Embedding Reflection and Learning into Agile Software Development

Jeffry Babb, West Texas A&M University
Rashina Hoda, University of Auckland
Jacob Nørbjerg, Aalborg University

// The theoretical underpinnings of agile methods emphasize regular reflection as a means to sustainable development pace and continuous learning, but in practice, high iteration pressure can diminish reflection opportunities. The Reflective Agile Learning Model (REALM) combines insights and results from studies of agile development practices in India, New Zealand, and the US with Schön's theory of reflective practice to embed reflection in everyday agile practices. //

IMPLICIT IN THE agile tenet—"at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly"[1]—is the fact that agile teams should regularly assess their process and its outcomes. Individuals and teams are expected to engage in regular reflection to maintain and improve their software processes.[1–5] Using reflective practice, agile developers can determine the degree to which the process, or aspects of it, can be expanded, adapted, altered, or abandoned.[3] Thus, agile software development projects are expected to consistently yield results in both areas of productivity (regular delivery of working software) and the reflection required to maintain an effective software process. However, in practice, this balancing act between productivity and continuous process improvement and learning is nontrivial to achieve and sustain.[3] Individuals and teams often perform under sustained pressure to deliver customer value (iteration pressure) and might abandon or under-engage in core agile practices related to reflection and learning, leading to diminished opportunities for knowledge sharing, learning, and reflection.[3,6] Without an ongoing reflective dialog about their practices, software teams may experience process erosion.[7]

In this article, we present the Reflective Agile Learning Model (REALM), showing where and how to integrate reflective practice in agile software development. The model combines insights and results from industrial studies of agile development practices in India, New Zealand, and the US with Schön's theory of reflective practice.[8–10] Although a theoretical synthesis of agile methods and Donald Schön's reflective practice isn't a novel suggestion,[2,5,11,12] our approach underscores the effectiveness of reflective practice to sustain and improve an agile development team. REALM provides concrete guidelines for embedding reflective practice into an iterative and agile software engineering

