with a prototype prompt recommendation system, where a contextual and timely reflective prompt is delivered

ITiCSE'19, 15-17 July 2019, Aberdeen, UK

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

to a student designing an Android app with a login screen. The contributions of this paper are:

- instructional strategy for automatically instigating reflection using prompts in the course of software development assignments or projects;
- design of a recommendation system based technique for automated prompt-based reflection;
- preliminary evaluation results of the efficacy of the instructional strategy.

The rest of this paper is organized as follows. Section 2 outlines the background and related work. In Section 3 we describe the automated system for reflection and the set of prompts we defined for teaching secure mobile development. Section 4 discusses preliminary results in applying the system and instructional strategy in a mobile development focused course. Finally, we conclude the paper and discuss the future work in Section 5.

2 BACKGROUND AND RELATED WORK

The practice of reflection helps learners gain tacit knowledge. Different from a novice, an expert has a larger body of tacit or experiential knowledge from which the expert can instinctively draw insights and apply them to current problem settings [15]. Software development is a discipline where the tacit knowledge is not easily captured by a set of rules and formulae. Therefore, it is difficult to be communicated by an instructor using lecture, and often requires project-based or problem-based instruction. Reflection is a key component in connecting concepts discussed in lecture, with those observed during non-trivial projects or assignments.

Encouraging learners and practitioners towards quality reflection on their experiences has been attempted and investigated by many educators and researchers. For instance, a frequently used reflection technique is for students to write reflective essays or blogs [26]. Our approach is different as it is based on prompts, which are far more specific about what is being reflected than an open-ended essay writing or blogging. Therefore, text of the prompts plays an important role in the quality of the student reflection.

Two other important characteristics that make reflective prompts effective are adaptivity and timeliness [7, 12]. Adaptivity means that reflection should be specific to the context of the current work by the student, and therefore able to influence the students future action on the project. Timeliness refers to the sensitivity of the prompts to the current actions by the students, in a way that the prompts do not completely disrupt the student at key points of their work. While adaptivity and timeliness can be obtained automatically using the system we describe in the paper, the quality of the prompts' content still requires careful tuning by the instructor.

There are numerous existing uses of reflection either in computer science education or aided by automated tools [6, 10, 13, 19, 22, 24–26]. Researches have also developed automated reflection tools that help learners, more specifically, designers, reflect [19, 24]. These tools are not prompt-based, focusing instead on guiding designers to carry out open-ended reflection activities, such as, writing reflective logs, curating visual bookmarks, and visualizing creative designs [24, 26].

Edwards investigated students' reflection via a test-driven development approach in lower-division programming courses, suggesting reflection's effectiveness in the fostering of comprehension, analysis, and hypothesis-testing skills across programming assignments [6]. Different from the type of reflection we describe, Edwards used reflection that is explicit and algorithmic as expressed in the program code to construct automated test cases.

Fostering effective reflection across a wide range of students is in effect a very challenging and complex topic. Although reflection's benefits are often regarded as self-evident and central to building capacity for lifelong learning, many concerns exist in applying reflective practice [8]. For instance, from the perspective of cognitive load theory, reflective practice, if not applied well, can overwhelm students and have negative impacts on students' learning [11, 23], in particular on novices. Proper scaffolding and guidance are important to reduce students' cognitive load when instigating reflection [9, 17]. This paper describes a specific approach that allows individualized delivery of prompts after students' incremental submission of their work. We also investigate whether our prompts and the automated timing of their delivery can target opportune times for reflection, which does not produce undue cognitive burden in reflection.

3 RECOMMENDING PROMPTS FOR REFLECTION

3.1 Rule-Based Recommendation System

From the educator's perspective, a typical use case of the our automated prompt-based reflection consists of a few steps. First, the instructor composes a set of reflective prompts specific to an assignment or project. The reflective prompts may reference content from lecture, or reference external documents, such as book chapters, API documentation, and research articles. The set of reflective prompts and the supporting explanation carry the reflection guidance provided to students.

