In: Katsikas S. et al. (eds) Computer Security. SECPRE 2018, CyberICPS 2018. Lecture Notes in Computer Science, vol 11387. Springer, Cham
https://doi.org/10.1007/978-3-030-12786-2_10

Understanding challenges to adoption of the Protection Poker software security game

Inger Anne Tøndel[1,2], Martin Gilje Jaatun[2], Daniela Cruzes[2], and Tosin Daniel Oyetoyan[2]

[1] Department of Computer Science, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway
[2] Department of Software Engineering, Safety and Security, SINTEF Digital, NO-7465 Trondheim, Norway
inger.anne.tondel@ntnu.no

Abstract. Currently, security requirements are often neglected in agile projects. Despite many approaches to agile security requirements engineering in literature, there is little empirical research available on why there is limited adoption of these techniques. In this paper we describe a case study on challenges facing adoption of the Protection Poker game; a collaborative and lightweight software security risk estimation technique that is particularly suited for agile teams. Results show that Protection Poker has the potential to be adopted by agile teams. Key benefits identified include good discussions on security and the development project, increased knowledge and awareness of security, and contributions to security requirements. Challenges include managing discussions and the time it takes to play, ensuring confidence in the results from playing the game, and integrating results in a way that improves security of the end-product.

1 Introduction

Current software development is increasingly based on agile methods. Agile software development aims to reduce development time and prioritise value through an iterative approach to development [6]. Agile methods claim to be risk driven [3] and risk management can be said to be treated implicitly in agile development projects [18, 25], e.g. through prioritising tasks in the beginning of the iteration.

Security risk is one type of risk that software products face today. Literature reviews on software security in agile development have found that security requirements are often neglected in agile projects [19, 15]. Though several approaches to agile security requirements engineering have been suggested in literature, there has been little empirical work done on evaluating how these security requirements approaches work in real life settings [26]. Studies have however identified security benefits that can be traced back to using an incremental risk analysis approach [2]. Thus, it is important to understand better how agile teams can be supported in analysing software security risks and requirements in an ongoing manner.

This paper presents a study of Protection Poker [32, 33] in a capstone development project with six development groups. Protection Poker is based on Planning Poker [10], and is a security risk estimation technique for agile development teams. It is intended to be played as part of every iteration planning meeting, in order to rank the security risk of each feature to be implemented in that iteration, and possibly identify additional security mechanisms that have to be implemented to maintain an acceptable risk level. The full team together identifies assets related to the features and uses the Protection Poker game to rank the features according to their security risk; assessing the value of their assets and the ease of attack.

Protection Poker has been evaluated previously in industry with positive evaluation results [33], however that study did not focus on adoption, but rather on awareness and knowledge raising through using the technique. Furthermore, despite positive evaluation results, the team that was studied stopped using Protection Poker sometime after the study was completed.

Motivated by a need to understand why Protection Poker or similar techniques have not yet gained widespread adoption in industry, the goal of the study presented in this paper was to assess to what extent the Protection Poker game would be accepted as a technique in agile teams, and if possible to determine obstacles to adoption of the game. Our investigation was centered on the following research questions:

RQ1: To what extent is Protection Poker accepted by the players, both short-term and longer term?
RQ2: What lessons learned and improvements are identified by the players?

The rest of this paper is structured as follows: In Section 2 we provide an overview of related work on security requirements engineering in agile development, to position Protection Poker related to other approaches in this research area and further motivate why Protection Poker was selected for study. In Section 3 we describe the research method used for the study that we conducted, as well as details on the version of Protection Poker used in this particular study. We present the results in Section 4, and discuss in Section 5. We conclude the paper in Section 6.

2 Security requirements in agile software development

Existing approaches to security requirements in agile software development are by and large in line with findings in a review of lightweight approaches to security requirements engineering [27], that points to three important and commonly recommended activities: identifying security objectives, identifying assets, and analysing threats to the system. To illustrate, Peeters [20] introduced abuser stories by extending the concept *user stories* that is commonly used in agile practices. Boström [4] suggested an approach to introduce security requirements engineering into XP. The approach includes identifying critical assets, formulating abuser stories and assessing their risk, and defining security-related user

stories and coding standards. Vähä-Sipilä [29] explained how security requirements can be described as security-related user stories or as attacker stories (abuse cases). Then, in development, security is added to the sprint's Definition of Done, by introducing a security threat analysis into the sprints and by flagging potentially risky tasks. Savola et al. [23] explained that security requirements are translated into negative user stories and into non-functional requirements. Nicolaysen et al. [17] suggest that security threats are identified related to functional requirements that is to be implemented, and that after the risk has been calculated at least one misuse story is created. Pohl and Hof [21] suggest the Secure Scrum approach that includes four components; identification, implementation, verification and definition of done. The identification component consists of two steps; 1) ranking user stories according to their loss value, and 2) evaluating misuse cases and ranking them by their risk. Specific tags and marks are used to link the security issues identified to the user stories in the backlog, and the approach specifies how to use these tags and marks in the development. Renatus et al. [22] suggest to split the task of security requirements into two steps performed by different roles; the security curator and developers. This allows for security curators that do not have in-depth technical expertise on the product that is developed and for developers without in-depths security expertise. The security curator's task is then to pre-model features that is to be implemented, identifying affected parts of the system and performing initial threat modeling. The developer then, during the sprints, *"figures out the details and implements the controls"* [22].

Though some of the above mentioned approaches stem from industry settings [20, 29, 23] or have been tried out in real development projects, the current evidence on how these approaches work in practice is limited. The approach of Renatus et al., which was broader than what is presented above on requirements engineering, was evaluated in one SME [22], however the research method used is not described in detail. The approach suggested by Pohl and Hof was evaluated in small student projects lasting only one week [21]. The approach suggested by Boström has been used in one student thesis project [4]. Thus, there is a need for empirical research that can shed light on how agile security requirements approaches work in practice, and what can be done to improve them and increase their adoption by agile development teams.

