The Risks of Agile Software Development

Learning from Adopters

Amany Elbanna, Royal Holloway, University of London

Suprateek Sarker, University of Virginia

// Researchers conducted interviews with 112 people in 28 organizations and with 25 agile software development (ASD) contractors and consultants. The interviews revealed key potential ASD risks that require careful management to achieve the desired project outcomes. //

THE POPULARITY OF agile software development (ASD) continues to grow as organizations increasingly seek to quickly develop software and to cope with changes in the current fast-moving business environment. ASD encompasses many approaches, including Extreme Programming, Scrum, the Dynamic Systems Development Method, Crystal, agile project management, feature-driven development, and lean software development. These approaches might differ in terminology and practices but share many of the core values of the Agile Manifesto (http://agilemanifesto.org).[1] Generally, they aim to provide development that's responsive to both customers' and developers' needs. They also share a general orientation toward short "time-boxed" iterative development cycles, frequent communication with customers, constant adaptation, and accommodation of change.

ASD changes how development teams work (in terms of processes and team organization) and their relationship with business teams. Such changes have generated benefits and resolved many development risks. However, as with many other process innovations such as business process reengineering, ASD can also have undesirable effects.

In addition, a study showed that of 12 typical IT critical success factors, ASD approaches supported only half, with different degrees of strength.[2] This suggests that ASD projects' risk (and success) factors differ from those of traditional development environments. (For a look at other research on software development risks, see the sidebar.) Identifying and managing risk is an important area of research and good practice in software development, which views risk identification and analysis as the first step in reducing the chance of project failures.[3]

Given the differences between traditional approaches and ASD, we believe the identification of ASD risks needs a fresh view, one that doesn't rely overly on knowledge of traditional-development risks. Toward that end, we performed a study of people and organizations involved in ASD. Our study didn't provide statistical testing or compare ASD theory with organizations' practices. It aimed to provide insight based on the participants' views of ASD practices and risks. Future research should subject our findings to empirical testing using large, well-designed samples, as research in software development risk did decades ago.

Our study was guided by the following research questions: What key risks do adopters of ASD face? Why

0740-7459/16/$33.00 © 2016 IEEE

RELATED WORK IN SOFTWARE DEVELOPMENT RISKS

Susan Sherer and Steven Alter noted that many risk factors that apply to information systems (IS) operations also apply to IS projects.[1] Studies of IS risk have examined operational and project risk in general. The highlighted risks are typically related to users, such as poor communication with users and stakeholders, lack of user involvement, failure to manage user expectations, misunderstanding of user requirements, and failure to accommodate changes in requirements and scope.

Agile software development (ASD) can effectively deal with these issues.[2] For example, the short iteration cycles allow for accommodating change and clarifying users' requirements. Also, the face-to-face interaction and constant communication with users and business stakeholders improve communication and develop a common understanding of requirements.

However, recent failures of projects adopting ASD and the reported challenges associated with it[3,4] have drawn attention to its possible risks and the importance of identifying, assessing, and mitigating them.

For example, in 2013, the UK Department for Work and Pensions ceased all agile software development on its Universal Credit program after the cost escalated to nearly £12.8 billion (six times the original estimated cost) and the project failed to deliver.[5] Universal Credit is one of the largest ASD projects in the UK, so you might dismiss the unacceptable outcome as an exception. You might also suspect that ASD could have been the scapegoat for other project management problems. Nevertheless, the project continues to draw attention to ASD's possible risks.

A software development company's recent discontinuation of Scrum is yet another case highlighting risks that must be managed.[6]

References

1. S.A. Sherer and S. Alter, "Information System Risks and Risk Factors: Are They Mostly about Information Systems?," *Comm. Assoc. Information Systems*, vol. 14, no. 2, 2004, pp. 29–64.
2. R. Baskerville and H. Pries-Heje, "Short Cycle Time Systems Development," *Information Systems J.*, vol. 14, no. 3, 2004, pp. 237–264.
3. B. Tessem, "Experiences in Learning XP Practices: A Qualitative Study," *Extreme Programming and Agile Processes in Software Engineering*, LNCS 2675, Springer, 2003, pp. 131–137.
4. D. Rosenberg and M. Stephens, *Extreme Programming Refactored: The Case against XP*, Apress, 2003.
5. M. Ballard, "Why Agile Development Failed for Universal Credit," *Computer Weekly*, 4 July 2013; www.computerweekly.com/news/2240187478/Why-agile-development-failed-for-Universal-Credit.
6. P. Ralph and P. Shportun, "Scrum Abandonment in Distributed Teams: A Revelatory Case," *Proc. 2013 Pacific Asia Conf. Information Systems* (PACIS 13), 2013, paper 42; http://aisel.aisnet.org/pacis2013/42.

do these risks arise, and what are their consequences? How do different organizations deal with them?