Second, the instructor utilizes a prompt recommendation and delivery system, an essential component of the automated reflection approach described in this paper. This prompt recommendation and delivery scheme can range from the instructor specifying rules for the delivery of each prompt (e.g. a student invoking a method on the `Log` static class could trigger the prompt in Figure 1) to more general and automated approaches that allow for flexible expression of how prompts should be delivered.

Third, the student responses to the prompts are collected and delivered to the instructor. While directly grading the prompts is discouraged [8], their content provides valuable feedback to the instructor on students understanding of specific concepts.

In building a prompt recommendation system, there are design choices along two dimensions: 1) input data; and 2) models of student software development activity. The prompt recommendation's system input data can be static, i.e., submitted source code upon the completion of the project, episodic, i.e., versions of students' source code committed to a source code repository, or live, i.e., students' activity streaming from their IDE. From episodic data we can learn a student's progress on a project, while from interaction data we can infer what actions the student is carrying out at each moment in time. Along the second dimension, student software development

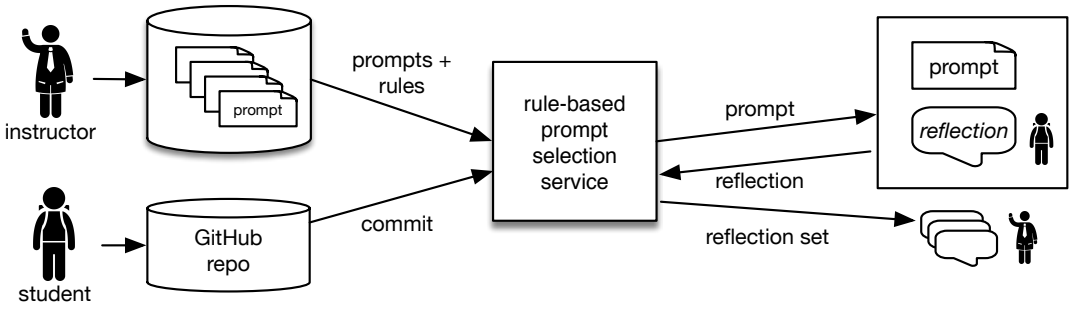

Figure 2: Design of the rule-based prompt delivery system.

activity models can be deterministic, typically consisting of a set of rules, or be probabilistic, building a probability distribution over the reflective prompts. In this paper, we concentrate on describing a a *rule-based* approach based on episodic changeset data. The advantage of a rule-based system is its simplicity, while the disadvantage that it requires the instructor express a specific rule (i.e. a regular expression over some aspect of the changeset) for the delivery of each prompt. Next, we describe the architecture of our system.

Figure 2 illustrates the major components and actors in the rule-based prompt recommendation system. We base our architecture of the rule-based prompt recommendation system on GitHub, which offers integration abilities via RESTful Web services. The instructor initially constructs reflection prompts [1], and rules for their delivery that are represented as regular expressions. The set of prompts are stored in a reflection prompt database, which we implement as a separate Web service. As students push their local commits to the central repository, GitHub's Webhook mechanism communicates their content to the rule-based prompt selection (Web) service. The delivery service examines the content of each commit (or diff), executes each rule (i.e. regular expression) from the database, selecting a prompt from the set of available reflection prompts. Following the selection, the service extracts the student's e-mail address from GitHub's payload, and uses it to immediately send an e-mail to the student containing the reflection prompt and brief instructions. The e-mail uses the instructor's account, for which we use GMail's RESTful Web service, although many other general alternatives are possible. Once the prompt is delivered to students, the students compose answers to the prompts as e-mail replies that are sent to the instructor of the course.

3.2 Reflection Prompts

We designed a set of reflection prompts to help students learn secure design principles as they are applied in mobile application development. We considered two dimensions when designing reflection prompts, content relevancy and work relevancy.