As already stated in the introduction, security requirements are often neglected in agile projects [19, 15]. A study of practitioners' posts on LinkedIn [26] shed some light on why this is the case: *"People do care about security, but do not think about it"*, *"Security requirements are often poorly defined and owned"*, *"Security requirements get often delivered in the last minute"* and *"Agile techniques are vulnerable for forgetting things like security."* These problematic aspects point to the need for concrete ways to introduce security into an agile project so that security is not forgotten, but rather considered throughout. Any approaches however need to consider the general lifecycle challenges related to security identified in the review by Oueslati et al. [19]: *"Security related activities*

need to be applied for each development iteration" and *"Iteration time is limited and may not fit time consuming security activities"*.

Weir et al. [31] identified, based on interviews with 14 specialists, what they consider the three most cost-effective and scalable security interventions in software development. These were all *"cultural interventions"* that influence the work of the teams rather than the artefacts produced; 1) *"developing a 'threat model'*, 2) having *"a motivational workshop engaging the team in genuine security problems"*, and 3) having continuing *"'nudges' to the developers to remind them of the importance of security."* Of the security requirements approaches introduced above, all can be said to develop a threat model of some sort. The security requirements can work as 'nudges' that remind developers of security. Protection Poker is however the technique that most clearly engages the whole team in discussing security problems, and does so in a way that is concrete and (hopefully) fun. Additionally, it is specifically designed to be applied for each development iteration, addressing the challenge identified by Oueslati et al. [19]. Though Protection Poker is not a full blown security requirement approach, but rather includes parts of the activities commonly used for requirements engineering, resulting in a ranking of features rather than specific security requirements, we believe that a study of Protection Poker with its focus on assets and ease of attack has the potential to be of use for researchers working on other agile security requirement techniques as well; identifying challenges and lessons learned that can be used to improve approaches in this research domain.

3 Research method

This section describes the research method used for the case study. Additionally, it describes in detail the version of Protection Poker used for this study as well as the motivation for the adjustments of the original technique [33].

3.1 Case study method

The research method in this case study is the same as that of a parallel study[3] [28], thus the text below that describes the method is similar in these two studies.

Regarding the *case context*, the study was performed in the *Customer Driven Project* course (TDT4290) at the Norwegian University for Science and Technology (NTNU), autumn 2016. This course is mandatory for 4th year computer science students. In this course, the students are divided into development teams (5-8 students per team). Every team is given a development project from an

[3] In addition to Protection Poker, one other technique (Microsoft EoP) was studied in the course. Groups were assigned to use either Protection Poker or EoP by two researchers in cooperation based on name of the project and name of the customer. In deciding which group should use which technique, the researchers aimed for a balance in size and type of customer and in the type of systems developed so that both games had a mixture of different project types.

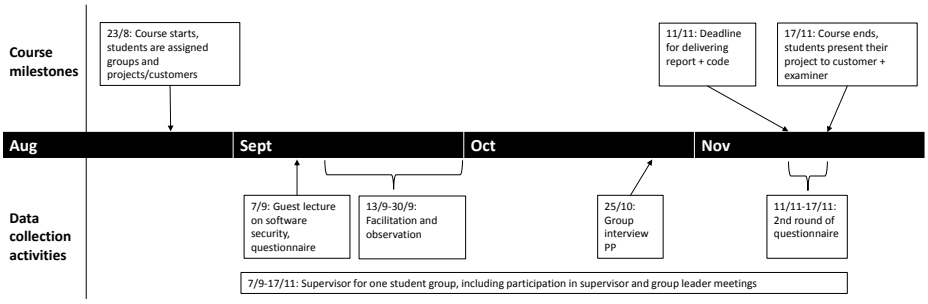

Fig. 1. Overview of data collection activities

external customer (i.e. private companies, public organisations or research institutes). The students are expected to investigate the needs of the customer, develop software, do some testing of this software and document everything in a report and a presentation given to the customer. In general, all student groups use agile methodologies to some extent. Six groups, consisting of 34 students in total, were required to use Protection Poker for their project. This was the first year software security was included as part of the course.

An overview of data collection activities can be found in Figure 1. As most students had received limited formal training on software security before this course[4], we arranged a lecture where all students were given a short plenary introduction to software security and the Protection Poker game. They played the game on an example project, and responded to a questionnaire that covered the students' acceptance of the technique. Data collection proceeded through facilitation and observations of students playing Protection Poker in their group, and the observations were followed by group interviews towards the end of the course, allowing detailed student feedback on the technique. Additionally, the main author of this paper acted as supervisor for one of the student groups and took part in project leader and supervisor meetings throughout the course. The questionnaire on acceptance was repeated towards the end of the course. The study has been reported to the national Data Protection Official for research. In the following we explain the data collection methods in more detail; the questionnaire, the observations and the group interviews.

The main motivation for using a *questionnaire* was to capture students' immediate and longer term acceptance of the Protection Poker technique (RQ1). A questionnaire could easily reach a large number of the students, and could easily be repeated. We decided to base the questionnaire on the Technology Acceptance Model (TAM) for two reasons. First, TAM, although being criticized [16], is considered a highly influential and commonly employed theory for describing an individual's acceptance of information systems. TAM, adapted from the Theory of Reasoned Action [1] and originally proposed by Davis [7], suggests that

[4] No mandatory training in security, except security being a minor part of some courses that mainly covered other topics.

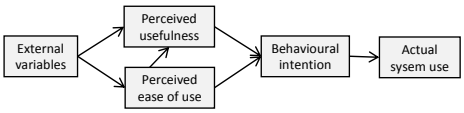

Fig. 2. TAM [30]

when users are presented with a new technology, a number of factors influence their decision about how and when they will use it (see Figure 2), notably:

- *Perceived usefulness:* this was defined by Davis as *"the degree to which a person believes that using a particular system would enhance his or her job performance"* [8]
- *Perceived ease of use:* Davis defined this as *"the degree to which a person believes that using a particular system would be free from effort"* [8].
- *External variables:* include *"system characteristics, training, user involvement in design, and the nature of the implementation process"* [30]

Thus, we believed TAM could help us understand the different reasons for acceptance of Protection Poker by the students, and that TAM-based questions could trigger comments from the students related to acceptance. Second, we were able to adapt questions from an existing questionnaire [9] to the phenomena we are studying (the questions used are shown in Figure 5).