0740-7459/14/$31.00 © 2014 IEEE

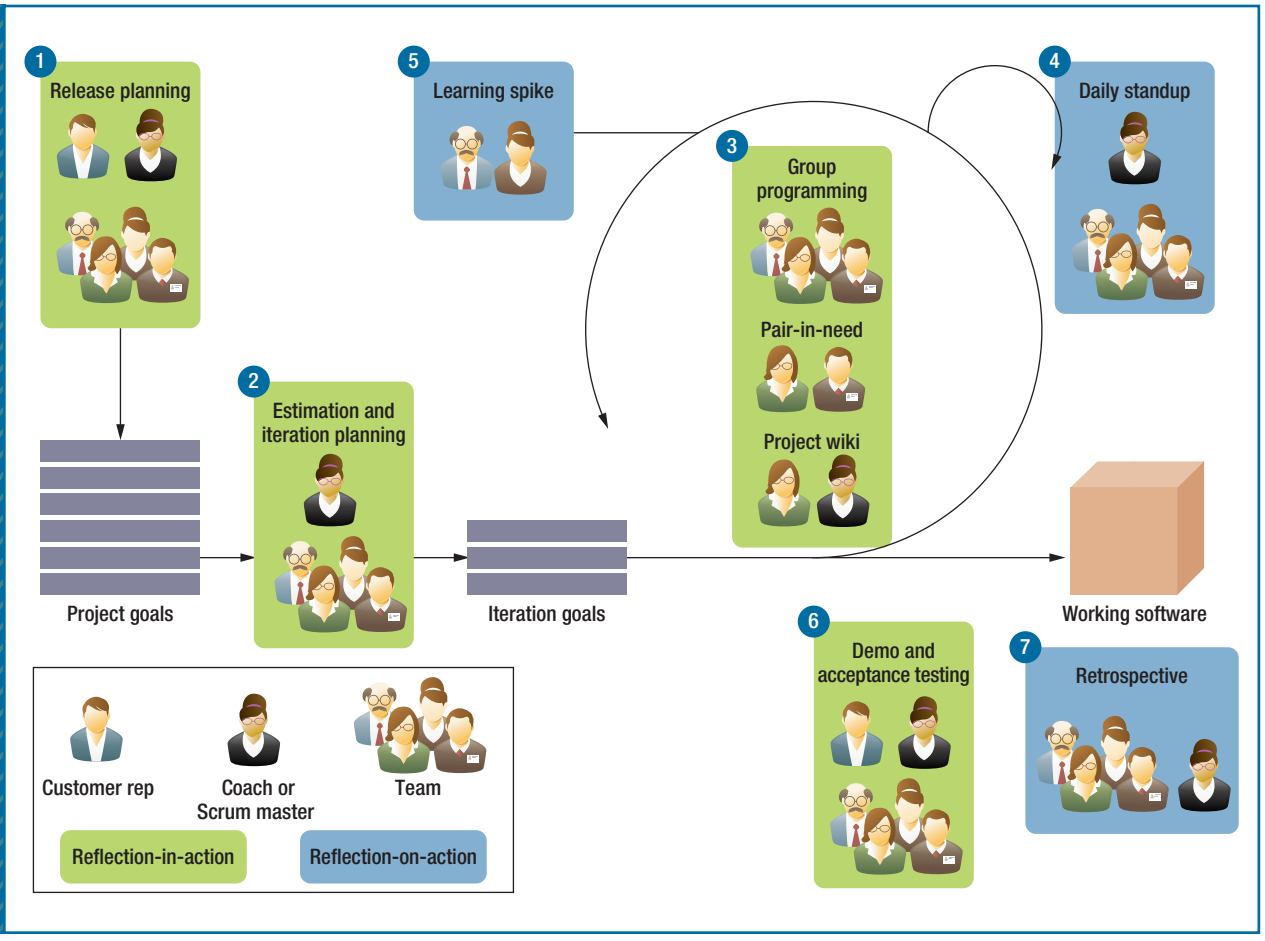

FIGURE 1. The Reflective Agile Learning Model (REALM). Standard and adapted agile practices with implicit reflection-in-action opportunities are depicted in green; learning-focused agile practices with explicit reflection-on-action opportunities are depicted in blue.

development cycle. This model is an evolution of previous models but has been refined to present a reflective process maintenance perspective.[3,6]

Drawing upon Evidence

We created REALM by drawing upon evidence from two industrial-studies of agile practices: a longitudinal Action Research study of agile software development in a small software company in the US, and a Grounded Theory study of software teams and practices involving 58 software practitioners from 23 different companies in New Zealand and India, most of whom were using agile methods such as Scrum, extreme programming (XP), or a combination of the two. Both studies involved observations of workplaces and practices plus direct face-to-face interviews of team members as a means to collect qualitative data.

A prominent finding emerging from both studies independently was the teams' adaptations of standard agile practices to suit particular organizational or project contexts. In particular, we found that the practices most likely to be lost, sacrificed, or compromised over time were those most closely related to learning and reflection.[3,6]

A model for the infusion of reflective practice into XP was a salient outcome of the US study. Developers engaged in daily learning and reflection habits that reinforced and complemented activities already endemic to agile methods. We refined and augmented REALM via the learning-focused practices of the Scrum and XP teams in the Grounded Theory study. The findings from both studies complement one another and combine to inform a new model of reflection in agile practice.

TABLE 1

Learning and reflection opportunities in the Reflective Agile Learning Model (REALM).

Agile practices	Reflection-in-action
Release planning	Learning about the project, domain, and product
Collective estimation and iteration planning	Learning about each other's skills and areas of expertise, and the team's collective capacity to delivery (velocity)
Group programming	Learning new skills and expertise from each other
Pair-in-need	Novices learning about the team and project as they pair with experts; pairs learning a new technology or skill as they work on a challenging task
Project wiki	Capturing project-based learning about the domain and individual reflections (blogs)
Demo and acceptance testing	Learning from acceptance test failures that fuel refinements and guide product vision
Agile practices	**Reflection-on-action**
Daily standup	Reflection on action performed the previous day, planning for the day's actions, and highlighting impediments
Learning spike	Adapted practice that embodies reflection on action as some members devote exclusive time to new learning
Retrospective	Standard agile practice devoted exclusively to reflection on actions performed in previous iteration