Conducting the Study

We adopted a qualitative, interpretive approach, which is particularly useful when researchers seek to understand people's experience. Gaining such insight would have been difficult with "experience-far" approaches such as simulation, modeling, or surveys.

We collected data primarily through interviews. Interviews have many strengths (for example, in-depth, holistic coverage; being close to practice; and the opportunity to discover patterns) and weaknesses (for example, bias owing to the selection of interviewees, faulty recall, the social desirability of responses, and the respondents' political agendas). We used established guidelines to minimize the limitations.

Our results are based on an investigation of the adoption of ASD in 28 organizations, four of which we examined longitudinally over several years. Table 1 presents an overview of the organizations.

We interviewed 112 people at those organizations: development team members, project managers, IT directors, business managers, and users. Thirty-nine interviewees let us record their conversations so that we could transcribe them. If interviewees refused to let us record their interviews, we took notes instead. When feasible, at the end of those interviews, we presented a summary of the conversation and key points to the interviewees for validation or clarification.

We analyzed the data in three cycles.[4] In the first cycle, we identified the key risks the participants experienced. In the second cycle, we grouped those risks by their nature

TABLE 1

An overview of the studied organizations.

Sector	No. of companies	Purpose of development			
		Internal use	External customer	Commercial	Internal and commercial
Utility	2	1	—	1	—
Transportation	3	3	—	—	—
Telecommunication	3	—	—	2	1
Media	2	2	—	—	—
Financial services	7	7	—	—	—
Software company	9	—	7	—	2
Local administration	1	1	—	—	—
Travel and leisure	1	1	—	—	—
Total	28	15	7	3	3

and origin. Finally, we refined them through cross-case comparisons. The risks reported here were prominent in at least four cases; we don't report the risks found in three or fewer cases, owing to the nature of the article and space constraints.

Besides the organization stakeholders, we interviewed 17 freelance developers and eight ASD consultants. Triangulating the interviews helped us assess the findings' reliability and enhance their validity.

Also, one of us attended nine business-oriented conferences or workshops focusing on ASD from 2007 to 2013. At these meetings, she presented parts of this study's findings and spoke with other participants. The conversations with and feedback from those participants, who weren't involved with the studied organizations, helped ensure the findings' face validity and external validity.

The Findings

Table 2 lists the key ASD risks we identified, the number of organizations experiencing those risks, and how the organizations dealt with them.

Development and Deployment Risks

The key development and deployment risks fell into the following categories.

Technical debt. Technical debt is the visible and invisible results of past decisions and short-term compromises about software that create complexities and could negatively affect the software's future.[6] This concept originally referred to coding practices; it now includes architectural, testing, or documentation debt.

Although technical debt was initially discussed in the context of traditional software development,[7] ASD can intensify technical-debt concerns. In ASD, debt can quickly accumulate owing to the need to significantly reduce development time, adhere to strict time boxing, and constantly deliver functional requirements for business use. This accumulation can lead to unexpected project delays and lower software quality as the software becomes more complex, less understandable, and thus more difficult to maintain. (Ward Cunningham, who coined the metaphor of technical debt, asserted, "A little debt speeds development so long as it is paid back promptly with a rewrite. The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt."[8])

One form of technical debt is security debt, which can increase the risk of security breaches and data corruption. Security debt often accumulates when developers conscientiously attend to critical and

TABLE 2

Key agile software development (ASD) risks in the 28 studied organizations.