For the *observations*, we created a rota where one of the authors served as facilitator, and at least one other author participated as observer. After each observation session, both the facilitator and the observer filled in reflection notes in a template that contained the following topics: group information; questions from the students on the technique; suggested changes to the game; participation; mood; topics discussed; what worked well with the game; challenges with the game, and; reflections on the observation and how the researchers may have influenced the process. After playing one session of Protection Poker, all groups were encouraged to keep on playing by themselves during the project, and we offered to return and offer support and/or facilitation at a later time, according to their needs.

Towards the end of the course, all groups were invited to send two to three participants to an event where the technique would be discussed in more detail. This event was organised as a *group interview* and was scheduled to last for two hours. The following topics were covered: students' expectations to the event; use of the game in the group; brainstorming and discussion on the 4Ls (Liked, Lacked, Learned, Longed for) [5]; suggestions for improvements to the technique; suggestions for improvements to how software security was handled in the course, and; feedback on the event. Discussions were recorded and transcribed. To encourage participation, all participants were served pizza and they had the opportunity to win cinema gift cards. Non-responding groups were reminded via email. To promote active participation in the group interviews, each event was split in two parallel sessions.

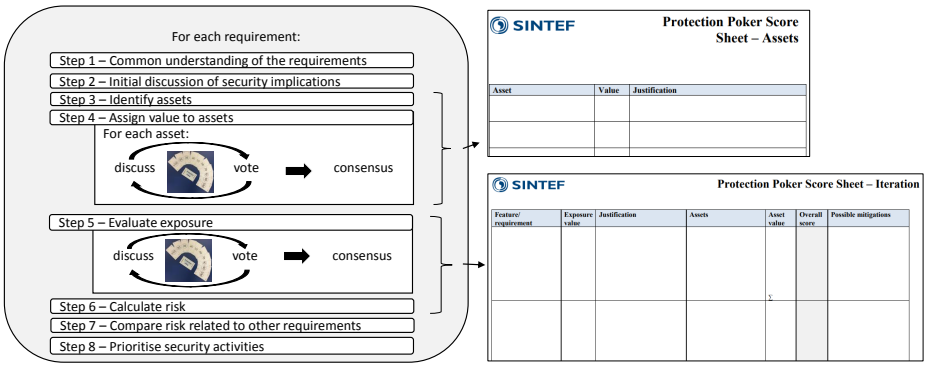

Fig. 3. Playing Protection Poker

3.2 Protection Poker as used in this study

The version of Protection Poker used in this study is a variation of the original Protection Poker game [33]. In this section we explain how Protection Poker is played, and the rationale for the modifications made to the game. In addition, we explain what a typical session looked like in this study. A similar but more detailed description of Protection Poker can be found in a previous publication [14]. Playing cards and score sheets to be used during playing are available online[5].

How to play Protection Poker is played during an iteration planning meeting, and it is recommended that the full development team participates. One person should have the role as moderator, and this person will be responsible for leading the team through the game, and point the discussions in a good direction. Ideally, a separate person should be tasked with taking notes on important security solutions and ideas that emerge during play. Focus is on the specific requirements the team will likely implement during the next iteration. A basic overview of the steps involved in playing Protection Poker can be found in Figure 3. The actual playing using the Protection Poker cards is done in steps 4 and 5. Players use the cards to make votes on the risk involved in the requirement they are playing on, and the votes are a basis for further discussions on the risk, and eventually agreeing on a risk value for the requirement. This may require several rounds of voting by using the Protection Poker cards. Below we explain two central concepts of the game, namely risk calculation and calibration.

Protection Poker uses a slight variation of the traditional computation of risk:

$$risk = (\sum \text{asset values}) \times (\text{the exposure}) \quad (1)$$

Risk is always related to a requirements that is to be implemented in the next iteration, often this will be some new, enhanced or corrected functionality. *Ex-*

[5] https://www.sintef.no/protection-poker

posure relates to how hard or easy the added or changed functionality makes it to attack the system. For *asset value*, one identifies the assets that are related to a requirement and considers their value for various actor types. Assets are typically considered to be *"data stored in database tables or system processes that the new functionality controls"* [33], however in this study we did not use a strict definition of the term asset. In previous work [13], we have defined assets as *"anything of value that needs to be protected"*.

To be able to prioritise between different requirements, it is important to be able to get a spread in the numbers assigned. Thus the highest card (100) should be used for asset values and exposures that are high for this project, and similarly the lowest card (<10) should be given to asset values and exposures that are low for this project. This is to avoid that, e.g., high risk projects rate every requirement with a high number. That would make it very hard to prioritize within the project. As the goal is not to establish a "perfect" and "universal" risk value, but rather to rate the security risk of the requirements in order to be able to better prioritize security effort, it is recommended to perform *calibration* in the beginning in order to arrive at a common understanding of the end-points of the scale, i.e., the team agrees what a <10 or a 100 means for this product. When playing about asset value and exposure, numbers should be assigned relative to these endpoints, as well as the values assigned for previously assessed assets and features.

