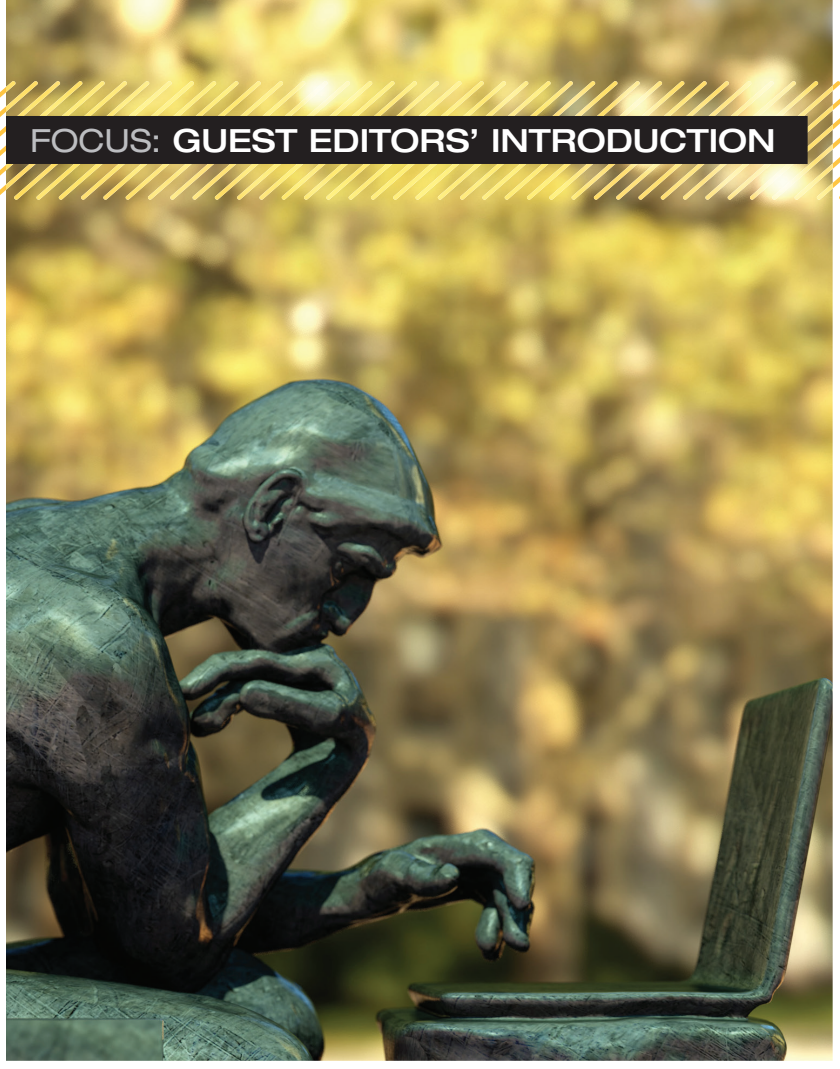


The Reflective Software Engineer:

Reflective Practice

Tore Dybå, SINTEF

Neil Maiden, City University London

Robert Glass, Computing Trends

"Life can only be understood backwards; but it must be lived forwards." —Søren Kierkegaard

THE CAPACITY TO reflect on past practice is important for continuous learning in software development. Reflection often takes place in cycles of experience followed by conscious application of learning from that experience, during which a software developer might explore comparisons, ponder alternatives, take diverse perspectives, and draw inferences, especially in new and/or complex situations. Such reflective practice has been shown in different disciplines to be an effective developmental practice for organizations, for teams, and for individuals.

For example, a reflective agile developer who, after collaborating with client end users to implement a new software feature, might choose to reflect on the effectiveness of the process of co-designing wireframes, the choice of selected software libraries with which to prototype the feature, and the value of the daily scrum to manage project progress. He or she might consider alternatives and then infer what could be done more effectively next time. Information that could be provided to the developer to enable this reflection might include the number of user changes to the wireframe design, time taken to implement the software, and concrete outcomes of daily scrums. The developer uses this information to encourage sensemaking, critiquing, and the identification of new forms of development work.

However, such reflection in practice happens all too infrequently in software development. Not only do software developers lack the tools to capture, analyze, and present information upon which to reflect, but most software projects don't actively support reflection, or budget or schedule for it. Tom DeMarco,[1] for example, challenged downsizing,

0740-7459/14/$31.00 © 2014 IEEE

restructuring, and cost cutting in software management in the name of efficiency and global competition. The resulting costs in our human capital—stress, pressure, and overcommitment—are self-defeating and often remove the incentives and resources needed for effective reflective learning.