To establish content relevancy, we identified a list of secure design principles, secure coding standards, and secure coding best practice. Saltzer and Schroeder devised a list of principles for information protection, such as, fail-safe default, least privilege, separation of privilege, and complete mediation [16]. These principles are often used as guidelines to develop secure systems, and discussed in well-known textbooks, such as, "Computer Security: Art and Science" by Bishop [3] and "Security in Computing" by Pfleeger and Pfleeger [14]. More recently, Smith revisited Saltzer and Schroeder's principles and gave them a contemporary treatment, yielding a list of overlapping secure principles [20]. In addition, for Android mobile app development, more specific secure development recommendations exist, such as, the Android Secure Coding Standard by the Software Engineering Institute [21] (the secure coding standard) and Android's "App Security Best Practices" [1] (the best practice).

We establish a mapping among the secure design principles, the secure coding standard, and the best practice, and write prompts aiming at providing students with opportunity to reflect on a selected list of secure design principles. Table 1 lists all the concepts, prompts and rules that we created. Although in this study we focus on secure software design principles, note that the reflective prompts and the system we described can be applied to guide students through various CS topics (e.g. operating systems, programming languages) with easy to discern patterns in the source code.

4 PRELIMINARY STUDY RESULTS

To investigate the impact of the automated prompt-based reflection on students' learning, we conducted a study with about 80 computer science students, focusing on concepts in software security. During the study we collected a set of preliminary results based on students' reflections and surveys, that we present in this section.

4.1 Method

We deployed the system and the prompts in a Fall 2018 Software Engineering course taught at one of the authors' affiliated institutions to junior and senior undergraduate students. The course is based on a large scale group project, an Android App of student's own choosing, spanning 3 graded iterations each lasting roughly 2 weeks. A lecture on secure software development was delivered in class, discussing overall principles and a set of examples extracted from real-world Android security exploits [1, 16, 20].

Students were informed about the automated prompt-based reflection project upon beginning of the study. Participation in the study was voluntary, although was encouraged with a bonus point.

We run the rule-based automatic prompt-based reflection system for the period about 5 weeks, covering 2 out of 3 project iterations, starting from October 4, 2018 to November 12, 2018. Based on

[1] An alternative to this is for the instructor to reuse existing prompts on a specific topic.

Table 1: Security-related reflection prompts targeting mobile development.

Secure Principles by Saltzer and Schroeder [16] and Smith [20]	Regex Rule	Prompt
Least Privilege	`"<uses-permission"` in AndroidManifest.xml	I noticed that your app requires the use of a new permission (defined in the Android Manifest file). Permissions may pose a risk to users' privacy so I would like you to answer these questions: 1) Is the permission necessary for the functioning of your app? 2) Does your app need to hold the permission all the time when the app is running? If not, when should you request the permission and when should you relinquish the permission? 3) Explain how does the Principle of Least Privilege that we discussed in class relate to this scenario?
Fail-safe default / Deny by default	`"<provider"` in AndroidManifest.xml	I noticed that you declared a new Content Provider in your Android Manifest file. Recall that if you do not wish to send data from your app to another app, which you probably do not, you should understand the "android:exported" attribute of the provider element and make sure the value of it is set properly. Explain how does the use of a content provider and setting this attribute relate to the the Principle of Deny by Default we discussed in class?
Least common mechanism / Chain of control	`"new\s+Intent.+ACTION"` in *.java	I noticed that you are using an implicit Intent. Think about the fact that the data we package in an implicit Intent can be received by anyone that registers for that Intent broadcast. Do you think this exploit is possible for your application? Explain how this scenario relates to the principle of Chain of Control we discussed in class?
Least common mechanism / Chain of control	`"openFileOutput\("` in *.java	I noticed that your app stores data in a file. Does the stored data contain any sensitive information? Who has access to the data (i.e. is it internal or external storage)? Explain how this scenario relates to the principle of Chain of Control we discussed in class?
Least common mechanism / Chain of control	`"Log\."` in *.java	I noticed that you wrote logging statements. This is great as logging is an important means to diagnose an application in development and in production. Consider the contents of the logged data; could it be considered sensitive to security or privacy? Explain how this scenario relates to the principle of Chain of Control we discussed in class?

the student's commit pushed to the remote repository on Github, the system automatically determines whether a security related reflection prompt should be sent. The student may receive multiple reflection prompts, however a reflection prompt is not sent twice to the same student. Upon the conclusion of each iteration, we sent an anonymous and voluntary post-reflection survey.