Modifications. The changes made [14] to the original Protection Poker technique came from initial experiences with playing the game in two EU funded research projects in spring 2015. In the first trial, two of the authors played the game together with one colleague on an incident management tool for cloud service providers that was under development, while another author took part as observer. In the second trial, the technique was tried out in a research project that developed a health app. Four researchers from the project played the game on their project, and two of the authors took part as facilitator and observer. The adjustments made concern two main aspects: terminology and the scale used. On terminology, whereas the original Protection Poker version uses the term "ease of exploitation", we found that this concept was distracting or not properly understood by some pilot players, e.g., leading them to focus too much on threats such as "shoulder surfing" that are easy to perform. In order to focus more on how a feature increases the attack surface of an application, we decided to change it to "exposure". Regarding the scale, the original Protection Poker version uses the same cards as Planning Poker [10], used for effort estimation in agile teams. Planning Poker cards follow a Fibonacci-like sequence, after the rationale that it is easier to have an opinion on whether a task takes 1 or 2 days than whether it takes 40 or 41 days. We argue, however that the same is not true when it comes to relative value of assets or degree of exposure, and since we are less concerned about small risks and more interested in the bigger risks, we opted for a uniform scale instead. This enables us to differentiate between big risks, not just the small ones. Thus, we used Protection Poker cards with the

following numbers to determine asset value or system exposure: <10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

What was a typical session like? When we facilitated the students in playing the Protection Poker game in this study, we covered steps 1-6 in Figure 3 in addition to calibration. The Protection Poker sessions lasted between 50 and 70 minutes. The session started by having the students explain their system to the facilitator. Then the facilitator led the students to start identifying assets and calibrate assets. We prioritised calibrating the top end of the scale. The groups played on two to three assets, and spent between one and 17 minutes per asset played. For most (10 of 14) of the assets, the students were able to agree on a value with two rounds of playing the cards.

Features were identified and calibrated in the same way as assets, however calibration of features was skipped in three of the groups due to limited time left. We prioritised playing about features above identifying and calibrating features. One group did not play on any features, because the nature of their project (creation of an algorithm) made it difficult to come up with features. The other five groups played on one to three features. The students spent between two and nine minutes per feature played. For all but one feature, it was necessary to play two rounds.

Throughout, the facilitator was active in helping the group reach a consensus by suggesting compromise values. This was done to speed up the playing, terminating discussions when it seemed most arguments had been raised. The session ended with reflection about the experience, and the students were asked to provide feedback and suggest improvements.

4 Results

This section presents the results according to the two research questions of the case study. In addition, Table 1 gives an overview of the *observer notes* related to what worked well and what was challenging, and Figure 4 gives an overview of the students' feedback in the group interviews. In the observations we found that only two of the four groups had obvious security concerns. The group interviews, however, had low participation from those groups; only one participant from only one of those groups, while all the groups with limited security concerns participated with 2-3 people.

4.1 Acceptance of Protection Poker

Acceptance of Protection Poker (RQ1) was mainly studied through the TAM-based questionnaire in the beginning and the end of the course. In this section we provide an overview of the questionnaire responses and explain how observations and group interview responses add to the findings from the questionnaire. Figure 5 gives an overview of the questionnaire results on the TAM-variables *future use intention*, *perceived usefulness* and *perceived ease of use*. The results marked

Issue	Worked well	Challenge
Calibration	• Easy to find the top asset/feature (5) • Calibration resulted in involvement and good discussions (2) • Managed to use the full scale although calibration was not done/only done for highest value (2)	• Did not calibrate the bottom end of the scale (4) • Calibration was skipped for features (3) • Lack of calibration made the scale unclear and this impacted discussion negatively (3) • Few assets or features to calibrate (2) • Took time (1) • Facilitator influence a lot - propose and they agree (1)
Identify features	• Easy to identify features (4)	• Different understanding of features (2) • Few features in the system (2) • Difficult to identify features (1)
Identify assets	• Easy to identify assets (4)	• Some types of assets are difficult (e.g. access rights, reputation, libraries)(4) • Few assets in the system (3) • Confidentiality vs. integrity vs. availability of assets (2)
Discuss and vote on asset value	• Led to good discussions (4)	• Asset value get mixed with exposure (2) • Difficult to relate to and use the whole scale (2)
Discuss and vote on exposure	• Worked well in the group (3) • Clarification, led to common understanding of features (1)	• Exposure and asset value gets mixed up (5) • The term exposure is difficult to explain and understand (2) • Few features (1) • Limited time (1)
Calculate risk level	• Worked well in the group (4) • Triggered security discussions (2) • The score sheet helped progress by making students aware of next asset/feature to discuss (1)	• Questions on how to fill it out (3) • Assets at different levels may lead to skewed risk values (1) • Many scores to assign before calculating the first feature (1) • One feature without an asset (1) • Limited time (1)
Facilitation	• Facilitation ensured progress in the playing (2) • Easy to achieve consensus (2)	• Some participants posed a challenge for the facilitator (3)
Keep track of important parts of the discussion	• The score sheet included a column for "justification" and this triggered students to put something there (3) • Students took additional notes in note books (2)	• Scores are included on score sheet, but important aspects from the discussion are lost (3)

Table 1. Observation notes - count shows for how many groups an aspect was noted by the observer/facilitator.

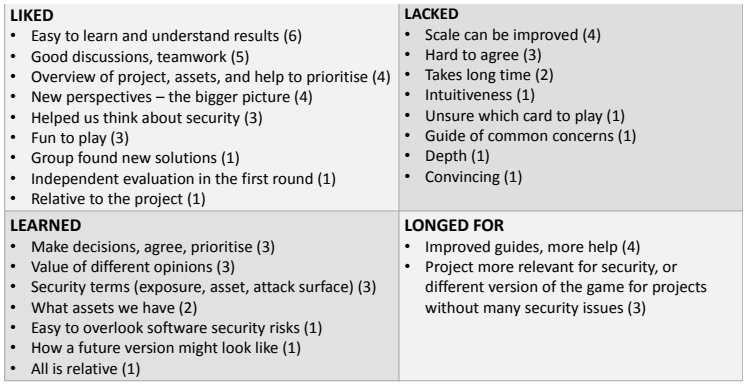

Fig. 4. Result of 4L brainstorming on Protection Poker

before refer to responses at the end of the introductory lecture (29 responses), and the results marked *after* refer to responses at the end of the course (30 responses).

Four questions together cover the variable *future use intention*. Though slightly declining throughout the course, the students tend towards being positive to use Protection Poker. One obvious reason for the decline in future use intention, especially concerning questions 1 and 2 where the biggest decline is observed, is the requirement to use Protection Poker in the course, something that will not be the case for any future projects the students encounter. In the end, half (15) of the students agree that they would like to use Protection Poker in the future, while only 5 respond not wanting to use Protection Poker (question 4).

