Practitioners' Perspectives on Security in Agile Development

Steffen Bartsch
TZI, Universität Bremen
Bremen, Germany
Email: sbartsch@tzi.org

***Abstract*—Agile methods are widely employed to develop high-quality software, but theoretical analyses argue that agile methods are inadequate for security-critical projects. However, most agile-developed software today needs to satisfy baseline security requirements, so that we need to focus on how to achieve this this level for typical agile projects. In this paper, we provide insights from the practitioner's perspective on security in agile development and report on exploratory, qualitative findings from interviews. Our findings extend the theoretical prior work and suggest to focus on adequate customer involvement, developer security awareness and expertise, and continuously improving the development process for security.**

***Keywords*-Agile development; Secure software development; Security requirements; Developer awareness; Empirical study**

I. Introduction

Agile development [3], [16] is reported to have disadvantages related to secure software development [4], [13]. Heavy-weight security practices are not as applicable as in plan-driven development and may reduce the benefits of agile methods. However, the analyses primarily target security-critical contexts. Conversely, we need to understand the interrelation of security and agility also for projects with baseline security requirements, such as typical Web applications, for which agile development is often employed. Accordingly, rather than asking whether agile methods are adequate, we need to focus on how to improve the security in typical agile contexts.

Prior work on security in agile development primarily analyzes the challenges and mitigation theoretically [4], [13]. In contrast, this paper focuses particularly on the practitioner's perspective to expand the scope. Aiming to understand the effects of agile development concerning security, one question is how developers and customers interact on security. We further study what implications the security awareness and expertise of developers have and how they employ security practices. In this paper, we report on interviews with agile practitioners on these and related questions and discuss mitigations after giving a brief overview of prior findings on agile development and security in agile development.

II. Agile development

Major characteristics of agile development [3] include the iterative and incremental development, reflective process improvement, customer participation, and a high level of communication [16]. There are numerous empirical studies on effects of agile development, although the quality of the evidence is challenged [10].

Quality and productivity: Studies on agile methods and specific agile practices indicate increased software quality and productivity with no changes at worst [10]. Surveys, case studies, and experiments reported mixed [25], neutral [21], and positive effects [24], [27], [34].

Motivation: According to Highsmith, agile developers are motivated because the process reflects how software is actually being developed [16]. Although often limited in scope, the existing studies, such as surveys [22], [26] and case studies [18], indicate higher satisfaction and motivation.

Communication: Case studies show positive effects of agile practices on internal communication between the developers, both informal and formal [27], [28]. Concerning the external communication between developers and the customer, case studies indicate improved common understanding [28] and a reduction in defects due to the more informative communication media [20].

Learning: The increased communication can also lead to improved sharing of knowledge and learning. A survey of students in an academical setting found that agile development helped them to learn the development skills [24]. Also, the sharing of tacit knowledge has been indicated to increase in agile development [2].

III. Security in Agile Development

A. Challenges

In theoretical publications, authors have studied how compatible agile methods are with security practices, for example, contrasting XP with SSE-CMM and Common Criteria (CC) [33] or generally with secure software development practices [4]. As shown in Table I, the challenges from literature can be grouped into three areas. *Process life-cycle challenges* concern conflicts with the dynamics of agile methods: Dynamic process changes hinder process audits [29] and the iterative nature of the processes result in difficulties with in-depth assurance activities [4], [13].

The second area are the *communication and interaction* patterns in agile development: The focus on functional requirements and their elicitation result in neglected non-functional security requirements [14], [6] and a lack of trace-

Table I
CHALLENGES IN LITERATURE AND MENTIONS IN THIS STUDY

	Literature	Mentions
Process life-cycle model		
Process dynamics	[29]	
Emergent requirements, design changes	[9], [8], [4], [13]	6
Security is difficult to retrofit	[7], [33]	
Assurance does not fit with life-cycle model	[4], [13]	1
Communication/interaction		
Neglected non-functional security reqs.	[14], [6]	
Missing requirements traceability	[29]	
Missing assurance objectivity	[4], [13]	1
Lack of/outdated security documentation	[13], [4]	2
Lack of customer awareness/sec. reqs.	–	5
Trust in team and individuals		
Implicit trust in developers	[13]	
Lack of specialist expertise on team	[13], [14], [33]	
Neglected practices from pressure	–	1