A New REALM of Application

As part of a learning team, a reflective practitioner is one who engages in two separate but intertwined reflective processes: *reflection-in-action* concerning the individual responses to shaping the problem at hand and its possible solutions, and *reflection-on-action* concerning a post hoc evaluation process for extending and validating individual and team repertoire. Schön defines repertoire as an accumulation of ideas, examples, situations, and actions gleaned from the whole of a practitioner's experience such as it is accessible for understanding and action.[9]

At first, this might seem either too trivial or too obtuse to systematically guide a software process. However, without ongoing and explicit attention to reflection, even an agile development team can lose its ability to learn and improve. Thus, the purpose of REALM is to routinize awareness of ongoing reflection in and on action in agile development projects. The model's development is strongly grounded in the reflection-in and -on action practices identified across the two studies. Figure 1 illustrates how REALM highlights opportunities for reflection-in-action, which is implicit in standard and adapted agile practices (green); reflection-on-action includes explicit learning-focused agile practices (blue). Table 1 summarizes the model's steps.

Reflection-in-Action

A practitioner who's more conscious of her reflection-in-action will be more apt to take note of her reactions and solutions to new problems in a manner that can inform her repertoire. This is useful both to the practitioner and to the team she works with. In REALM, these reflections are most evident in release planning, estimation and iteration planning, group programming, pair-in-need, project wikis, and product demo and acceptance testing. During these agile activities, developers are most engaged with the materials of design and development.

Release planning. In release planning, overall domain context, project goals, and product features are discussed, which drives the tempo and direction of team practices throughout the project. Any clues that a project might reveal about past contexts and experiences will become evident to the developer at this stage. The reflective practitioner, with heightened awareness, could be more apt to draw from repertoire at this point.

Collective estimation and iteration planning. The team collectively selects and estimates tasks for the current iteration, and in so doing, team members acquire and confirm

knowledge of each other's skills and areas of expertise. Learning about the team's collective capacity to deliver (velocity) starts here. Including the whole team—not just developers or leads, but stakeholders, too—ensures a shared vision and understanding of the project based on inputs from multiple perspectives and collective agreements. Teams can use this as an excellent opportunity to discuss customer expectations with regard to team productivity, which is a key element in managing learning throughout the project. The reflective practitioner will look for metaphors and patterns to allow past experience to guide and clarify task formulation and prioritization.

> Individuals and teams need to constantly add to and draw from their repertoire.

Group programming. Software development acutely draws on tacit knowledge and repertoire. The working rhythm and velocity of the team will be established, and any corrections and adjustments, particularly those made "on the spot," will most benefit from reflection-in-action. Overtly and tacitly, habits, skills, and knowledge pass between team members in a revelatory manner in terms of project trajectory and risk mitigation.

Pair-in-need. Agile teams might not pair-program all the time in practice. However, they do pair up on an as needed basis such as when a newcomer to the team pairs with an experienced member to learn "the tricks of the trade." Other instances include team members pairing up to learn and apply a new technology or skill to solve a challenging task.

Project wikis. Teams often capture project-based terminology and context-specific conventions in a wiki format shared between the team and the customer. Individuals might also engage in writing blogs to capture and share learning gained from project experiences.

Product demo and acceptance testing. Here the team can again draw on its collective repertoire, as it must adjust to the customer's interaction with and feedback regarding the software's emerging functionalities. Although agile methods call on teams to constantly involve the customer, the customer can rarely immerse into the same "zone" as the developers at the height of the development process, making for a constant translation process. Reflection-in-action at the moment of iteration demonstration and acceptance will have an appreciable impact on the next iteration and the nature of the product.

Reflection-on-Action

Reflection-on-action is the "after-action report" on how repertoire has changed. It's during reflection-on-action that teams attempt to answer questions regarding what to keep, discard, and modify in repertoire. Opportunities for reflection-on-action in REALM are the retrospective, the daily standup, and learning spikes.