Four questions together cover the variable *perceived usefulness*. In general the students seem to have found Protection Poker to be useful; in the end more students agree (14) than disagree (4) that the advantages of using Protection Poker outweigh the disadvantages (question 8). Expectations on what Protection Poker would deliver was in general high, however, it seems that it did not quite deliver in their current project (question 5). In particular, Protection Poker does not seem to have delivered on security; not improving the security of the product (questions 6) and not reducing security defects (question 7). The open-ended responses on the questionnaire shed some light on these responses. Though students did expect Protection Poker to have benefits, they were divided in their expectations. Ten out of the 28 that responded to the open ended question *"How do you think playing PP will influence the product?"* stated that they did not expect much influence. Of those that did expect an influence, the majority (11) expected it to improve security awareness. Other expectations included identifying the most important parts regarding security (4), a more secure product (3), discussions on security (2) and agreement on security issues in the group (2). Those that explained why they did not expect an impact from playing the game, explained that this was due to limited security issues in their project.

Open ended responses to the question *"How do you believe software security is important to your project?"* confirmed our observations that only two of the six groups had clear security issues that needed to be dealt with, while four groups had very limited attack surfaces or assets of little value, thus having limited security needs. Since it is likely that the limited security needs of projects influenced the usefulness of the technique when it comes to security, we additionally looked at the responses from the two groups that had security issues (10 responses) in isolation. We found that for question 6, all five students that agree that using Protection Poker improved security come from these two groups. When it comes to reduction of security defects (question 7), the students from the groups with security issues are more positive than the others, however, also these students in general do not agree that they experienced such a reduction from playing Protection Poker.

Despite limited need for security, the questionnaire responses indicate that students ended up being quite positive still regarding the usefulness of the game. Positive aspects of the game were discussed in the group interviews, and these can shed some light on what the students found useful. Overall, the students were positive to security and see the need for it in the general case. They explained that they learned many things from playing (see Figure 4). This included knowledge about security (assets; attack surface; easy to overlook security issues). However, other more general insights were more often pointed out, such as gaining experience in group discussions, making decisions, coming to consensus etc., and that they learned things about their own software projects and how it was understood by other group members.

Six questions together cover the variable *perceived ease of use*. Overall the responses to these questions are positive, and increasingly so towards the end of the course. To illustrate, in the end only one student found Protection Poker to be difficult to learn (question 9), as opposed to 25 finding it easy, and the majority of the students ended up finding Protection Poker to be clear and understandable (question 10, 22 students) and easy to use (question 12, 23 students). The observations and group interviews support these results on perceived ease of use. As can be seen from Table 1, many aspects of the game were observed to be easy to do in most groups. Students in the group interviews responded that the game was easy to learn and that it was easy to understand the results (see Figure 4). Still, the students we interviewed longed for more help and improved guides to support their playing of the game. Thus, some aspects of the game were still hard. This will be covered in more detail in the following subsection.

4.2 Lessons learned and improvements

Lessons learned and improvements (RQ2) were studied through observations and group interviews. We identified two main areas where improvements are needed; the discussions, and the scores and scale used.

Discussions and speed Many of the students in the group interviews responded that they liked the discussions they had while playing Protection Poker,

and found them useful (see Figure 4), e.g., that it made everybody participate, and uncovered differences in opinions and understandings of the system. However, discussions proved to be challenging as well, leading to students in the group interviews expressing concerns that: 1) some players end up with too much influence due to their personality; 2) difficulties in reaching consensus results in fighting instead of a common understanding; and 3) it may take a while.

A few students expressed the concern that, when they were not able to agree, this could cause problems later on. In the sessions where we facilitated them when playing Protection Poker, the facilitator was quite active in supporting the students in reaching a consensus. In the group interviews, students stated that they appreciated that the facilitator did this. However, the one group that had played Protection Poker on their own after the session we facilitated explained the following: *"We noticed that when we played the second time (...) we realised that we had to evaluate some things again, because that had either been cut off, because we had some issues where we we did not agree and had to stop the discussion."* Students additionally expressed the concern that, although they spent a lot of time discussing, they lacked confidence that they arrived at the "right" result, and were unsure how long their result was valid.

A main difference between student responses in the group interviews and the observation results is that where students were mainly concerned with conflicts and the influence of people with strong personalities, the observation notes additionally contain a concern regarding passive participants. Though Protection Poker requires everybody to participate by putting out cards, this does not require all students to be active in the discussions. Especially in one group, it was very hard to get the discussion going (*"a bit like pulling teeth"* (quote from observation notes)). This group was characterised by both dominating and passive members. The passive member seemed disinterested, did things on the computer while playing, and did not offer input to the discussion although specifically asked by the facilitator about the reasoning behind the card (justification generally referred to as "gut feeling"). From observation notes, half of the groups (3) were characterised either by dominant or passive participants, something that negatively influenced the general mood while playing.

Students did not have any clear suggestions for improvements that directly address the challenges of having both good discussions and ensure the playing of Protection Poker does not takes up too much time. However, one suggestion that addresses this challenge partially is to have fewer cards, and thus a more coarse-grained scale. In addition, students wanted more support on the security, in form of what to discuss and how to ensure they were on the right track. Both the response from the students after the sessions and our own observations suggest that it would have been very difficult for the students to start using Protection Poker without an external facilitator that could help on the game and bring in software security competence. The group interview feedback on need for support included guides, e.g. on finding assets, how to prepare for the session, and examples of what can happen, and ways to ensure the presence of security expertise. Though the need for an external facilitator was clearly

expressed by the students, it is important to add that the group that had tried out the Protection Poker technique on their own after the supported session reported that this had gone very well, and in some ways better since they did not have to explain the system to someone external.

Scales and scores Both in the observations and the group interviews we identified challenges relating to the scale used. These challenges were of two main types: understanding the relative scale, and understanding the concepts asset value and exposure.