Indeed, our agile developer's desirable reflective practice offers an alternative approach to learning a body of knowledge that has to be acquired by novices, which can then be applied to solve predefined problems in practice. (We refer here to a generic body of knowledge: assumed knowledge about a business, about how technology works, about good work practices, all of which need to be reflected on and learned from.) A reflective practitioner more often questions how to think and act, either after having acted (reflection on action) or in the midst of acting (reflection in action). The latter makes it possible to alter your current course of action by framing the problem in a new way or by improvising new ways of solving the problem at hand.

Therefore, the social and cultural context in which reflection takes place has a powerful influence over what kinds of reflection it is possible to foster and the ways in which this might be done.

Reflective Practice

The concept of reflective practice centers on the idea of lifelong learning. Fundamental to such reflective practice is the integration of theory and practice: the cyclic pattern of experience and the conscious application of the learning outcomes of that experience.

For the past 30 years, the literature has grown to focus on experiential learning and the development and application of reflective practice. An important insight from this literature is that the most powerful learning comes from direct experience. Among the most influential theorists who have explored the central role that such experience plays in the learning process are John Dewey, Kurt Lewin, and Jean Piaget. They inspired the work of later theorists such as Gregory Bateson, David Kolb, David Boud, and Donald Schön.

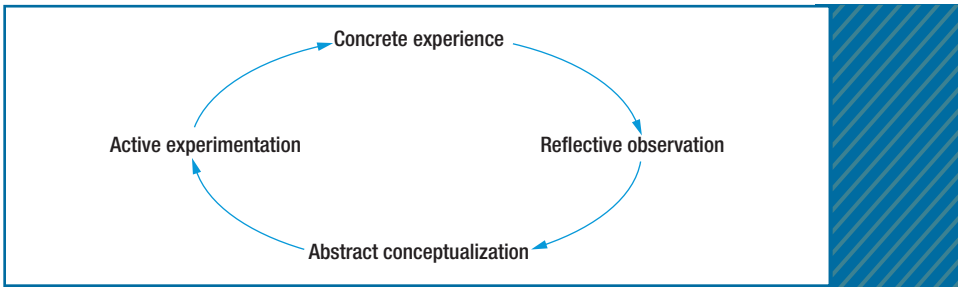


FIGURE 1. Experiential learning cycle.

Kolb's model of experiential learning summarizes the ideas of these theorists and explains the process of reflection at the individual level as a cycle consisting of four stages (see Figure 1).[2] The cycle begins with actual or *concrete experience* that deals with immediate human situations in a personal way. Next is the individual's ability to reflect on and observe experiences from many perspectives and develop these observations into collections of related ideas in a process that Kolb called *reflective observation*. The third stage in the learning cycle is making sense of our experiences, which involves constructing models to define and explain or predict what we observe; Kolb called this *abstract conceptualization*. In the final stage, we seek to test our ideas in new situations through *active experimentation* (trying something new). The outcome of such an experiment becomes concrete experience and thus a spiraling cycle of experiential learning becomes apparent.

Direct experience is also connected to processes of sensemaking: the combination of a past moment, a connection, and a present moment of experience is what creates a meaningful definition of the present situation. Thus, experiential learning has in common with sensemaking that it requires three things: two elements and a relation.

There has been a similar development and focus on learning and reflection at the group and organizational level. Victor Basili and Gianluigi Caldiera, for example, provided a quality improvement paradigm for the software industry that focuses on reused learning and experience by establishing so-called *experience factories*.[3] These experience factories support experience reuse and collective learning by developing, updating, and providing, on request, clusters of competencies to be used by the organization's projects.

The fundamental role of managers and leaders is to shape and create organizational contexts that are supportive of and conducive to reflection and organizational learning. To the extent that the software organization remains locked into the old context, no significant change or improvement is possible. This means that managers have to become

skilled in enabling the learning processes and in defining new and appropriate contexts for reflection.

The manager can create such contexts by enabling a belief-driven process that generates new understandings of a situation or by engaging in new actions through an action-driven process. The conventional way of thinking about software process improvement puts these in a sequential order—first understanding, then action. For example, in the early Japanese software factories, a strong emphasis was put on gathering data on existing processes before changing or improving them. From a learning perspective, however, actions and understandings often need to be reversed. Therefore, the generation of new understandings and new actions, in whatever order they evolve, are fundamental in creating the contexts that enable reflective practice.

> The concept of reflective practice centers on the idea of lifelong learning.

Occasions of Reflection

Whether software development takes place within projects or as part of a continuous product development and improvement process, it's convenient to discuss occasions of reflection in terms of three stages[4,5]: activities and experiences before the event, during the event, and after the event.