Table II
MITIGATIONS IN LITERATURE AND MENTIONS IN THIS STUDY

	Literature	Mentions
Process life-cycle		
Defining and auditing the process	–	3
Adapting the process for security	–	3
High-level security architecture up-front, early/late security sprint	[31], [4], [12]	3
Iterative, incremental security design	[1], [7]	6
Individual sec. activities per sprint	[31]	–
Close customer integration	–	6
Security requirements		
Abuse cases	[19], [5], [15], [23]	–
Security requirements traceability	[29]	–
Explicit risk analysis	[1]	2
Implicit security requirements	–	7
Concrete requirements and threats	–	6
Non-functional reqs. as done-definition	–	2
Design, implementation, assurance		
Secure design and implementation	[4]	2
Test cases/code as documentation	[33]	2
Automatic sec. testing, static analysis	[4], [19], [32], [11]	4
Security review meeting	[19]	2
Implicit/explicit code reviews	–	4
Formal change and integration proc.	[4]	2
Early secure deployment	[19]	–
Security awareness and expertise		
Prevalent security	–	7
Implicitly and explicitly spreading awareness and expertise	–	6
Rotating experts on team	[33]	1
Security training	[12]	2
Self-teaching security expertise	–	4
Holistic development approach	–	5

ability of requirements back to security goals [29]. Cross-functional teams and close customer participation are also seen critically, since this brings evaluators and developers closer together and endangers their objectivity [4], [13]. Third, the *trust in team and individuals* is difficult to align with the skeptic approach of traditional security processes and the fear of malicious developers [13]. Also, security expertise may be missing without the explicit security expert role in teams [14].

B. Mitigations

Together with the challenges of security in agile development, a host of mitigations for the identified problems have been proposed, mostly based on analytical or laboratory work (cf. Table II). One type of approaches modify the *process life-cycle* and, for example, introduce explicit security iterations [31] or emphasize the iterative and incremental nature of security design [1], [7]. A second type of approaches targets the elicitation of *security requirements*, for instance, through abuse cases. More common are proposals of adopting security practices for *design, implementation, assurance* [4]. A fourth set of enhancements fosters the *security awareness and expertise* of developers through security trainings [12] and security experts who rotate through programming pairs [33].

The listed challenges and mitigations in literature are either from theoretical analysis or laboratory experiments, and primarily target explicitly security-critical environments. However, to understand the actual problems and mitigations, an exploration of how security is addressed in typical agile development projects is still missing.

IV. STUDY DESIGN

The goal of this study is to expand on the theoretical findings on security-critical agile development through an exploration of the challenges and their mitigations in typical agile development projects. Since there is little prior work on this area, we chose an exploratory approach and conducted semi-structured interviews. While limited in sampling size, the interviews allowed us deeply explore the challenges and mitigations in practice. Based on the prior work, we identified the following areas to cover in the interviews:

- *Customer involvement*: Due to its theoretical nature, prior work is thin on the implications of customer involvement, particularly, how customers relate non-functional and functional security requirements.
- *Developer security awareness and expertise*: Prior work recommends security training for developers. How security-aware are developers typically and where do expertise and awareness originate?
- *Effects of "agile" on security*: Which effects do practitioners see from agile methods and practices?
- *Security practices*: Several security practices are suggested to improve security in agile development. Are they present and how well do they integrate?
- *Authorization*: Authorization is a particularly dynamic security measure [30]. How is it affected by agile development?