In the group interview, one student expressed that he liked the relative scale as it made it possible to get a prioritisation for the project despite security not being that relevant for them. However, a general feedback from the students in the group interview was that the relative scale was difficult to understand; *"You did not know if a '100' was Armageddon or it was just "we need to look into this"."* When we did observations, the relativity of the scale only seemed to be a major problem in one of the groups. There one student viewed one of the assets (the one we had calibrated as a '100') as so much more important than anything else and thus had the opinion that every other asset would belong in the '<10' category. Thus, the student wanted an exponential scale. Due to time issues, we had decided to skip calibration of the lower end-point of the scale ('<10'), something that may have contributed to their challenges in relating to the scale.

For assets it was sometimes difficult to know how to assess their value, as the value may be different if you consider just confidentiality than if you include other aspects of its value as well. Another challenge is related to how to divide up assets in a way that is consistent and does not impact the scores in a negative way. Students in one group pointed out that if you have assets at different levels of granularity this may skew the scores. One feature with many assets of low granularity may get a higher score, and thus priority, than a feature that has assets with a higher granularity. In the observations, questions on assets and on exposure were common, and these terms were often mixed up in the discussions (e.g. "the exposure of the asset" and "the value of the feature"). Especially the term exposure was found difficult to describe in a good way. Still, students expressed that they found the end result was easy to interpret, that it was predictable due to the process and that it gave them a nice way to prioritise the assets of the project (group interviews, see Figure 4).

The improvements suggested by students when it comes to the scale goes in two directions: to explain the relativity of the scale better, or to change the scale. The latter suggestion was less common. Additionally, students suggested to take the time to do a full calibration, even though playing Protection Poker without doing a full calibration worked well in many of the groups (see Table 1). Assets were suggested identified in advance, in addition to providing better guidance on this aspect. The challenges related to the term exposure were more an observation than a feedback from the students, and thus students did not suggest any improvements in that respect.

5 Discussion

In the following we discuss the implications of the results of this case study when it comes to adoption of Protection Poker and its effect on the security of the software. Finally, threats to validity are identified and discussed.

5.1 Adoption of Protection Poker and gaining impact from playing

The results of this study show that Protection Poker as a technique has the potential to be adopted by developers. Many of the students would like to continue using the technique, despite its limited usefulness for security in their current project. Students expressed the wish for projects with more security, or a technique that better fit their project, e.g. stating *"now it felt like we took a game that was not meant for our project and tried to play it with our project, even though it didn't fit,"* and *"I think it is a good game, I think it works fine, but I don't think I got that much out of it as I could have, and I could have learned more about the different parts of Protection Poker and software security if I had a game or a project with more security issues."* Based on this, we could state that Protection Poker is not that useful for projects with very limited security issues, either because of very limited attack surface or few assets of any particular value. This type of projects is probably not as prominent in development companies as in our case with student projects. Still, our results point to the need for some kind of criteria to evaluate whether there is enough security issues in a project to justify the effort needed to play Protection Poker.

If adopting Protection Poker, there is the issue of how often Protection Poker should be played in a project. In this study, playing Protection Poker did take quite a lot of time and unless there are important security issues for many of theproject's features, the effort needed to play Protection Poker for every iteration may not be justified. Many of the main benefits from the technique as identified in this study, such as new perspectives, increased security awareness, teamwork, etc, is not dependent on playing Protection Poker for every feature.

Identifying and prioritising assets was something that the students in general found useful, however there were challenges associated with doing this as part of playing Protection Poker: it took time and there was the concern that if this was not done at a similar granularity for the whole project this might skew the prioritisation one ended up making through the game. It is an open question whether identification of assets could benefit from being an activity that is decoupled from the actual playing of Protection Poker. It is possible that if asset identification is done beforehand for the project as a whole, this may speed up the playing - especially in the beginning - and help increase confidence in the results. An asset identification task may additionally be useful input to a decision on whether or not Protection Poker is a game that would suit the project.

Due to team-dynamics issues, the teams experienced the playing of Protection Poker quite differently. Protection Poker initially aims to support good discussions, and the voting involved when putting out a card is a way to ensure that all team members' opinions are made visible. However, the goal of reaching

a consensus is not realistic in many settings. Teams need to be aware that this is challenging, and not necessarily a strict goal. Though it is not beneficial to have everybody always agree, playing Protection Poker with participants that *never* agree, or always need to be right, is challenging. Based on the results from this study one can assume that how well Protection Poker will perform in a team is highly influenced by the team-dynamics, something that should be considered when deciding whether or not to adopt the technique in a development project. Additionally, since no particularly support in software security is built into the technique, teams that decide to use Protection Poker need to have at least one person that is knowledgeable about software security to have some confidence in the resulting risk estimations.

In this case study, we did not directly measure the impact of playing Protection Poker on the security of the end-product. However, responses from students in the questionnaire and in group interviews show that the students themselves assessed the impact on security from playing Protection Poker to be limited. This is, at least in part, due to limited security needs in four out of the six projects, as projects with security needs seemed to experience more of an impact from playing. Still, also for these groups there does not seem to have been a major impact on the end-product. It is thus important to consider what can be done to increase the security impact of playing Protection Poker.

We do not know what factors that potentially made it difficult to use the results from the game in the development. In this case study, it was up to the students to use the results from playing in any way they found fit. We did not follow up on how they used the results, and provided no specific guidance on how to do this. One potential issue is the limitation pointed out by one student in the group interview that the game does not include anything on *how* something can be attacked and *how* such an attack can be mitigated. If students lack this knowledge it can be difficult to understand what can be done to reduce the risk associated with what they consider high risk functionality in the software. The results of playing Protection Poker is a prioritisation, something the students found to be an important benefit (see Figure 4), however turning this prioritisation into actual development tasks is not necessarily straight forward, especially if the rationale for the scores is lost due to limited note taking (see Table 1).

5.2 Threats to validity

The threats to validity (and the following text) is similar to what has been identified in a parallel study previously [28]. In the study it is difficult to separate the effect of the technique itself from other factors, such as motivation, skills, group dynamics, and our influence as researchers. In particular, having researchers act as facilitators constitutes a threat to validity that may influence the process and the results and make the study harder to replicate. We have aimed to be aware of the impact of the context throughout the study. One way we did this is by having the first author be supervisor of one student group. Additionally, we made sure we reflected on our role as researchers and took this into account in the analysis (reflection on our influence as researchers was part of the template

for observation notes). As part of this, we made it clear for students that their opinion on Protection Poker would not have any impact on their grade in the course. We as researchers did not have any influence on the grades the students got, except for giving some input to evaluators for the group where the first author acted as supervisor.