Risk category	Type of risk	No. of organizations experiencing the risks	How organizations dealt with the risks
Development and deployment	Technical debt	23	List the outstanding technical issues, and convince management to allocate one or more iteration cycles to address them.
			Address only the most important technical issues that are short term and could immediately impact users.
			Under business pressure, ignore accumulated technical debt and continue development until issues surface and require attention.
	Separation of development and IT operations	21	Ensure that one ASD team member is from IT operations. Invite IT operations teams to ASD planning sessions and end-of-iteration demonstrations.
			Allocate an extra week in each iteration cycle to address IT operations issues.* (In one case, for a highly sensitive system, a large financial-services firm allocated two extra weeks to a typical two-week iteration cycle.)
	Increased defects in new ASD teams	13	Allocate experienced senior developers to ASD teams.
			Allocate at least one experienced developer to each ASD team.
			Provide on-the-job training and support to deal with developers' uneasiness with ASD, particularly those lacking ASD experience.
Project management	Unstandardized project management tools	17	Audit all ASD teams for their project management and software-testing tools, and discuss standardization.
			Negotiate limiting the number of tools while continuing to accept some variations.
			Don't consider this risk a problem as long as the ASD teams deliver on target.
			Specify standard tools and request all teams to comply. (However, ASD teams who valued autonomy resisted this approach.)
	Lack of knowledge retention	20	Use wikis to document project progress and explain key decisions.
			Create and use software code libraries, and encourage code reuse.
			Document important technical specifications and key technical decisions.
			Encourage the use of social media among ASD team members and across teams.

** Richard Vidgen and Xiaofeng Wang showed that each team needs to find its own pace in each specific context.[5] A team also needs to understand what pace it can sustain over time. A suitable pace strikes a balance such that the iteration cycle is long enough to accomplish meaningful work but short enough not to lose momentum and adaptability.*

high-level vulnerabilities but ignore or postpone the consideration of medium- and low-level vulnerabilities, leading to their accumulation. It's generally impossible to have zero security debt; the objective should be to identify and formulate a repayment plan.

The technical teams we studied understood the dangers of accumulating technical debt. However, they also explained that the pressure of delivering in short time-boxed iteration cycles made them take shortcuts to speed up delivery. This correlates with Mike Cohn's statement, "Technical debt is often the result of a rushed implementation."[9]

Furthermore, they highlighted the difficulties of logging in nonfunctional requirements as user stories in iteration cycles, and how the business-oriented stakeholders often deprioritized those requirements. (This might correlate with Viktoria Stray and her colleagues' findings in which an organization used daily Scrum meetings to justify continued investment in a failing course of action, leading to project setbacks.[10]) This problem might be due to the business participants' lack of comfort with "technical jargon" and to their single-minded focus on implementing functional requirements.

In summary, the respondents felt that although the business stakeholders' close involvement in ASD helped mitigate some traditional-development risks (for example, poor communication with users and stakeholders, lack of user involvement, and failure to manage user expectations), it often significantly contributed to technical debt. Nevertheless, the inability to appreciate or understand something's importance could occur because either the source (in this case, the IT staff) didn't communicate well or the receiver (the business staff) couldn't see the value.

The separation of development and IT operations. As the ASD and business teams are brought close together in the ASD environment, the ASD teams' work becomes tightly coupled with the business teams' needs, requirements, and timeline. However, the development work becomes increasingly separated from the IT operations teams. These teams include systems administrators, network engineers, and infrastructure specialists who ensure the software's stability and reliability in its operational environment. This separation is amplified in ASD owing to the teams' different work practices and pace.

This separation has five main consequences. The first is tension in the relationship between ASD and IT operations teams. This tension typically surfaces as ASD teams find that operations teams respond too slowly, whereas IT operations teams find ASD teams' push for continuous delivery too stressful as the IT operations teams try to cope with the provisioning of applications and infrastructure tasks.

This conflict occurs because ASD teams' pace challenges IT operations teams' work rhythm and priorities. For example, ASD teams work and deliver in short iteration cycles, demanding much parallelism in work and roles. This is inconsistent with the work of IT operations teams, which function rather linearly, on the basis of logging requests in queues and responding to them systematically. So, ASD teams' operations requests are typically met in different iteration cycles than the teams require.

The DevOps community has recognized the development and operation silos as it tries to find techniques to consistently address this problem.[11] Researchers have also mentioned the tension between ASD and other organizational functions such as quality assurance and strategic alignment.[12]

The second consequence is that ASD teams aren't fully aware of the IT operations environment. The IT operations teams' lack of communication and involvement with the ASD environment means that development and testing occur in environments that could differ from the operating or production environment. For example, development in one of the studied organizations used a particular version of a database technology, but the developers discovered later that the production environment used a different version. So, they couldn't implement the developed application on schedule, and preparing it for the operating environment's current version involved considerable rework.

The ASD environment lacks the mechanisms to deal with deployment and operability issues. ASD teams' inability to involve IT operations teams can delay production and deployment. One organization we studied tried to solve this problem by dedicating a sprint to operation issues; they planned it as a major user story in the project backlog early in the project. Another organization required ASD teams to involve at least one IT operations team member throughout each project.