We conducted the sampling of the participants in a way to represent a wide variety of agile development contexts. Sampling dimensions included the interviewee's process role and project characteristics, such as team size and development platform. Table III lists the ten interviewees from nine

Table III
INTERVIEWEE DETAILS

Pseudonym	Role	Affiliation type	Project characteristics (customer, team, platform)
Moritz	Developer	Medium development company	External customer, 7 developers, JavaEE
Ben	Developer	Medium development company	External, 3–6 developers, Ruby on Rails
Max	Developer	Small development company	External, 4–6 developers, Rails
Herbert	Developer	Small development company	External, 3 developers, Rails
Stefan	Developer	Medium-sized company	Internal, safety-critical, 11 developers, C#
Niels	Developer	Medium-sized company	Internal, security-critical, 10 developers, JavaEE
Wolfgang	Developer	Consultant developer in medium-sized company	Internal, 4–9 developers, Rails
Günther	Coach	Consultant	Internal, ca. 1000 developers
Klaus	Customer CEO	Medium-sized company	External, Business processes, 1–3 developers, Rails
Jan	Business analyst	Consultant, in medium-sized organization	External, 3 dev., 2 QA at customer, Delphi/ASP.NET

companies. We directly contacted the interviewees primarily through agile development meet-ups. The interviews lasted between 30 and 60 minutes. Based on detailed notes on each interview, we structured the statements by common concepts iteratively with each interview and clustered related aspects to derive the findings on challenges and mitigations (see Tables I and II for an overview, which includes the number of interviewees who mentioned each concept).

V. FINDINGS

A. Unclear security requirements

The interviewees portrayed the security awareness among customers as very heterogeneous, ranging from marginally to highly aware. Five interviewees mentioned problems related to the lack of security awareness. Ben noted that the security awareness at the customer "depends on the specific person that you talk to." Interviewees generally observed that the awareness on the customer side is often driven by specific values at risk and requirements and threats need to be discussed in concrete terms (6 mentions). Max: security "is only of interest [to the customer] when money-aspects are concerned". Security incidents involving similar systems increase the customer's high-level security awareness (Moritz: "the shop leak at [newspaper], that's where customers wake up"). Security requirements are discussed, "if at all, then only unspecific, together with functions" (Klaus) and are characterized as being approached from the positive side ("what is allowed" instead of "what is forbidden").

Jan's case is an exception in that the customer was very security-aware and developed very specific security requirements because the developers were rather unaware.

B. Implicit security requirements

Often, developers derive the security requirements from the functional requirements as implicit security requirements (7). For example, Moritz' development team has good domain knowledge in electronic commerce and can therefore propose adequate security requirements. For Herbert's team, the developers' best practices led to agreements between the customer and developers to minimize risks and outsource payment data directly to the service provider. According to Klaus, the trust relationship between the customer and the development team is very important in this respect. Non-technical customers often cannot comprehend the technological basis of each security measure. The necessary level of trust is built up with the close development participation. The trust on the technological level is similar to the trust on the inner quality of the software product, for which customers usually assume that the developers conduct adequate quality assurance measures.

C. Close customer involvement

Six interviewees mention that, irrespective of the customer's security awareness, close customer participation improves the security requirements elicitation with their domain knowledge. The security requirements are usually refined over several discussions and iterations (6 mentions). For example, developers propose technical approaches, which are discussed with the customer, then implemented and adapted in subsequent iterations. In Jan's project, authorization requirements were complicated to elicit bottom-up with large variations among the customer stakeholders, causing overly complex requirements. The simplified, top-down model that was implemented instead then needed to be adapted in production over time, with changes even after 1.5 years in production. Another reason for discussions are functional changes: Moritz reports that over the course of the project, functional changes, such as new interfaces between the ERP and the shop system, repeatedly require security discussions. In his experience, adaptations to the original plan are mostly necessary when the best practices of the developer team do not fit as expected in the customer context.

The style of customer interaction on authorization varies. A permission matrix can be appropriate, but more common are natural language discussions without a document as basis. Two interviewees mention that missing or outdated security documentation is a problem. In Jan's case, a five-page document is necessary to describe the high-level authorization concept used in the application in addition to seven separate lists with the actual permissions that are maintained in parallel to the actual permission configuration.