This study involves students, and thus not professional software developers. There are studies available that show that students in the later parts of their studies can be used with success in studies instead of professional software developers in some cases, namely for understanding dependencies and relationships in software engineering [11] and for requirements selection [24]. The topic of this study is related to, but not identical to, those studies. We do not claim that the results from our study can be generalised to software developers in general, but believe it to be likely that many of the same issues that we found would apply also in professional settings, in particular since many professionals in small and medium sized development organisations would also be considered novices when it comes to Protection Poker and have limited software security training [12]. However, the context would be different. Although the students in our study did have an external customer and the aim of the course is to have a setting that is as similar as possible to a real development project, the students had some concerns that professionals would not have (e.g. the report and getting a good grade) and this may have impacted the results. Their development projects were also likely to be simpler and with fewer security concerns than what many professional developers would likely encounter.

As explained in Section 3.2, we made some changes to the original Protection Poker game before this study regarding terminology (exposure) and scale. The issues that we aimed to address with these changes are however still difficult; students mixed exposure and asset value, and the scale was found to be difficult to relate to for some students. We do not know how the students would have responded to the original version of Protection Poker, but at the same time we do not have the reason to believe it is our changes that is the source of the problem. Rather, it confirms our initial concerns that led to the changes and points to the need for further improvements on these issues.

6 Conclusion

Protecton Poker is a technique that includes key security requirements engineering activities in a way that is particularly suited to agile development. This study of Protection Poker has identified both benefits and challenges regarding this particular technique, as used in capstone development projects. The technique led to good discussions on security, increasing awareness and knowledge about security in the team. In this study, however, its impact on the security of the end-product seem to have been limited. Improvements may be needed on the scale and on the terminology used. Having a facilitator that is knowledgeable about security and skilled in team work is likely to be a key to success in use of the game.

Acknowledgment

This work was supported by the SoS-Agile: Science of Security in Agile Software Development project, funded by the Research Council of Norway (grant number 247678). Thanks to the course organizers (Prof. Jon Atle Gulla and Prof. John Krogstie) and the participating students at NTNU. Thanks to Prof. Pekka Abrahamsson and Prof. Laurie Williams for input on the study design.

References

1. Ajzen, I., Fishbein, M.: Understanding attitudes and predicting social behavior. Prentice-Hall (1980)
2. Baca, D., Boldt, M., Carlsson, B., Jacobsson, A.: A novel security-enhanced agile software development process applied in an industrial setting. In: 10th International Conference on Availability, Reliability and Security (ARES). pp. 11–19. IEEE (2015)
3. Beck, K.: Extreme programming explained: embrace change. Addison-Wesley Professional (2000)
4. Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K., Kruchten, P.: Extending XP practices to support security requirements engineering. In: Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems. pp. 11–18. ACM (2006)
5. Caroli, P., Caetano, T.: Fun Retrospectives - Activities and ideas for making agile retrospectives more engaging. Leanpub, Layton (2015)
6. Cockburn, A., Highsmith, J.: Agile software development, the people factor. Computer **34**(11), 131–133 (2001)
7. Davis, F.D.: A technology acceptance model for empirically testing new end-user information systems: Theory and results. Ph.D. thesis, Massachusetts Institute of Technology (1985)
8. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS quarterly pp. 319–340 (1989)
9. Dybå, T., Moe, N.B., Mikkelsen, E.M.: An empirical investigation on factors affecting software developer acceptance and utilization of electronic process guides. In: 10th International Symposium on Software Metrics. pp. 220–231. IEEE (2004)
10. Grenning, J.: Planning poker or how to avoid analysis paralysis while release planning. Hawthorn Woods: Renaissance Software Consulting **3**, 22–23 (2002)
11. Höst, M., Regnell, B., Wohlin, C.: Using students as subjects – a comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering **5**(3), 201–214 (2000)
12. Jaatun, M.G., Cruzes, D.S., Bernsmed, K., Tøndel, I.A., Røstad, L.: Software security maturity in public organisations. In: Lopez, J., Mitchell, C.J. (eds.) Information Security, Lecture Notes in Computer Science, vol. 9290, pp. 120–138. Springer International Publishing (2015)
13. Jaatun, M.G., Tøndel, I.A.: Covering your assets in software engineering. In: The Third International Conference on Availability, Reliability and Security (ARES). pp. 1172–1179. Barcelona, Spain (2008)
14. Jaatun, M.G., Tøndel, I.A.: Playing protection poker for practical software security. In: The 17th International Conference on on Product-Focused Software Process Improvement (PROFES 2016) (2016)