4.2 Data Collection

Over the course of the study, we collected data about student's reflections, including answers to the reflection prompts, the time of sending a prompt and the time of receiving an answer. Overall, we sent 35 reflective prompts and gathered 26 responses from 20 students.

At the end of an iteration, we sent the post-reflection survey via email to students that received at least one prompt during the iteration. The survey was designed to evaluate the efficacy of the proposed approach from a viewpoint of a student. The questions in the survey relates to e.g. prompts difficulty and relevancy, effort required to answer a prompt and the influence of reflection on the students' knowledge.

4.3 Analysis and Evaluation

Data collected during the automated prompt-based reflection study is depicted by Figure 3 composed of part (a) related to to the students' reflections and part (b) presenting the survey summary. Overall, the results are consistent with the literature and confirm that students benefit from the reflection practice. Most of the students that received the reflection prompt, decided to participate in the study (20 out of 29) and more than half of the provided answers were completely to the instructor's satisfaction. The voluntary post-reflection survey was completed by 15 out of 20 students, with 13 students that responded to the received prompt. The post-reflection survey indicates that students generally had positive feeling toward the reflection process. It is noteworthy that 1/3 of students (5) who answered the survey realized their violation of secure design principles only after they received the prompts, and almost equal number of students (6) indicated that they either modified or planned to modify their code in a result. The majority of these students considered the reflection process had great impact on their learning of secure design principles and practice.

In the following, we discuss a few observations from the data.

Students' reflections sizes. We measure the size of a student's reflection using word count. The distribution of words per an answer is presented in the upper part of Figure 3. The number of words in a reflection is commonly over 100 words (16 out of 26 reflections) implying that most of the students engaged in the reflection process.

Timing of the prompts. One of the objective of the study was to provide students with the reflection prompt at convenient time. Analyzing the time interval between sending the prompt and receiving the student's answer, we observe that only 10 out of 26 responses were provided within the first day of receiving the reflection prompt, although this results may have been influenced by the medium used to disseminate prompts as the frequency of checking an e-mail application may vary between students.

To evaluate the timing of prompts from students' perspective, during the survey the students were asked to assess if they were