Reflection before Events

The emphasis here is on what we can do to make the most of future events. Although we can never predict fully what will occur, it's useful to reflect on what we bring to the event, what we want out of it, and what we need to be mindful of that could distract us from our intentions. It's also useful to reflect on what we need to know to make the event a productive one and on what ideas other people might have about what will happen. Furthermore, it might be useful to reflect on what to do if the assumptions about the event are wrong and what to fall back on to cope effectively.

A simple way of reflecting at the group level is to arrange a learning meeting or a reflection workshop when initiating a new project—participants in earlier projects are invited to share their experience, insight, and knowledge with the members of the new project.[5]

The purpose of such a reflection meeting before a project starts is to learn from others in order to contribute to the project, not to provide criticism of the project planning. The idea isn't to summon global experts or top management. It's all about finding people who "have done this before" and who come to share their insights with the new project team. Both parties learn—the external peers acquire a broader knowledge base, while the new project team will be able to exploit earlier experiences.

Reflection during Events

The core of the reflective practice is learning while doing. It is such reflection-in-action that lies at the heart of Schön's view of experiential learning.[6] Continuously changing environments and conditions often asks for adjustments, for example, to established processes and standards, and learning along the way implies the need for a large amount of creativity[7] and improvisation.[8]

At the individual level, it's our engagement with an event through noticing, intervening, and reflecting in the midst of action that constitutes a learning experience.[9] We learn by becoming aware of what's happening in and around us and by taking actions to change the situation in which we find ourselves. Schön's concept of reflection in action refers to such situations when we're able to consciously evaluate and make changes on the spot during an event.

At the group level, we can often schedule time for reflection in the form of short learning meetings immediately after an activity or an occurrence. Such meetings make it possible for us to learn from positive and negative experiences while we still are able to do something about them.[5] The point is that the continuous learning meetings are short and focused. Scrum's 15-minute daily meeting[10] in Scrum or the more comprehensive agile retrospectives after iterations or releases[11] are examples of such occasions for reflection in action.

Asking simple questions—What was supposed to happen? What actually happened? Why were there differences? What can we learn from this?—is often useful for learning meetings in the midst of action. These questions can trigger adjustments and improvised changes as the basis for continuous improvement on all levels, spanning from a simple activity or process to iterations and even to the project or company as a whole.

Reflection after Events

Important occasions of reflection occur once the immediate pressure of acting in real time has passed. Some learning inevitably takes time and requires the ability to view particular events in a wider perspective.

The basis of all learning at the individual level is the lived experience of the learner. Often, too little emphasis is placed on what has happened and how it was experienced at the time. Mentally revisiting positive and negative incidents can be an important step in the process of reevaluating the experience. This process of reevaluation includes relating new information to that which is already known, seeking relationships between new and old ideas, determining the authenticity for ourselves of the ideas and feelings that have resulted, and making the resulting knowledge our own—that is, a part of our normal ways of operating.[4]

At the group level, we reflect on our collective experiences when the project, or a larger part of, is finished in order to get a better understanding of what happened. A simple way of collectively reflecting on such experience after events is to arrange a learning meeting. Project postmortems and sprint, release, and project retrospectives are examples of such moments of reflection after events.[12]

Arranging such learning meetings is a quick and effective way of collecting knowledge and experience before the project is dissolved and an important part of any improvement strategy. Retrospectives can also be used for immediate transfer of knowledge to a new project that's about to start. This way, retrospectives also make sure that future teams can use the knowledge and experience that a recently finished project has acquired. These experiences can take the form of new or updated models and processes, or as other forms of knowledge made available for future projects.

A project retrospective or postmortem is more thorough than the learning meetings in the midst of action. They typically last from a couple of hours to a whole day. The project retrospective is also different by focusing on collecting experience and knowledge for future projects—not only on learning in action, inside a running project. The main motivation is to reflect on what happened in the project to improve future practice, for the individuals who participated in the project as well as for the organization as a whole.

In This Issue

We received 26 submissions to this special issue from all over the world. Based on the feedback of our expert reviewers, we eventually selected four articles that we thought could best address reflective software practice with its many facets. As well as recognizing a belief in the importance of reflective software practices, all the articles also agree on something else—that reflection in the hectic life of software practice is difficult to achieve.

> The core of the reflective practice is learning while doing.

In the face of that fact, each article describes an approach that the authors have evaluated in some practical way. Two of them describe an analysis of the results of using the approach they advocate in an at least somewhat practical setting. Another describes the use of its recommended technique via case studies of actual usage in industrial settings. And the fourth describes an experiment in the use of what they advocate. In other words, these articles aren't just about the theory of reflective practice; they're about its implementation.