15. Khaim, R., Naz, S., Abbas, F., Iqbal, N., Hamayun, M., Pakistan, R.: A review of security integration technique in agile software development. International Journal of Software Engineering & Applications **7**(3) (2016)
16. Li, L.: A critical review of technology acceptance literature. Department of Accounting, Economics and Information Systems, College of Business, Grambling State University. (2008)
17. Nicolaysen, T., Sassoon, R., Line, M.B., Jaatun, M.G.: Agile software development: The straight and narrow path to secure software? International Journal of Secure Software Engineering (IJSSE) **1**(3), 71–85 (2010)
18. Odzaly, E., Greer, D., Stewart, D.: Agile risk management using software agents. Journal of Ambient Intelligence and Humanized Computing (05 2017)
19. Oueslati, H., Rahman, M.M., ben Othmane, L.: Literature review of the challenges of developing secure software using the agile approach. In: 10th International Conference on Availability, Reliability and Security (ARES). pp. 540–547. IEEE (2015)
20. Peeters, J.: Agile security requirements engineering. In: Symposium on Requirements Engineering for Information Security (2005)
21. Pohl, C., Hof, H.J.: Secure scrum: Development of secure software with scrum. arXiv preprint arXiv:1507.02992 (2015)
22. Renatus, S., Teichmann, C., Eichler, J.: Method selection and tailoring for agile threat assessment and mitigation. In: 10th International Conference on Availability, Reliability and Security (ARES). pp. 548–555. IEEE (2015)
23. Savola, R.M., Frühwirth, C., Pietikäinen, A.: Risk-driven security metrics in agile software development-an industrial pilot study. J. UCS **18**(12), 1679–1702 (2012)
24. Svahnberg, M., Aurum, A., Wohlin, C.: Using students as subjects – an empirical evaluation. In: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. pp. 288–290. ACM (2008)
25. Tavares, B., Silva, C., Diniz de Souza, A.: Risk management analysis in scrum software projects. International Transactions in Operational Research (03 2017)
26. Terpstra, E., Daneva, M., Wang, C.: Agile practitioners' understanding of security requirements: Insights from a grounded theory analysis. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). pp. 439–442. IEEE (2017)
27. Tøndel, I.A., Jaatun, M.G., Meland, P.H.: Security requirements for the rest of us: A survey. IEEE software **25**(1) (2008)
28. Tøndel, I.A., Oyetoyan, T.D., Jaatun, M.G., Cruzes, D.: Understanding challenges to adoption of the Microsoft Elevation of Privilege game. In: Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS '18). pp. 2:1–2:10. ACM (2018)
29. Vähä-Sipilä, A.: Product security risk management in agile product management. Stockholm, Sweden (2010)
30. Venkatesh, V., Davis, F.D.: A model of the antecedents of perceived ease of use: Development and test. Decision sciences **27**(3), 451–481 (1996)
31. Weir, C., Rashid, A., Noble, J.: Developer essentials: Top five interventions to support secure software development (2017)
32. Williams, L., Gegick, M., Meneely, A.: Protection poker: Structuring software security risk assessment and knowledge transfer. In: International Symposium on Engineering Secure Software and Systems. pp. 122–134. Springer (2009)
33. Williams, L., Meneely, A., Shipley, G.: Protection poker: The new software security game. IEEE Security and Privacy **8**(3), 14–20 (2010)

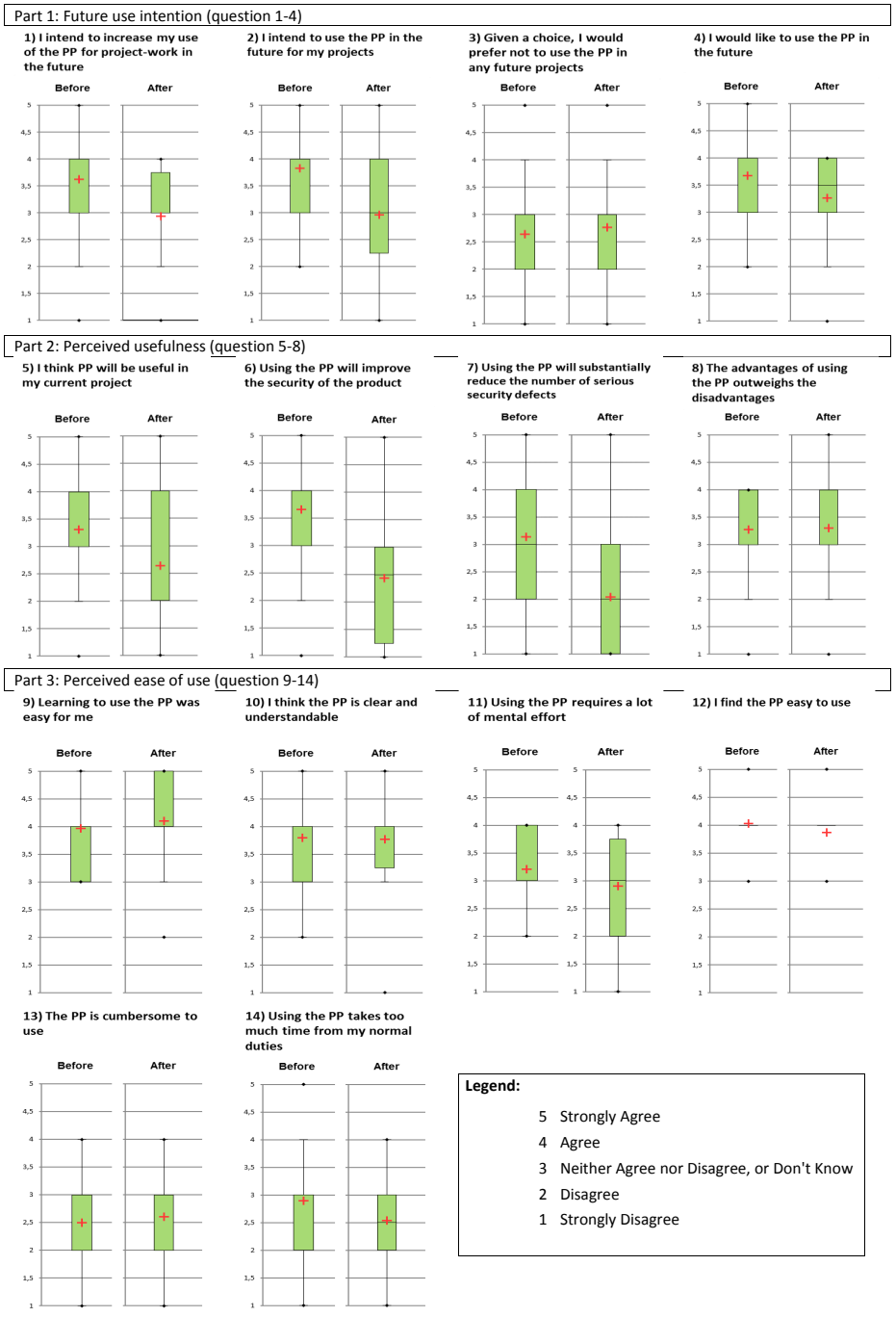

Fig. 5. Result from TAM-based questionnaire