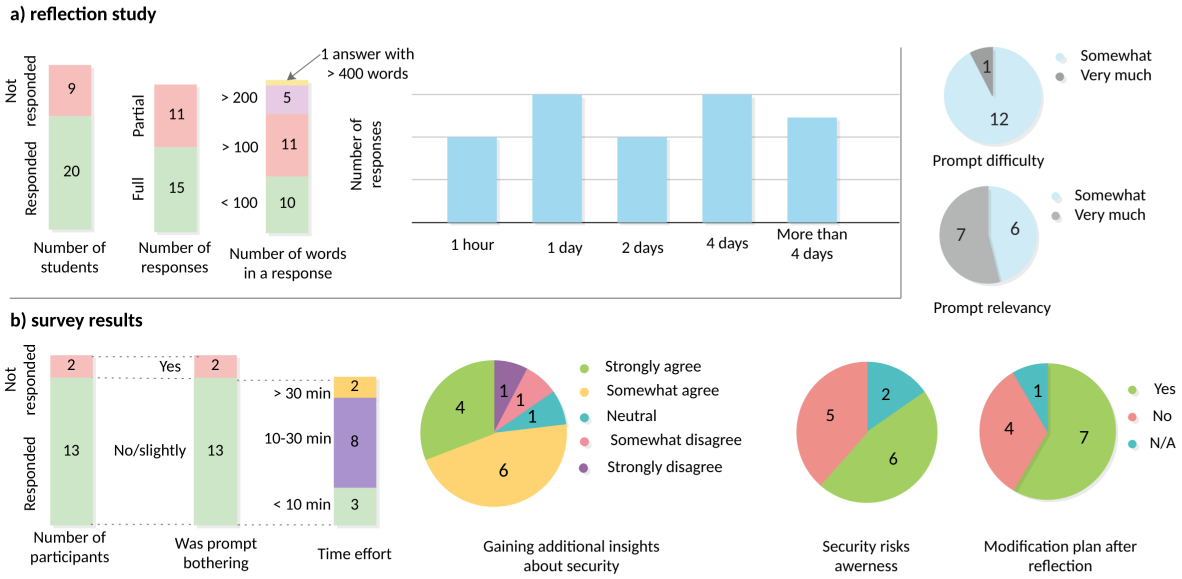

Figure 3: Overview of the data collected via the preliminary study consisting of reflection responses(a) and a post-reflection participant survey (b).

bothered by receiving the prompt while working on the project. All of 13 students that responded to the prompts answered that they were either not bothered at all or slightly bothered. However, the 2 survey participants who did not answer the prompts, indicated that the reason for not responding was that they were too busy to reflect. It seems that sending the reflection prompts when students update the remote code repository is an appropriate time only to some students *but not to all*. This also could be an indication that the students who are not bothered by the prompts may have reached a milestone in their project work.

Quality of the reflection prompts. We consider two factors when determining prompt quality: relevancy of the prompts to the course material and difficulty to answer the prompts. All the 15 survey respondents answered that prompts are *somewhat relevant* or *very relevant* to the course material. Only 1 student believed that the prompt was too difficult to answer, while 12 students indicated that the prompts were *somewhat difficult*. Based on these results, we conclude that the prompts were relevant and at a proper level of difficulty to reflect effectively.

Students' effort of reflections. Only 2 students claimed that they spent more than 30 minutes to answer the reflection prompts, while most of the students, 8 out of 13, estimated they needed between 10 and 30 minutes to provide a response. This result suggests that the prompts did not put significant additional burden on the students, while allowing them to refresh or expand their knowledge.

Impact on students' learning. Finally, we examine the impact of the automatically prompted reflection on students' learning effectiveness and outcomes. 5 out of 13 students reported they were not aware of potential security risks in their code, however, all of them indicated that they had revised or had a plan to revise the code as the result of answering the prompts. In addition, one of the students, that was aware of the risk, also decided to revise or improve the code. Most of the students, 10 out of 13, indicated that the reflection helped them gain additional insights or strengthened their knowledge of security principles.

4.4 Students' Comments and Reflection

Analyzing the student's free-text comments collected via the post-reflection survey, we noticed that students found the reflective practice to be beneficial for their learning process. For instance, one student wrote,

> *"I think it was extremely helpful as it forced me to research and refresh my mind to answer. It was also helpful in relation to our final [...]. The prompt was also good because it wasn't just a straightforward question, but it had information and terms I did not know so I was able to research and gain more knowledge before answering."*

and yet another indicated,

> *"Not logging certain information was something I never thought about being a possible security crack in a program. But it does make perfect sense that you need to be careful to not log information that could be exploited."*

These comments reveal that students, encouraged by the reflection, are able to expand their knowledge and gain new insights by conducting additional research. In addition, some of students' reflections show that the students pondered alternatives and gained practical experience by relating the reflection process to real-world applications, such as,

> *"[t]he logging statements I used were mainly for debugging purposes. However, they did print out the* `toString` *of events that the user created for the calendar. This could be considered sensitive information, as the user might not want any outside sources seeing these events. Much like the Principle of Chain Custody says, the programmer should not log anything that could be considered sensitive or private, as the log can be obtained rather easily, like by connecting an Android device to a PC. In a future iteration, I will be sure to remove logging statements that output the* `toString` *of the user's*

events before I update the repository, in order to protect the user's privacy, or ensure that I am not logging any private information."

4.5 Threats to Validity

This exploratory study aims to show that an automatic prompt-based reflection system can be applied to computer science courses to help students improve learning outcomes. While the sample size of this case study is not large enough to provide any robust and general conclusions, the collected data provides some evidence that the described intervention via the system is consistent with the literature on applying reflection.