Retrospective. While most agile methods call for a retrospective, in practice, it's an activity that can easily be skipped or skimped. Retrospectives provide a means of examining the thoughts, experiences, and notes on the various consultations with and examinations of both individual and team repertoire. This is a process of making the tacit explicit by asking of our actions, "What did/does this mean?" Therefore, reflection-on-action in general, and in particular during the retrospective, is both vital and challenging.

Daily standup. Repertoire is mostly called on during the "action" stages of a project: a daily standup meeting allows the practitioner and the team to recall recent reflection-in-action while it's still fresh. This daily routine constitutes a critically important component for engaging in reflective practice to benefit process maintenance. Throughout daily practice, practitioners on the team will ideally accumulate observations and reflections. Thus, the daily standup is an opportunity to aggregate reflections to reinforce or challenge team and individual repertoire. During the daily standup, the team can discuss project velocity and story burndown, as well as develop a platform for observations on repertoire use and understanding.

Learning spikes. Another test of repertoire is the development of proof-of-concept "spike" solutions to determine individual and team ability to branch out into new techniques, tools, or knowledge. Spike solutions

enhance individual and team repertoire via experiential learning.

Practical Examples

During the US study, the software team worked over a nine-month period to make reflection-in-action a habit and coerce reflection into an explicit activity. Among other items, they adopted the habit of on-the-spot reflection-in-action by "microblogging" in a team wiki to accumulate their thoughts and observations. At the end of the day, the developers also contributed to a team blog for reflection later that night or the next morning:

> *"I put it in the wiki right then, I knew I wanted to come back to it later. There have been a lot of times I knew I wanted to come back to something that I looked at six months ago, but I couldn't remember where I found it, so I don't want to lose this knowledge." – small shop owner and lead developer, Virginia, US*

This explicit process of reflective practice was slowly ingrained into the team by habits engaged in throughout the day. During development, particularly when pair-programming, developers would consciously remember to record tidbits of reflective writing in their team wiki. This habit of microblogging was used to "feed" the daily standup with a synopsis of interesting observations and other helpful items. This activity became a team expectation to be engaged in equally alongside more traditional activities that gauge project velocity, coordinate task completion, and so on. These daily habits further supported the project retrospectives (reflection-on-action) that formalized the development of individual and team repertoire. Because iteration pressure to deliver maximum customer value as quickly and efficiently as possible was quite high, reflection-in-action helped team members annotate their experiences in the hope that their body of knowledge could be tested, verified/validated, and documented:

> *"I think that the blogging, the wiki, and the daily standup are very important ways to keep an active project on track. We have so many projects going on that if we do not follow up, either through the daily standup or through the wiki, the projects will get off track." – developer, Virginia, US*

The blogs and wikis used by the US team certainly aren't novel tools, but the way in which they facilitated the habits necessary to understand the tacit-to-explicit process of reflective practice was. Daily and regular engagement of these tools provided the team with a means of raising awareness and consciousness of how individual practitioners framed, shaped, and set problems in their daily work.

In the Grounded Theory study, mature teams tended to use standard practices such as retrospectives effectively to drive action based on reflections on previous iteration:

> *"With every retrospective, we certainly came up with ideas to improve our process, and I think with all those retrospective sessions under our belt, with all the experience sizing, planning, everything combined, it really made us evolve as a team." – business analyst, NZ*

Reflections during retrospectives led to teams introducing adapted practices such as the learning spike to concentrate on exclusive learning. For example, a team in New Zealand, as a result of a retrospective session, discovered that manual testing was placing high demands on team productivity. As a result of team discussions and with management support, they agreed to conduct a learning spike where team members took exclusive time off project work to create an automated test suite:

> *"We've just basically reduced our velocity and taken the time to do those things because we knew they were important. We made a call that we were not going to wimp out, and go back to the manual testing...we've taken automation a lot further... doing 100 percent automation." – agile coach, NZ*

While the spike led to a reduction in project velocity in the short-term, the team clearly benefitted from automating the testing in the long-term across all projects. Lessons from this experience were captured and shared by the agile coach as presentations to the local agile community.

