Editor: Donald J. Reifer ■ Reifer Consultants ■ d.reifer@ieee.org

Scaling Agile Methods

Donald J. Reifer, Frank Maurer, and Hakan Erdogmus

Using agile methods to develop large systems presents a thorny set of issues. If large teams are to produce lots of software functionality quickly, the agile methods involved must scale to meet the task. After all, a small team could create the software if the functionality to be delivered was small and, conversely, could be delivered given we had the time. Scaling agile teams thus becomes an issue if the only option for meeting a system delivery deadline is to have many developers working concurrently.

At the First Invited Canadian Workshop on Scaling Agile Methods held 20–21 February in Banff, Alberta, 35 concerned professionals met to discuss these issues. The industrial delegates, who had been putting agile methods to work (using lightweight methods like Crystal, Extreme Programming, Dynamic Systems Development Method, Feature-Driven Development, Scrum, and a scaled-down version of the Rational Unified Process) addressed a wide range of issues, notably how to

- Scale agile methods to very large projects with barely sufficient up-front planning and architectural work
- Deploy a federation of coordinated teams (each internally operating as an agile team) in scaling up agile ideas
- Use agile methods in teams larger than a typical XP team
- Characterize the agile continuum through different project caricatures, ranging from typical collocated XP projects to large, multiteam, multiyear ones

The academic delegates shared their experiences and ideas on how to

- Use agile practices, such as test-driven development and pair programming, as pedagogical tools in software engineering curricula
- Investigate agile practices' effectiveness
- Reconcile agile methods with architectural paradigms
- Deploy agile methods in research projects

Significant issues

During the first day, delegates reached consensus on the following top seven issues associated with scaling agile methods.

Non-pure agile methods: reconciling agile methods with plan-oriented practices

Most delegates agreed that agile methods fit small projects where problems of scale are minor. However, difficulty arises in scaling these methods to fit large projects where teams of teams work together. Most delegates also agreed that deploying agile methods in organizations involves using a mixed metaphor: agile methods must operate in a world where both agile and traditional methods can be used together. The issue then becomes how to scale agile methods without sacrificing the underlying principles of the Agile Manifesto (see www.agilemanifesto.org for the history of the agile movement, its goals, and its governing philosophies).

Generating guidelines for non-sweet-spot agile projects

A sweet-spot agile project typically involves a small, self-organizing, collocated team of

0740-7459/03/$17.00 © 2003 IEEE

fewer than 20 developers and one or more on-site customers. The team usually works together on a variable-scope application with unstable or emergent requirements, relying mainly on an oral culture based on high-bandwidth, face-to-face communication. What constitutes a large project and where the sweet spots are for deploying agile methods was probably the workshop's most hotly debated topic. Some argued that agile methods shouldn't be used outside their original domain of fit. Others suggested that developers would scale agile methods to larger projects whether delegates liked it or not. All agreed that guidelines for scaling them are needed.

Augmenting agile practices to fit large projects

The additional practices that would be necessary to scale agile methods came up repeatedly, especially when discussing non-sweet-spot agile projects. For example, some suggested that the concept of a daily team meeting could extend to include a daily intrateam project meeting. Others discussed the issues associated with scaling planning meetings and requirements engineering concepts. Although workshop attendees exchanged ideas, choosing proper practices and strategies is still an open issue.

Addressing integration issues in an agile project

Most delegates agreed that reported "green field" projects are idealistic. Most applications exploit existing architectures and use extensive amounts of legacy software, commercial off-the-shelf packages, and components. How these reusable components constrain the use of agile methods as elements of scale drew considerable discussion. To accommodate legacy and COTS software, some suggested that test-drivers be written to reveal all relevant external behavior. Coordinating the use of standard frameworks and components across teams of teams when agile methods and collective ownership principles are used was another hot issue.

Scaling agile in an enterprise across applications

Coupled with the issues of legacy were those associated with product families and lines. Participants agreed that applications do not exist in isolation in an enterprise. At the workshop, delegates pondered how to handle the communications barriers across the enterprise when using agile methods to build systems. Most wanted to avoid stovepipes when scaling a project, as applications are produced using short iterative cycles. Some thought that systematically generalizing and aggregating user stories across similar applications to factored-out framework stories could help in managing requirements.