The work suffers several threats to validity. For instance, there are a few threats to external validity. First, we informed students that we wanted to observe how they would respond to reflection prompts, which would influence students' decision whether to take part in the study. This is in effect an instance of "Hawthorne effect (attention causes differences)". Second, it might as well be that these group of students were the very first to be prompted for reflection. This is an example of "novelty and disruption effect (anything different makes a difference)". Because this intervention is new to them, the students responded to the uniqueness of the approach rather than the actual reflective practice. Third, we designed and implemented the system, and unknowingly implemented reflective intervention in a way that worked with this group of students. However, whether other educators can achieve the same results while using our system remains untested. This is in effect called "experimenter effect". We discuss in part some approaches to alleviate these threats in Section 5.

5 CONCLUSIONS AND FUTURE WORK

This paper describes a strategy for automated prompt-based reflection. Using a rule-based prompt delivery system, we instantiate this strategy for teaching secure development in mobile programming courses. We present preliminary results based on our experience in applying the system and strategy to a junior-level computer science course. The results show that the system enables us to scale-up prompt based reflection. The students were relatively positive on their experience responding to the prompts, while the quantitative data also show high rate of prompt responses.

Apart from growing the scale of this study to increase sample size and variety, we consider a few enhancement of the automatic prompt-based reflection system. First, the prompts at present are hand-written and selected by the system based a set of rigid rules. It is easy to make a mistake in the rules which results in the wrong prompt being delivered to the student. Our aim to enhance the way instructors can express the prompt delivery rules, or provide automated targeting of prompts without rules, e.g., based on past student commit histories.

Apart from this, we are also interested in improving and expanding the set of prompts targeting mobile application development. As one student commented in our post-reflection survey: *"could ask more challenging questions that we haven't covered, like those related to databases or storing passwords, etc."*

REFERENCES

[1] Android. 2018. App security best practices. (2018). available via https://developer.android.com/topic/security/best-practices, retrieved November 12, 2018.

[2] Eric P.S. Baumer, Vera Khovanskaya, Mark Matthews, Lindsay Reynolds, Victoria Schwanda Sosik, and Geri Gay. 2014. Reviewing Reflection: On the Use of Reflection in Interactive System Design. In *Proceedings of the 2014 Conference on Designing Interactive Systems (DIS '14)*. ACM, New York, NY, USA, 93–102. https://doi.org/10.1145/2598510.2598598

[3] Matt Bishop. 2003. *Computer security: art and science*. Addison-Wesley Professional.

[4] Christopher N. Bull and Jon Whittle. 2014. Supporting Reflective Practice in Software Engineering Education through a Studio-Based Approach. *IEEE Software* 31, 4 (July 2014), 44–50. https://doi.org/10.1109/MS.2014.52

[5] John W. Coffey. 2017. A Study of the Use of a Reflective Activity to Improve Students' Software Design Capabilities. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 129–134. https://doi.org/10.1145/3017680.3017770

[6] Stephen H. Edwards. 2004. Using Software Testing to Move Students from Trial-and-error to Reflection-in-action. *SIGCSE Bull.* 36, 1 (March 2004), 26–30. https://doi.org/10.1145/1028174.971312

[7] Angela Fessl, Oliver Blunk, Michael Prilla, and Viktoria Pammer. 2017. The known universe of reflection guidance: a literature review. *International Journal of Technology Enhanced Learning* 9, 2-3 (2017), 103–125.

[8] Linda Finlay. 2008. Reflecting on reflective practice. *PBPL paper* 52 (2008), 1–27. Available: https://www.open.ac.uk/opencetl/resources/pbpl-resources/finlay-l-2008-reflecting-reflective-practice-pbpl-paper-52.

[9] Cindy E Hmelo-Silver, Ravit Golan Duncan, and Clark A Chinn. 2007. Scaffolding and achievement in problem-based and inquiry learning: A response to Kirschner, Sweller, and Clark (2006). *Educational psychologist* 42, 2 (2007), 99–107.

