Software Engineering Institute | Carnegie Mellon®

Agile Security – Review of Current Research and Pilot Usage

Carol Woody

April 2013

OVERVIEW

This paper was produced to focus attention on the opportunities and challenges for embedding information assurance (IA) considerations into Agile development and acquisition in support of the Agile acquisition research underway at the Software Engineering Institute. U.S. Department of Defense (DoD) implementers of Agile have pointed to delays in IA approvals as one of their key frustrations. The use of Agile is providing results faster, only to have operational implementation delayed for six months or more waiting for IA approvals.

What success and challenges for embedding IA into an Agile approach have been realized elsewhere based on a literature review? How might the DoD benefit?

Leveraging Agile Capabilities for Security

The opportunity to quickly (within a few Agile iterations) address a new security flaw would seem to be an advantage of Agile over traditional system engineering approaches. The design principles in Agile support the need for reliability and robustness which should enhance the ability of the operational results to function as intended. Test-driven development supports the validation needs of effective security [10].

However, Agile projects focus on features and functionality requested by the stakeholders over infrastructure, and operational release can be hampered by security mandates tied to the infrastructure. Many of the DoD security controls mandated for IA are focused on infrastructure. Test development will of necessity be focused on features initially in support of requirements selected as high priority by stakeholders [10]. Test environments that incorporate security infrastructure mandates might mitigate this delay but involve extensive investment of time upfront which interferes with the Agile focus on stakeholder needs.

The Agile emphasis on quality code would seem to indicate a strong correlation with more secure results. Data from Capers Jones's measurement research [11] indicates that in projects of 1000 function points, Agile delivers results with .40 defects per function point which puts it well ahead of Average Quality which

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

delivers at .75 defects per function point. Best in Class delivers at .13 defects per function point. However, for larger projects of 10,000 function points the defect rate rises to .72 for Agile indicating quality results are very little better than average.

Security is not the only quality to be challenged by an Agile approach. For safety-critical systems, just as with security, the requirements cannot be simply discovered by a dialog with the customer and quickly translated into user stories for iteration allocation, but there have been successes in using Agile for safety critical mandates. The Boeing 777 program applied lean software development and negotiated stringent release criteria with the Federal Aviation Administration (FAA) that exceeded the usual testing criteria. A long-range flight certification was obtained before delivery of the initial plane, which was highly unusual. Aggressive testing at every stage of integration in partnership with the FAA with highly visible results and monitoring allowed the FAA to have confidence in the safety of the plane and issue the certification ahead of final delivery [12, page 118]. The mindset needed for this level of result emphasizes continual improvement and the recognition that "when things go wrong, the cause is almost always inherent in the system, and therefore it is a management problem…Leadership must envision and institute fundamental changes that address the systemic causes of current problems." [12, page 123-124]. Application of a similar approach to security engineering and oversight partnership in an Agile approach could address existing IA limitations and provide a strong operational impact.

Literature Review: Embedding IA into an Agile Approach

Much has been written about addressing security issues within Agile methods. One research paper reported a review of 34 papers relevant to security in Agile [1]. The survey from this report identifies critical disparities between established security methodologies and agile approaches in the areas of documentation, requirements, and testing.

Security Documentation

Security assessment mechanisms such as Common Criteria and DoD Information Assurance Certification and Accreditation Process (DIACAP) require extensive documentation, which works directly against Agile's effort to minimize the use of paperwork for its own sake [2]. The lack of documentation hampers meeting the DoD IA mandate for an independent review [5] that must rely on documented evidence to conduct the review.

In addition, this lack of documentation is viewed by security experts as a lack of compliance proof resulting in delays and possible rejection in submissions for operational certification.

Security Requirements

Beznosov proposes writing security requirements in the form of user stories, which brings security aspects into the Agile flow. However, security experts are not typically part of the stakeholder negotiations, making prioritization and trade-off decisions extremely difficult [4]. It would be most appropriate for security issues to be defined in terms of threat level and risk so that stakeholders could have an appropriate means of evaluation and prioritization. Just articulating security controls as user stories is not sufficient [6]. One project started by including security experts in the Agile team, but could not impact stakeholder priorities nor reduce delays in deployment until the risks were directly connected to desired stakeholder functionality [13].

In another project a different response was formulated by establishing collective ownership of security matters by the team and extending the iteration planning meetings to include time focused on security [7]. Another source, Aydal et al., addressed this disparity for a Web application by deferring the security checking until after all functionality was done and dedicating a few additional iterations to address the security aspects [3]. Significant development and refactoring time had to be added to the project just for security, which could have been avoided if the security stories had been part of the normal project flow.

Security Testing

Vulnerability testing has been identified as a key step in satisfying a security related activity for Common Criteria [8]. Static analysis tools that find vulnerabilities fit well with the Agile approach, but the tests can be difficult to execute, complicated to automate, and produce many false positives whose interpretation requires security expertise not always available to an Agile team. In addition, Kongsli reported that the tools went beyond the application level to include deployment environment elements that may not be under the control of the Agile team [7]. Testing requirements for information assurance involve the system as a whole and often cannot be accomplished by automatic testing at the project level. One project attempted to address this issue by adding iterations where the whole team would attempt to break the functionality [3]. While this effort could be useful in improving the quality of the fielded code, it does not sufficiently address the mandates of IA for independent validation. Independent review, and testing, of the security controls is required for certification [5]. Testing that is

incorporated into the development structure but not meticulously tracked and documented is not easily reconfirmed by external sources.