D. Awareness and expertise spreads

Seven interviewees describe security as being generally present within the developer teams and expertise as being homogeneous. Exceptions are interns and new employees who "need some time to take up the necessary security awareness" (Max). There are several ways in which the security awareness and expertise is built. Generally, security expertise and awareness spreads between developers (6 mentions) and security is part of informal discussions. According to Moritz, "there is exchange of information between the people of the team and between the teams, 'how have you handled the problem?' " Self-taught awareness and expertise from the news and blogs was also mentioned frequently (4). One reason is the motivation of feeling "responsibility for the project" due to the holistic development approach (5). Holistic development also increases the awareness and expertise: New developers, for example, rotate through multiple project teams to work on various parts of a project. Comparing the motivation to pre-agile development, Stefan states that the improved communication among developers and quality assurance helped. Other motivating factors are external audits and the inner-company competition for quality between teams.

Less common are explicit trainings for formal security requirements (2). Other institutionalized knowledge sharing occurs through security review meetings (2). Moritz also reports on coding Dojos (collective development sessions), in which developers train specific quality-improving agile practices, such as test-driven development.

E. Developing securely

Three interviewees mention that there are problems of integrating assurance into the agile life-cycle or of neglected assurance practices from the pressure of short iterations. The full-stack tests can be more difficult with agile processes because the effort is too high for rigorous testing on each iteration. Short iterations can cause pressure to make visible progress, in some cases even causing practices, such as test-driven development, to be neglected.

Two interviewees state that the iterative and incremental development allows them to find more direct approaches and keep the software design simpler as recommended in secure software design. The design also benefits from the holistic development approach (5) since it offers a more complete picture for the individual developers. If non-functional requirements, such as quality and high-level security aspects, should be made explicit, they can be formulated as "done definitions" (2) that specify when a feature is considered complete.

F. Employing low-overhead assurance

In most cases, the assurance practices are integrated into the development process. Four interviewees mention testing and test-driven development, typically conducted by the development team. In contrast, the customer took over most of the testing activities in Jan's project to guarantee the necessary product quality. Penetration tests were conducted in two cases, either as part of the done criteria or in parallel to the development on the production system. Code reviews are more common (4), for example, as a four-eye principle on each software repository check-in or as part of the done criteria. Three interviewees report that the practices are adapted over time to changing contexts, so that, for example, the code review practices evolved with the company and development processes.

G. Adapting the process model to high-criticality

In three cases, the processes are defined and externally audited in parallel to development to assure the process quality. The same three projects also employ dedicated security or release iterations before the actual release to create the necessary documents and thoroughly test the product. Explicit risk analysis and management is mentioned by two interviewees, either assessing the risk for individual features before deployment or following regulations on a systematic risk management process, including explicit risk meetings and sign-off procedures.

VI. Discussion

We interviewed agile practitioners to explore the problems and mitigations in practice. The interviews offer only subjective data and are prone to researcher or participant bias. Since the sample size is limited for interviews, we focused on covering a broad range of development contexts. The results are, by study design, not sound and representative, but extends the prior theoretical findings with a practical perspective and offer a description as an initial hypothesis for further research. We discuss the key results in the following:

Institutionalize customer involvement: Despite not being covered in prior work, a good interaction concerning security between developers and customers is a prerequisite to achieve adequate security measures. In this study, the interaction is very heterogeneous among projects and customers can often only state unclear security requirements, leading to implicit security requirements. Non-functional requirements need to be elicited together with customers and then iteratively turned into functional requirements. Thus, *customers and developers need a common understanding of the roles in the project regarding security*, including:

- Which side has how much security awareness and expertise,
- Who is responsible for triggering discussions of non-functional and functional security requirements.

Risk assessments are often only implicitly part of functional discussions. While not necessarily as explicit security iteration as suggested in literature [31], *customers and developers should explicitly address the risks and the non-functional security requirements early in the project* to have a common

framing for the later decisions on security measures and implementation practices. *Developers should also focus on concrete threats and security requirements to improve the common understanding* [35].

Foster developer security awareness and expertise: The overall security in a project depends on the security expertise of the individuals, either on the customer or developer side. This corresponds to the agile value of "individuals and interaction over processes and tools" [3]. The developers in this study for the most part feel responsible to craft secure systems and are motivated to learn and spread security expertise, also supported by the communicative nature of agile processes. Nevertheless, this cannot be taken for granted. *The spreading of awareness and expertise among developers should thus be further supported.* Not only through explicit security trainings [12] and experts on teams [33], as suggested in literature, but also by motivating the exchange between developers, for example, through agile practices with positive effects on internal communication.