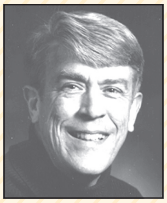
The first article, "Reflecting on Evidence-Based Timelines" by Elizabeth Bjarnason, Anne Hess, Richard Berntsson Svensson, Björn Regnell, and Joerg Doerr, proposes the use of a project-based timeline to focus reflection on certain aspects of a project. This, of course, requires some preparation for the reflection itself because a project timeline must be prepared, and key events along the timeline recorded.

The second article, "Supporting Reflective Practice in Software Engineering Education Through a Studio-Based Approach," by Christopher N. Bull and Jon Whittle, takes a rather different view of how to achieve reflection. It suggests that one reason why reflection in practice is lacking is that education in reflection, a necessary prelude, is also lacking, so it proposes a way to "teach reflective techniques from the start." Toward that end, it proposes what the authors call a "studio-based approach," and describes in detail how such an approach could be implemented in an educational setting.

ABOUT THE AUTHORS

TORE DYBÅ is chief scientist at SINTEF, Norway. His research interests include agile software development, evidence-based software engineering, and management of large-scale software projects. Dybå received a Dr. Ing. in computer and information science from the Norwegian University of Science and Technology. Contact him at tored@sintef.no.

NEIL MAIDEN is a professor of systems engineering at City University London. His research interests include requirements engineering, computer-based creativity support, and technologies to support people living with dementia. Maiden received a PhD in computer science from City University London. Contact him at N.A.M.Maiden@city.ac.uk.



ROBERT GLASS is president of Computing Trends, Australia. His research interests include software engineering in practice, software quality, and software maintenance. Glass received an honorary PhD from Linkoping University, Sweden. Contact him at rlglass@acm.org.

The third article, "Embedding Reflection and Learning into Agile Software Development," by Jeffry Babb, Rashina Hoda, and Jacob Nørbjerg, notes that the agile philosophy calls for reflection as part of its software construction process, but, like the other articles, the authors observe that even on agile projects, this seldom occurs. The article presents a "reflective agile learning model," showing where and how to integrate reflective practices and learning into agile software development. The article relies on evidence from studies of agile practices in a small software company in the US, and on a Grounded Theory study of 23 different companies in New Zealand and India. Based on those findings, it recommends approaches that can be used in agile practice to achieve the reflection that the agile practices recommend.

The last article, "Coderetreats: Reflective Practice and the Game of Life" by David Parsons, Anuradha Mathrani, Teo Susnjak, and Arno Leist, describes an experiment in an academic setting to explore a very specific way of achieving reflection. It borrows ideas from "The Game of Life," in which participants focus intensely on a narrowly described part of a program but from the point of view of reflection on the fundamentals of simple modular design. In the experiment, the authors gathered a range of data to assess the relevance and effectiveness of what they proposed.

In *The Reflective Practitioner*, Donald Schön emphasized a need for consistent systems for ongoing learning at both the individual and the organizational level.[6] Correspondingly, our aim with this special issue is to present examples of methods, tools, and experiences that support such individual and group reflection. We hope this issue provides you with some ideas for how you can create a context for the sharing of knowledge and experience at a personal level as well as at the team and project levels in your organizations, and that it stimulates rich discussions on how to learn from and improve practice.

References

1. T. DeMarco, *Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency*, First Broadway, 2001.
2. D.A. Kolb, *Experiential Learning: Experience as the Source of Learning and Development*, Prentice Hall, 1984.
3. V.R. Basili and G. Caldiera, "Improve Software Quality by Reusing Knowledge and Experience," *Sloan Management Rev.*, vol. 37, no. 1, 1995, pp. 55–64.
4. D. Boud, "Using Journal Writing to Enhance Reflective Practice," *New Directions for Adult and Continuing Education*, vol. 90, no. 3, 2001, pp. 9–17.
5. T. Dybå, T. Dingsøyr, and N.B. Moe, *Process Improvement in Practice: A Handbook for IT Companies*, Kluwer Academic, 2004.
6. D.A. Schön, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, 1983.
7. R.L. Glass, *Software Creativity 2.0*, developer.* Books, 2006.
8. T. Dybå, "Improvisation in Small Software Organizations," *IEEE Software*, vol. 17, no. 5, 2000, pp. 82–87.
9. D. Boud, R. Keogh, and D. Walker, eds., *Reflection: Turning Experience into Learning*, Kogan Page, 1985.
10. K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, 2009.
11. E. Derby and D. Larsen, *Agile Retrospectives: Making Good Teams Great*, Pragmatic Bookshelf, 2006.
12. T. Dingsøyr, "Postmortem Reviews: Purpose and Approaches in Software Engineering," *Information and Software Technology*, vol. 47, 2005, pp. 293–303.