Survey - Security Engineering from Agile Perspective

Baca and Carlsson surveyed 12 Agile software developers from a large telecommunications manufacturer to evaluate, from the Agile perspective, compatibility of security engineering activities with an Agile approach. The activities under consideration were selected from the Microsoft Security Development Lifecycle (MS SDL), Cigital Touchpoints (CT) and the Common Criteria (CC) and assigned by the authors to the following areas of a project life cycle: requirements, design, implementation, testing, and release. In addition, three other security activities the authors identified as highly recommended for Agile security development (Countermeasure Graphs, Diff Review, and Pair Programming) were included in the survey. Survey participants were asked to consider the benefits relative to the cost of addressing each activity within an Agile approach. Based on the strength of preference, each security activity can be viewed as valuable, marginal, or detrimental [9].

Security Activities Considered

Selected from Cigital Touchpoints [14]:

Security Activity	Life Cycle Phase	Cost/Benefit
Security Requirements	Requirements	Very Valuable
Abuse Cases	Requirements	Valuable
Risk Analyses	Design	Detrimental
Assumption Documentation	Design	Very Valuable
Static Code Analyses	Implementation	Very Valuable
Penetration Testing	Testing	Marginal
Red Team Testing	Testing	Marginal
Risk Based Testing	Testing	Valuable
External Review	Release	Very Detrimental

Selected from Common Criteria [15]:

Security Activity	Life Cycle Phase	Cost/Benefit
Security Requirements	Requirements	Very Valuable
Agree on definitions	Requirements	Detrimental
Risk Analyses	Design	Detrimental
Critical Assets	Design	Detrimental
UMLSec	Design	Very Detrimental
Requirements Inspection	Design	Marginal
Repository Improvement	Requirements	Very Valuable

Selected from Microsoft SDL [16]:

Security Activity	Life Cycle Phase	Cost/Benefit
Security Require-ments	Requirements	Very Valuable
Role Matrix	Requirements	Valuable
Design Requirements	Requirements	Detrimental
Quality Gates	Design	Detrimental
Cost Analysis	Design	Very Detrimental
Threat Modeling	Design	Detrimental
Attack Surface Re-duction	Design	Valuable
Security Tools	Implementation	Very Detrimental
Coding Rules	Implementation	Marginal
Static Analysis	Implementation	Very Valuable
Dynamic Analysis	Testing	Valuable
Fuzzy Testing	Testing	Marginal
Code Review	Testing	Detrimental
Incident Response Planning	Release	Detrimental
Final Security Review	Release	Marginal

Other security engineering activities considered:

Security Activity	Life Cycle Phase	Cost/Benefit
Countermeasure Graphs	Design	Very Valuable
Diff. Review	Implementation	Detrimental
Pair Programming	Implementation	Marginal

Survey results indicated strong preference for security requirements (CT), Role Matrix (MS SDL) and Abuse Cases (CT) for addressing requirements. For design the preferences were the following: Assumption Documentation (CT), Countermeasure Graphs (other), Requirement Inspection (CC), and Critical Assets (CC). Preferences for handling security in implementation were Static Code Analysis (CT) and Coding Rules (MS SDL). For testing, Dynamic Analysis (MS SDL) and Risk Based Testing (CT) were supported. Only repository improvement (CC) was well supported for implementation [9].

From the nine CT security engineering activities, five were identified as appropriate for Agile. CC contains seven security engineering activities and three are identified as usable for Agile. MS SDL has fifteen security engineering activities but only six were selected by Agile professionals. In order to effectively address security engineering practices within an Agile approach, the researchers analyzing the survey results recommend consideration of an Agile life cycle specifically tailored to incorporate security engineering considerations appropriate to an Agile approach. The sample size was small, but the survey results do point out that just inserting existing security engineering activities into an Agile effort

could be unnecessarily costly and, for some activities, may detrimentally impact results.

Case Study: Federal Agency Success Story

Integration of security into Agile can be done. A Federal agency has embraced Agile for a high assurance system and has successfully maintained an operational certification [13]. New functionality must be fielded every six months and the program could not afford a 6-12 week compliance review effort for each release. Security is embedded into the Agile process, but this has required extensive change in the way information security resources interact with the project and the way in which the project addresses security. The integration is not considered complete since security and development resources operate independently within the project, but the organization is considering ways to more closely align the processes. Blending these disciplines requires extensive change in how individuals address their work, and not all existing resources can make this transition. Developers are accustomed to making decisions within their application that must now be checked with security for control compliance. Security resources are accustomed to focusing their review on controls and not relating their decisions to stakeholders and developers.