The third consequence is lack of knowledge of deployment and operability. As technology advances, developers don't necessarily have the required operational knowledge and experience. The increasing use of off-the-shelf components, particularly in Web development, requires specialists to operate them. Web development includes components such as webservers, application containers, networking, and databases that aren't integrated with the software. Nevertheless, these components must be configured and maintained, which requires considerable effort. These tasks are beyond many developers' capabilities, so ASD team management must carefully oversee them.

Although such a risk occurs in most development environments, the ASD environment accentuates it by narrowing the ASD teams' focus to delivering working software with the desired business functionality. Testing, especially unit testing, test-driven delivery, and user acceptance testing, is geared to ensure the software's functional requirements and general performance.

The fourth consequence is increased firefighting. In our study, IT operations teams felt that the way ASD teams performed their work necessitated much firefighting to stabilize the systems and make them available for use. ASD teams struggle to balance developing new features and handling the project backlog—dealing with the rework requested in defect reports or enhancement requests. Our study participants reported that ASD teams usually focused on developing new features instead of handling rework requests. This could lead to emergency patches and significantly more work downstream for both ASD and IT operations teams.

The final consequence is a business push for new features. Resolving infrastructure and networking operational issues is necessary for systems to operate in their live environment. However, when developers focus on business needs, important technical issues such as servers or network upgrades receive insufficient priority. So, they don't get included in the iteration cycles until the end, when ASD teams start thinking of the going-live phase. This can cause unanticipated delays in the overall project. One team member expressed his disappointment: "I am afraid that the time we save in development is actually wasted in operations."

Increased defects in new ASD teams. In smaller organizations and especially in early-adoption projects, the number of defects rose beyond the organizational average. However, the defect rates approached acceptable levels as the ASD team matured.

The participants believed that developer experience and quality were more important in the ASD environment than in the traditional environment. According to 13 organizations, the product quality depended largely on the developers' quality and their experience in the ASD environment. Developers who lacked ASD experience tended to perform poorly on their first ASD projects.

Reflecting on her early days with ASD, a developer recalled her frustration: "When I came here, they were doing agile; I found myself spending my time talking and not actually doing [development]. … Now I can see it works." Another developer, whose first project used Scrum, reflected that "thinking of the project I'm on at the moment, … we had to put in complete defect management around it … as an afterthought."

As technology advances, developers don't necessarily have the required operational knowledge and experience.

Project Management Risks

The key project management risks fell into these categories.

Unstandardized project management tools. ASD advocates self-organized teams, which offers the teams considerable freedom and autonomy and can be motivating for many team members. So, when developing work routines, ASD teams tend to choose the project management and testing tools they prefer. This could lead to an unstandardized toolset across the enterprise, with negative consequences.

For example, in 19 of the organizations we studied, ASD teams had moved away from the traditional whiteboard and sticky notes. Many developers found these tools inconvenient because notes tended to fall down and lose their order. Others found it cumbersome to keep moving the sticky notes across the whiteboard. Moving away from the past concern of "which sticky note is the stickiest,"[13] they used agile-project-management software. The choice of software depended largely on their knowledge and preferences. In many organizations, the lack of standardized software caused much confusion. IT managers reported that merging teams and managing workloads became challenging in such conditions.

Another example involved version control systems and user acceptance testing. In one organization, ASD teams used eight version control systems and four user-acceptance-testing protocols. Being autonomous, ASD teams felt that "we use what we're comfortable with; they can use whatever they want and are happy with." Such an unstandardized toolset resulted in the company's loss of negotiation power with tool vendors and of the potential economies of scale in licensing. It also impaired knowledge sharing across projects and teams and was disorienting for operations team members, who had difficulty finding systems-related data.

Lack of knowledge retention. ASD often privileges face-to-face communication over written documentation. This implicitly assumes that team members will stay on a team throughout a project and that team members developing a particular application will remain with the organization to oversee development of later versions of that application. These assumptions don't necessarily hold. We found that in multiproject environments, shuffling and reassignment of team members were common.

Our study also revealed that the team velocity and quality tended to drop noticeably when the team got a new member. In addition, a high turnover in IT personnel means it's unlikely that all the development team will be involved with subsequent versions of an application. So, developing those versions could take longer, as the new team tries to unravel the logic behind the code and understand why certain development decisions were made in a past iteration.

Many of the risks we described aren't self-evident and thus should be consciously assessed and carefully managed throughout the life of ASD projects. In highlighting these risks, this article offers a word of caution and highlights issues for organizations to think about as they decide whether to, and how to, embrace agile approaches.