Pair-in-need was a common practice used across teams in India and New Zealand to boost learning while benefitting from investing two minds on a particularly challenging problem:

> *"The way we do it is that if things are unpredictable, we always take up user stories as a pair … if something [task] requires—this is complex, this is design-intensive—we sit together and pair it." – agile coach, India*

While newer teams (with less than a year of experience) in the Grounded Theory study struggled to embed learning in everyday practice

ABOUT THE AUTHORS

JEFFRY BABB is an assistant professor of computer information systems at West Texas A&M University. His research interests include small-team software development, agile software development, and mobile application development. Babb received a PhD in information systems from Virginia Commonwealth University. Contact him at jbabb@wtamu.edu.

RASHINA HODA is the director of the Software Engineering Processes, Tools and Applications research group in the University of Auckland's Department of Electrical and Computer Engineering. Her research interests include self-organizing teams, agile software development, and human-computer interaction. Hoda received a PhD in computer science from Victoria University of Wellington. Contact her at r.hoda@auckland.ac.nz.

JACOB NØRBJERG is an associate professor in Aalborg University's Department of Computer Science. His research interests are in systems development organization and management and software process improvement. Nørbjerg received a PhD in information systems development from the University of Copenhagen. Contact him at jacobnor@cs.aau.dk.

to varying degrees, mature teams mastered reflection-in and -on action. Using standard and adapted reflective practices, these teams were able to discover better ways of working and also worked better together. They became highly self-organizing in nature.

From Learning Teams to Learning Organizations

In the software process context, a learning organization[8] is an environment most able to engage in reflective practice.[2] A learning organization recognizes the importance of individual and collective knowledge within the organization and the need to manage it as an asset.[2]. A learning organization values each member's professional development, realizing that effective people will make an effective product. Whereas reflective practice focuses on developing individual repertoire, a learning organization promotes the accumulation of individual repertoire.[9] In this sense, a learning organization provides a working environment that values learning as a daily activity.[8]

In essence, the introduction of reflective practice to a software development organization facilitates a learning system in which no particular software development method is as important as the ability to adapt and develop knowledge appropriate in the team. This leaves little doubt as to why agile methods have been so compelling in software engineering for the past decade: agile practically demands a learning organization. Moreover, the habits required to engage in reflective practice would more likely be fostered and nurtured in a learning organization.

REALM illustrates the relevance of reflective practice to software engineering for organizational learning as it helps accumulate individual and team repertoire. As individuals build their repertoire, and in turn establish a team repertoire, both the team and the individual are reinforced. Team reflections-on-action can reinforce individual reflection-in-action to cement learning in a manner that strengthens individuals, who in turn strengthen team acumen, and ultimately the organization itself.

Recommendations for the Reflective Agile Practitioner

A reflective agile practitioner is any software engineering professional—developer, tester, manager, coach—empowered by supportive management,[3] who engages in reflective practice as a means to enhance his or her personal and team productivity and fuel long-term process and team improvements. Based on our experiences with REALM, we can provide some recommendations for such reflective practitioners.

Harness Standard Agile Practices to Implicitly Capture Reflection-in-Action

It's easy to "go through the motions" when performing various agile practices. By being conscious of the learning opportunities present in everyday agile practices as suggested in REALM (such as release planning), an agile practitioner can gain significant knowledge about the product domain, usage context, required technologies (including learning new skills), and individual and team capabilities.

Make Effective Use of Retrospectives to Achieve Continuous Improvement

The retrospective is a standard agile practice explicitly designed to capture reflection-on-action. However, retrospectives are often easily abandoned or diminished to lip-service as teams get more used to the process and begin to focus exclusively on delivery. Reflective practitioners should hold on to the practice of retrospectives and use them effectively to reflect on their processes, thereby achieving continuous improvement.