Continuously improve the process for security: Three teams in this study are actively adapting the development processes over time to optimally fit the environment. While this has not been the focus in literature, *developers should consider the security level of the development process in retrospectives and adapt it to meet security needs.* Retrospectives are also a good place to consider the integration of secure development practices, for example, from SSE–CMM [17]. Three cases in this study show that it is possible to integrate heavy-weight security practices, in contrast to the skeptical prior work.

Promote implicit documentation: Literature and two cases in this study point to the problem of outdated security documentation in agile development. Not only test cases [33], but also other artifacts, such as, authorization policies can provide implicit documentation. *Developers should consider to move the security documentation into the code where possible* to create current documentation as the basis for discussions with customers.

VII. Conclusion

We interviewed ten agile development practitioners in order to improve the understanding of security in actual agile development projects. Despite its limited sample size and the subjectivity of the interview method, the study offers a more practical perspective than the theoretical prior work. The studied cases show problems with the customer involvement and emergent requirements. However, several challenges and mitigations from literature, such as the lack of specialist expertise or abuse cases, are not relevant to the studied cases. Conversely, approaches not discussed in prior work, such as implicit security requirements and the spreading of expertise and awareness among developers, are helpful for the studied cases. Our recommendations expand on existing mitigations to build upon the strengths of agile development for improved security in projects of average security-criticality.

Acknowledgment

I sincerely thank all the interviewees in this study who reported openly about the sensitive topic of security.

References

[1] E. G. Aydal, R. F. Paige, H. Chivers, and P. J. Brooke. Security planning and refactoring in extreme programming. In P. Abrahamsson, M. Marchesi, and G. Succi, editors, *XP*, volume 4044 of *Lecture Notes in Computer Science*, pages 154–163. Springer, 2006.

[2] B. Bahli and E. S. A. Zeid. The role of knowledge creation in adopting extreme programming model: an empirical study. In *Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on*, pages 75–87, 5-6 2005.

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development. Online, retrieved 10 Jul 2010, 2001.

[4] K. Beznosov and P. Kruchten. Towards agile security assurance. In *NSPW '04: Proceedings of the 2004 workshop on New security paradigms*, pages 47–54, New York, NY, USA, 2004. ACM.

[5] G. Boström, J. Wäyrynen, M. Bodén, K. Beznosov, and P. Kruchten. Extending xp practices to support security requirements engineering. In *SESS '06: Proceedings of the 2006 international workshop on Software engineering for secure systems*, pages 11–18, New York, NY, USA, 2006. ACM.

[6] L. Cao and B. Ramesh. Agile requirements engineering practices: An empirical study. *Software, IEEE*, 25(1):60–67, jan.-feb. 2008.

[7] H. Chivers, R. F. Paige, and X. Ge. Agile security using an incremental security architecture. In H. Baumeister, M. Marchesi, and M. Holcombe, editors, *XP*, volume 3556 of *Lecture Notes in Computer Science*, pages 57–65. Springer, 2005.

[8] N. Davis. Secure software development life cycle processes: A technology scouting report. Technical Report CMU/SEI-2005-TN-024, CarnegieMellon, December 2005.

[9] B. De Win, R. Scandariato, K. Buyens, J. Grégoire, and W. Joosen. On the secure software development process: Clasp, sdl and touchpoints compared. *Information and Software Technology*, 51(7):1152–1171, 2009.

[10] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859, 2008.

[11] G. Erdogan, P. H. Meland, and D. Mathieson. Security testing in agile web application development - a case study using the east methodology. In W. Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, C. Szyperski, A. Sillitti, A. Martin, X. Wang, and E. Whitworth, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 48 of *Lecture Notes in Business Information Processing*, pages 14–27. Springer Berlin Heidelberg, 2010.

[12] X. Ge, R. F. Paige, F. Polack, and P. J. Brooke. Extreme programming security practices. In G. Concas, E. Damiani, M. Scotto, and G. Succi, editors, *XP*, volume 4536 of *Lecture Notes in Computer Science*, pages 226–230. Springer, 2007.