Handling dispersed development in an agile project

Team size and how to grow it so that the team could operate in a dispersed (and often geographically separated) environment also drew discussion. Although some had experience growing teams of teams across geographic distances, the participants formed little consensus on how to overcome the communications barriers. Some delegates suggested a team hierarchy and a phased rollout model—for example, architecture team first, then feature and integration teams. Others considered this a serious threat to agility. Synchronizing the beats of the individual teams and the possibility that the slowest or the weakest team would determine the project's overall pace or fate were major risks.

Most delegates agreed that the agile approach of having customers write acceptance tests as new features are implemented pays dividends.

Integrating testing as systems get bigger

Most delegates agreed that the agile approach of having customers write acceptance tests as new features are implemented pays dividends. However, in a large project, securing sufficient customer involvement is unrealistic. Someone suggested that QA teams of business analysts could take over the task of writing customer acceptance tests. The issue becomes how to mechanize the agile approach without unduly burdening the customer community.

Early adopter experiences

Next, we discussed ways to resolve these issues. Although we didn't quite answer the questions, we did capture our early adopters' experiences relative to these issues, primarily in the form of lessons learned. We present the top seven lessons early adopters learn.

It's going to happen

While several participants, especially Martin Fowler in his keynote address, argued that scaling XP and agile projects is probably the last thing you would want to do, most delegates agreed it would happen anyhow. Because of the communal push for scaling, developers will do it despite strong warnings otherwise.

The shorter, the better

Everyone agreed that spending time getting releases out according to preestablished, short, time-boxed schedules, with whatever functionality they could, is advantageous. Although no consensus emerged on the optimum duration, most agreed that weeks rather than months was the proper range for iterations and releases. Participants agreed on the value of continuous integrations and automated build-and-test processes. Instead of talking about what the product functionality would be, developers could show customers products that are "potentially shippable." This approach removes the subjectivity from the discussions.

Improving communications on large projects

For large projects, participants advocated the benefits of using daily project meetings where team leaders get together to work through issues; identify desired functionality; and coordinate release contents, iterations, and progress.

Up-front investment will help

In a departure from current thinking, delegates working on large projects agreed that some architectural development was needed before pushing ahead with iterations. This could be done quickly using an architecture team. Team members then could move on to seed a large-scale agile project's subteams. While the architecture will continue to evolve over the project's life cycle, the tendency will be to stabilize it and discourage any significant changes. A stable architecture will provide teams with the context for decision making. However, this approach poses a threat to agility because it might tip the scale in favor of up-front planning rather than letting the architecture emerge naturally.

A federation strategy can help

Several delegates suggested that a federation of onsite customers could give developers the daily feedback they need. Knowing who speaks for the customer on what issue is the key to success in such an arrangement. Establishing known points of contact for specific items, therefore, is essential in large projects, delegates felt. Multiple customer representatives might not speak with a single voice anymore, making binding decisions more difficult to reach.

Packaging components can reduce scaling problems

Some delegates recommended that packaging components with wrappers that communicated essentials about their features to those unfamiliar with them would minimize integration problems on large projects. Writing test drivers to reveal how components behave in a specific context is another strategy. Because the focus rests on releases, this strategy would ease the communications problems and permit using components in their proper context. While wrapping cloned components for use in different contexts will reduce commit clashes and alleviate integration problems in a collectively owned code base, it will increase redundancy and thus reduce maintainability.

Set lower expectations

Larger teams tend to be less tolerant to change. Adopters of agile methods should therefore be extremely conservative when setting expectations for change, especially when using these new methods on large projects.

The workshop built bridges that let delegates from academia, industry, and government discuss their experiences and concerns. Momentum to keep the dialogue flowing was probably the workshop's most important output. Toward these ends, plans for a second workshop on scaling agile methods for next year are moving forward. Planners are augmenting the Canadian Agile Network Web site (http://can.cpsc.ucalgaray.ca) with additional resources for those interested in putting agile methods into practice in industry, academia, or government. The organizers thank the delegates for a job well done. We learned a great deal and are encouraged by the positive results of agile method use.

Donald J. Reifer is a visiting associate with the Center for Software Engineering at the University of Southern California and president of Reifer Consultants. Contact him at dreifer@earthlink.net.

Frank Maurer is an associate professor in the Computer Science Department at the University of Calgary. Contact him at maurer@cpsc.ucalgary.ca.

Hakan Erdogmus is a research officer at the National Research Council of Canada. Contact him at hakan.erdogmush@nrc-cnrc.gc.ca.