ABOUT THE AUTHORS

AMANY ELBANNA is a senior lecturer in information systems at Royal Holloway, University of London. One of her main research interests is IT project and operations management. Elbanna received a PhD in information systems from the London School of Economics and Political Science. She's a member of the Association of Information Systems, UK Academy of Information Systems, and International Federation for Information Processing Working Group 8.6—Transfer and Diffusion of IT. Amany is a member of the editorial boards of *Information & Management*, the *Journal of the Association of Information Systems*, *Information Technology & People*, and the *Journal of Enterprise Information Management*. Contact her at amany.elbanna@rhul.ac.uk.

SUPRATEEK SARKER is a professor of commerce at the University of Virginia's McIntire School of Commerce and a part-time chair of technology and information management at Royal Holloway, University of London. He also holds a Visiting Distinguished Professorship at Aalto University's School of Business. His research interests are qualitative research, theory development, mixed-methods research, systems development, distributed work, and IT-enabled organizational change. Sarker received a PhD in information systems from the University of Cincinnati. He's the editor in chief of the *Journal of the Association for Information Systems*, a senior editor of *Decision Sciences Journal*, a senior editor (emeritus) of *MIS Quarterly*, a member of the *Journal of Management Information Systems* board of editors, and an editorial board member of *IEEE Transactions on Engineering Management* and *Information Technology & People*. He received the Operational Research Society's Stafford Beer Medal. Contact him at suprateek.sarker@comm.virginia.edu.

Acknowledgments

This article is a significantly enhanced version of "Identifying the Risks Associated with Agile Software Development:

An Empirical Examination."[14] Amany Elbanna acknowledges that a grant from the British Academy (SG-4835), Loughborough University, and Royal Holloway, University of London, supported this research.

References

1. K. Conboy, "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development," *Information Systems Research*, vol. 20, no. 3, 2009, pp. 329–354.
2. T. Chow and D.-B. Cao, "A Survey Study of Critical Success Factors in Agile Software Projects," *J. Systems and Software*, vol. 81, no. 6, 2008, pp. 961–971.
3. R. Schmidt et al., "Identifying Software Project Risks: An International Delphi Study," *J. Management Information Systems*, vol. 17, no. 4, 2001, pp. 5–36.
4. M. Miles, A. Huberman, and J. Saldana, *Qualitative Data Analysis: A Methods Sourcebook*, 3rd ed., Sage, 2014.
5. R. Vidgen and X. Wang, "Coevolving Systems and the Organization of Agile Software Development," *Information Systems Research*, vol. 20, no. 3, 2009, pp. 355–376.
6. P. Kruchten, R.L. Nord, and I. Ozkaya, "Technical Debt: From Metaphor to Theory and Practice," *IEEE Software*, vol. 29, no. 6, 2012, pp. 18–21.
7. E. Lim, N. Taksande, and C. Seaman, "A Balancing Act: What Software Practitioners Have to Say about Technical Debt," *IEEE Software*, vol. 29, no. 6, 2012, pp. 22–27.
8. W. Cunningham, "The WyCash Portfolio Management System," *OOPSLA '92 Addendum to Proc. Object-Oriented Programming Systems, Languages, and Applications*, 1992, pp. 29–30.
9. M. Cohn, *Succeeding with Agile: Software Development Using Scrum*, Pearson Education, 2010, p. 321.
10. V.G. Stray, N.B. Moe, and T. Dybå, "Escalation of Commitment: A Longitudinal Case Study of Daily Meetings," *Agile Processes in Software Eng. and Extreme Programming*, Springer, 2012, pp. 153–167.
11. D. Spinellis, "Don't Install Software by Hand," *IEEE Software*, vol. 29, no. 4, 2012, pp. 86–87.
12. B. Fitzgerald et al., "Scaling Agile Methods to Regulated Environments: An Industry Case Study," *Proc. 35th Int'l Conf. Software Eng.* (ICSE 13), 2013, pp. 863–872.
13. "Which Sticky Note Is the Stickiest?," Stack Exchange, 2011; http://pm.stackexchange.com/questions/3219/which-sticky-note-is-the-stickiest.
14. A. Elbanna, "Identifying the Risks Associated with Agile Software Development: An Empirical Examination," *Proc. 2014 Mediterranean Conf. Information Systems* (MCIS 14), 2014; http://aisel.aisnet.org/mcis2014/19.