The Federal agency project in question started with the NIST 800-53 documented security control objectives and mapped them to specific versions of user stories which are labeled as “controls” to distinguish them from feature driven user stories. This identification allows them to stay visible for security monitoring and auditor review (external verification). The aforementioned security control objectives are composed in language appropriate to stakeholder understanding, allowing risks to be identified and mapped to desired system capabilities.

Agile life cycle tools supported the extensive interaction needed between the Agile developers and the security sub-teams and provided an audit trail of issues and decisions to satisfy the needs of independent reviewers. Tools supported the ability to maintain visibility of every control and its status on the project. When the security team identifies a problem with the handling of a security control, they must describe the issue in terms of risks that allow the stakeholders to make risk-informed choices. These risks are fully integrated into the project risk management processes and are addressed within the normal project decision-making structure.

The integration of IA into the Agile project has been cost effective. For example, in a Web application a security control required a five minute timeout for a Web screen, but the user stories were defined with timeouts of an hour. This dis-

crepancy was identified at the requirements review, making it much cheaper and easier to fix than at integration when security controls are normally evaluated.

Implementation of this approach did not require Federal agency policy changes and the required security documentation is still delivered to the Designated Approving Authority (DAA), but the content is built from the data assembled from the Agile tool environment instead of a separate security process. External auditors use the data in the tools to verify security controls and confirm testing results.

As with the safety example identified earlier in this paper, transparency for all project information with all project participants at all levels is key to successful implementation. Participants must partner on delivering the most effective project results and not on the specific steps of the individual development and security processes. Project management and stakeholders must make decisions as risk-informed agents. Project resources, both in development and security, must focus on communicating to decision makers all that they need to make effective choices.

Summary

There is a great deal of potential for embedding security into an Agile approach, but research indicates that existing security activities need to be adjusted to integrate effectively. Existing Agile life cycle tools can be used to include IA but appropriate structuring of input is required to ensure that IA validation requirements can be met. Security knowledgeable personnel must be added to the Agile team to take responsibility for identifying and communicating IA issues to developers and stakeholders within the Agile approach. Success requires change in the way security is addressed by program management, IA, developers, and stakeholders.

Early successes have shown that IA does not need to be a deterrent to Agile success and costs can be reduced as security issues are identified and addressed earlier in the life cycle. Stakeholders can make risk informed decisions when they are provided with information that relates security issues to the capabilities they are seeking to field. Change of this magnitude is not easy and will not come without resistance. An organization seeking to capitalize on the potential for Agile must have a commitment to success at all levels of the project.

References

[1] Alnatheer, A., Gravell, A., and Argles, D. 2010. Agile Security Issues: A Research Study. http://esem2010.case.unibz.it/idoese/docs/alnatheer.pdf
[2] Boström, G., Beznosov, K. and Kruchten, P. 2006. Extending XP Practices to Support Security Requirements Engineering. *Security*, (2006), 11-17.
[3] Aydal, E.G., Paige, R.F., Chivers, H., and Brooke, P.J. 2006. Security Planning and Refactoring in Extreme Programming. *Work*, (2006), 154 - 163.
[4] Beznosov, K. and Kruchten, P. 2005. Towards agile security assurance. *Proceedings of the 2004 workshop on New security paradigms - NSPW '04*, (2005), 47.
[5] DoD Instruction 8500.3. Information Assurance Implementation. February 6, 2003. http://www.dtic.mil/whs/directives/corres/pdf/850002p.pdf
[6] Chivers, H., Paige, R., and Ge, X. 2005. agile Security Using an Incremental Security Architecture. *Proceedings of Extreme Programming and agile Processes in Software Engineering 6th International Conference, XP 2005, Sheffield, UK*, (2005).
[7] Kongsli, V. 2006. Towards agile security in web applications. *Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications - OOPSLA '06*, (2006), 805.
[8] CC, ISO 15408 Common Criteria for Information Technology Security Evaluation Version 2.1, August 1999.
[9] Baca, D. and Carlsson, B., 2011. "Agile Development with Security Engineering Activities". *ICSSP'11*, May 21-22, Waikiki, Honolulu, HI.
[10] Martin, R. 2003. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall, Pearson Education, Inc., Upper Saddle River, NJ.
[11] Jones, C. 2012. "Software Quality in 2012: A Survey of the State of the Art." DHS Software Assurance Forum, May 28, 2012. McLean, VA.
[12] Poppendick, T. and Poppendick, M. 2007. *Implementing Lean Software Development: from concept to cash*. Addison-Wesley.
[13] Brown, D., and Woodhull, P., "Who is at the Helm? An Agile Assurance Case Study", International Conference of Software Quality (ICSQ), October 31, 2012, Indianapolis, IN.
[14] Steven, J., "Adopting an enterprise software security framework". *IEEE Security and Privac*y, Vol. 4, No. 2, 84-87, 2006
[15] Mellado, D., Fernandez-Medina, E., Piattini, M., "A Comparison of the Common Criteria with Proposals of Information System Security Requirements", The First International conference for Availability, Reliability, and Security (ARES), 2006. ARES '06. 20-22, April 2006
[16] Howard, M., Lipner, S., *The Security Development Lifecycle*. Microsoft Press, 2006

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

States Department of Defense.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.

DM-0000754