[10] Norio Ishii and Kazuhisa Miwa. 2005. Supporting Reflective Practice in Creativity Education. In *Proceedings of the 5th Conference on Creativity & Cognition (C&C '05)*. ACM, New York, NY, USA, 150–157. https://doi.org/10.1145/1056224.1056246

[11] Paul A Kirschner, John Sweller, and Richard E Clark. 2006. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist* 41, 2 (2006), 75–86.

[12] Äli Leijen, Kai Valtna, Djuddah AJ Leijen, and Margus Pedaste. 2012. How to determine the quality of students' reflections? *Studies in Higher Education* 37, 2 (2012), 203–217.

[13] Stephen MacNeil. 2017. Tools to Support Data-driven Reflective Learning. In *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17)*. ACM, New York, NY, USA, 299–300. https://doi.org/10.1145/3105726.3105745

[14] Charles P Pfleeger and Shari Lawrence Pfleeger. 2002. *Security in computing*. Prentice Hall Professional Technical Reference.

[15] Gary Rolfe. 1997. Beyond expertise: theory, practice and the reflexive practitioner. *Journal of clinical nursing* 6, 2 (1997), 93–97.

[16] Jerome H Saltzer and Michael D Schroeder. 1975. The protection of information in computer systems. *Proc. IEEE* 63, 9 (1975), 1278–1308.

[17] Henk G Schmidt, Sofie MM Loyens, Tamara Van Gog, and Fred Paas. 2007. Problem-based learning is compatible with human cognitive architecture: Commentary on Kirschner, Sweller, and Clark (2006). *Educational psychologist* 42, 2 (2007), 91–97.

[18] Donald A. Schön. 1987. *Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions*. Wiley. https://books.google.com/books?id=qqxsQgAACAAJ

[19] Moushumi Sharmin and Brian P. Bailey. 2013. ReflectionSpace: An Interactive Visualization Tool for Supporting Reflection-on-action in Design. In *Proceedings of the 9th ACM Conference on Creativity & Cognition (C&C '13)*. ACM, New York, NY, USA, 83–92. https://doi.org/10.1145/2466627.2466645

[20] Richard E Smith. 2012. A contemporary look at Saltzer and Schroeder's 1975 design principles. *IEEE Security & Privacy* 10, 6 (2012), 20–25.

[21] Software Engineering Institute. 2018. Android Secure Coding Standard. (2018). available via https://wiki.sei.cmu.edu/confluence/display/android, retrieved November 12, 2018.

[22] Jeffrey A. Stone. 2012. Using Reflective Blogs for Pedagogical Feedback in CS1. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. ACM, New York, NY, USA, 259–264. https://doi.org/10.1145/2157136.2157216

[23] John Sweller, Paul A Kirschner, and Richard E Clark. 2007. Why minimally guided teaching techniques do not work: A reply to commentaries. *Educational psychologist* 42, 2 (2007), 115–121.

[24] Andrew M. Webb, Rhema Linder, Andruid Kerne, Nic Lupfer, Yin Qu, Bryant Poffenberger, and Colton Revia. 2013. Promoting Reflection and Interpretation in Education: Curating Rich Bookmarks As Information Composition. In *Proceedings of the 9th ACM Conference on Creativity & Cognition (C&C '13)*. ACM, New York, NY, USA, 53–62. https://doi.org/10.1145/2466627.2466636

[25] Yu-Chun Grace Yen, Steven P. Dow, Elizabeth Gerber, and Brian P. Bailey. 2017. Listen to Others, Listen to Yourself: Combining Feedback Review and Reflection to Improve Iterative Design. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition (C&C '17)*. ACM, New York, NY, USA, 158–170.

[26] Jose P. Zagal and Amy S. Bruckman. 2007. GameLog: Fostering Reflective Gameplaying for Learning. In *Proceedings of the 2007 ACM SIGGRAPH Symposium on Video Games (Sandbox '07)*. ACM, New York, NY, USA, 31–38.