Introduce Adapted Agile Practices to Explicitly Capture Reflection-on-Action

It's common for practitioners to devise new or adapted practices to overcome particular challenges and needs, such as learning spike, pair-in-need, and project wikis. A reflective practitioner should be open to inventing his or her own strategies or endeavor to be in sync with the latest in the practitioner community to help with learning and applying strategies devised by other practitioners.

Share Lessons within the Organization and Wider Software Community

Presenting experience reports at local and international conferences or workshops is a great way to distribute the knowledge and learning gained from personal experiences. Most agile conferences or workshops host practitioner tracks that welcome experience reports and presentations. Blogging is another effective practice to achieve similar outcomes with limited investments. Popular social networking communities also provide ample avenues to share with and learn from the wider practitioner community. Such efforts are typically well-received and even commended by peers and management and enable the practitioner to emerge as a thought-leader in his or her professional sphere. An even less demanding mechanism to share lessons with the wider community is to participate in research efforts that serve to capture the experiences of several practitioners, which in turn can help inform the international software community.

Although reflection as a means of learning is a highly espoused benefit and expected outcome of agile software methods, its practice doesn't explicitly provide guidance on how to create an environment that will support ongoing learning and reflection. Theoretically, however, the focus of agile principles on continuous learning presents a unique opportunity to benefit from the application of the reflective practice paradigm developed by Schön as a means to achieve process maintenance (and evolution).

REALM uses reflective practice to develop micro habits for uncovering tacit knowledge among a software development team. Whether an individual must adopt a new beneficial habit or cease an existing detrimental one, habit-forming is a behavior modification concern that's "easier said than done." As it has been suggested that practitioners capable of success with agile methods should generally be high-functioning, the agile tenet for self-organization is highly correlated with the habits required for successful reflective practice.

Using REALM, software teams can consider ways to implement reflective practice to achieve and sustain a much-emphasized element of agile methods: the ability to adapt and evolve methods for a continuously improving software process at the individual, team, and organizational levels.

References

1. A. Cockburn and J. Highsmith, "Agile Software Development, the People Factor," *Computer*, vol. 34, no. 11, 2001, pp. 131–133.
2. O. Hazzan and J. Tomayko, "The Reflective Practitioner Perspective in eXtreme Programming," *Extreme Programming and Agile Methods: XP/Agile Universe 2003*, F. Maurer and D. Wells, eds., Springer Berlin Heidelberg, 2003, pp. 51–61.
3. R. Hoda, J. Babb, and, J. Nørbjerg, "Toward Learning Teams," *IEEE Software*, vol. 30, no. 4, 2013, pp. 95–98.
4. N.B. Moe, T. Dingsøyr, and T. Dybå, "A Teamwork Model for Understanding an Agile Team: A Case Study of a Scrum Project," *Information and Software Technology*, vol. 52, 2010, pp. 480–491.
5. S. Nerur and V. Balijepally, "Theoretical Reflections on Agile Development Methodologies: The Traditional Goal of Optimization and Control is Making Way for Learning and Innovation," *Comm. ACM*, vol. 50, no. 3, 2007, pp. 79–83.
6. J. Babb, R. Hoda, and J. Nørbjerg, "Barriers to Learning in Agile Software Development Projects," *Agile Processes in Software Engineering and Extreme Programming*, H. Baumeister and B. Weber, eds., Springer, 2013, pp. 1–15.
7. G. Coleman and R. O'Connor, "Investigating Software Process in Practice: A Grounded Theory Perspective," *J. Systems and Software*, vol. 81, no. 5, 2008, pp. 772–784.
8. C. Argyris and D. Schön, *Organizational Learning: A Theory of Action Perspective*, Addison Wesley, 1978.
9. D. Schön, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, 1983.
10. D. Schön, *Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions*, Jossey-Bass, 1987.
11. J.A. Highsmith, *Adaptive Software Development: A Collaborative Approach*, Dorset House, 2000.
12. J. McAvoy and T. Butler, "A Failure to Learn by Software Developers: Inhibiting the Adoption of an Agile Software Development Methodology," *J. Information Technology Case and Application Research*, vol. 11, no. 1, 2009, pp. 23–45.