[13] K. M. Goertzel, T. Winograd, and P. Holley. *Security in the Software Lifecycle: Making Software Development Processes–and the Software Produced by Them–More Secure*. DHS, August 2006. Draft Version 1.2.

[14] K. M. Goertzel, T. Winograd, H. L. McKinley, L. Oh, M. Colon, T. McGibbon, E. Fedchak, and R. Vienneau. Software security assurance: A state-of-art report. Technical report, Information Assurance Technology Analysis Center (IATAC), 2007.

[15] J. Heikka and M. Siponen. Abuse cases revised: An action research experience. In *PACIS 2006 Proceedings*, 2006.

[16] J. A. Highsmith. *Agile software development ecosystems*. Addison-Wesley, Boston, MA, USA, 2002.

[17] ISSEA. The systems security engineering capability maturity model (SSE-CMM), model document, 2003. Version 3.0.

[18] D. Karlström and P. Runeson. Combining agile methods with stage-gate project management. *IEEE Software*, 22(3):43–49, 2005.

[19] V. Kongsli. Towards agile security in web applications. In *OOPSLA '06: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 805–808, New York, NY, USA, 2006. ACM.

[20] M. Korkala, P. Abrahamsson, and P. Kyllonen. A case study on the impact of customer communication on defects in agile software development. In *Agile Conference, 2006*. IEEE Computer Society, 2006.

[21] F. Macias, M. Holcombe, and M. Gheorghe. A formal experiment comparing extreme programming with traditional software construction. In *Mexican International Conference on Computer Science*, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[22] K. Mannaro, M. Melis, and M. Marchesi. Empirical analysis on the satisfaction of it employees comparing XP practices with other software development methodologies. In *Extreme Programming and Agile Processes in Software Engineering, 5th International Conference, XP 2004*, pages 166–174. Springer, 2004.

[23] D. Mellado, E. Fernández-Medina, and M. Piattini. A comparative study of proposals for establishing security requirements for the development of secure information systems. In M. L. Gavrilova, O. Gervasi, V. Kumar, C. J. K. Tan, D. Taniar, A. LaganÃ, Y. Mun, and H. Choo, editors, *ICCSA (3)*, volume 3982 of *Lecture Notes in Computer Science*, pages 1044–1053. Springer, 2006.

[24] G. Melnik and F. Maurer. A cross-program investigation of students' perceptions of agile methods. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 481–488, New York, NY, USA, 2005. ACM.

[25] P. Middleton. Lean software development: Two case studies. *Software Quality Control*, 9(4):241–252, 2001.

[26] D. Parsons, H. Ryu, and R. Lal. The impact of methods and techniques on outcomes from agile software development projects. In *Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda*. Springer, 2007.

[27] K. Petersen and C. Wohlin. The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering*, 2010.

[28] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, and J. Still. The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3):303–337, 2008.

[29] M. Poppendieck. XP in a safety-critical environment. *Cutter IT Journal*, September 2002.

[30] S. Sinclair, S. W. Smith, S. Trudeau, M. E. Johnson, and A. Portera. Information risk in the professional services – field study results from financial institutions and a roadmap for research. In *Proceedings for the 3rd International Workshop on Enterprise Applications and Services in the Finance Industry*, 2007.

[31] B. Sullivan. Streamline security practices for agile development, 2008.

[32] A. Tappenden, P. Beatty, and J. Miller. Agile security testing of web-based systems via httpunit. In *AGILE*, pages 29–38. IEEE Computer Society, 2005.

[33] J. Wäyrynen, M. Bodén, and G. Boström. Security engineering and extreme programming: An impossible marriage? In C. Zannier, H. Erdogmus, and L. Lindstrom, editors, *XP/Agile Universe*, volume 3134 of *Lecture Notes in Computer Science*, pages 117–128. Springer, 2004.

[34] C. A. Wellington, T. Briggs, and C. D. Girard. Comparison of student experiences with plan-driven and agile methodologies. In *Proceedings 35th Annual Conference on Frontiers in Education, 2005. FIE '05*, 2005.

[35] R. West. The psychology of security. *Commun. ACM*, 51:34–40, April 2008.