Top 5 lessons learned about scaling Agile from a leading insurance provider

Richard Knaster

January 23, 2012

Agile development is a collaborative, incremental and iterative approach to software development that can produce high-quality software on schedule and cost-effectively. Agile practices were initially designed for small collocated teams, but you can adapt them to fit a more complex environment. IBM has experience with this not only internally, but with other large enterprise clients. Among the most valuable lessons we've learned about implementing agile in the enterprise were gained while helping a large insurance company, we'll call "Insurance Co," with their agile adoption and their implementation of IBM Rational Team Concert.

Agile adoption in the enterprise

Changing any process in an enterprise is never an easy task, and moving from traditional to agile development is no exception. Adopting agile requires a great deal of effort that can take several years because you must change organizational culture, beliefs, governance, HR policies, titles, funding and more. Neither a unilateral nor a haphazard implementation will work in the long run. It takes a disciplined, incremental approach. This is one of the lessons IBM learned with one client, a large insurance company I'll call "InsuranceCo."

InsuranceCo's software delivery challenges

The InsuranceCo journey to agile started out as a Rational Team Concert deployment. They wanted to improve project management with more transparency, better collaboration and a more uniform process for different types of development teams working very closely together. After my team and I began this project, it soon became evident that there were serious impediments to these goals.

For example, they had hundreds of developers on numerous distributed teams, more than 20 different types of work items, almost no reporting and inconsistent change management. In addition, most teams were not practicing continuous integration, they could not see how projects were progressing to reduce risk early and their time to market needed significant improvement.

We soon realized that this company would not truly benefit from their tools investment without changing their processes. We suggested an agile transformation and learned five valuable lessons about scaling agile:
1. A team-by-team, incremental approach is best.
2. Measurement and management tools can help you get and sustain executive buy-in and improve the development process.
3. Mentoring and coaching should be provided for the process first and for the tools second.
4. Integrated tools help demonstrate value.
5. Retrospectives are essential for continuous improvement.

A team-by-team, incremental approach

There's no such thing as a "one size fits all" approach to agile adoption. In InsuranceCo's case, they had all kinds of development projects to support, including Java, IBM WebSphere and mainframe, along with specialized resources such as database administrators and data architects. It simply would not work to implement agile methodologies identically for all these teams or attempt to transform all of them at once.

Instead, we analyzed how each team worked and determined what practices they could adopt and what modifications to the basic agile processes were needed. For example, the mainframe developers had to deliver all their code together; they could not incrementally add code to their test environment. So we created a story for the development and a story for the test, which is different from pure and simple agile methods. Normally, one story covers everything. However, because of their constraints, we realized things had to be done differently to work for their teams. Eventually, we could figure out how to change their environment and infrastructure so that they could become even more agile.

Tools for both executives and developers

Critical to the adoption of agile in a large enterprise is executive buy-in. This was particularly challenging at InsuranceCo because their CIO was initially opposed to agile methods. So, we met with the executive team to explain what would be different and how it would work. We used real-world examples to show them that the outcome would be worth the investment. To sustain their buy-in after we began the project, we presented them with a set of metrics and measurements. We provided them with dashboards and reports that showed what the teams were working on, whether they were making progress, and how quickly teams were completing their releases. Providing transparency into the work of the development team helped keep the executives on board.

For individual agile teams, we implemented tools that provided a lightweight approach to task management. Predefined process templates introduced Rational Team Concert work item capabilities. These templates helped the InsuranceCo development teams use and share agile best practices, provided administrators with support for different processes, and enabled updates to be shared so that teams could improve their processes independently.

Process training first, tools second

During the agile transformation process, we discovered that InsuranceCo had identified subject matter experts (SMEs) to help educate the teams, but many of these SMEs were not really experienced with agile practices. They weren't really able to facilitate agile practices because you cannot give someone a little bit of training and expect them to coach other teams.
So, we provided additional mentoring and coaching. We held 30-minute daily meetings for process and tool questions and answers, where internal coaches were mentored and gained experience. We also implemented a transition community with internal discussion forums, wikis, FAQs and training documents. As we rolled out agile practices to each team, we held mentoring sessions for them, and we adjusted processes and tools for each team where needed. We also identified repeatable patterns that each team could use as-is or adapt to meet their unique needs.

Integration helps show incremental value

Just as a unilateral approach to agile development in a large enterprise is not recommended, neither is a unilateral, "rip-and-replace" approach to tools. We were able to use the fact that Rational software integrates with other systems from other companies to show incremental value. InsuranceCo was using a quality tool from another company and they were able to keep on using it after we implemented our software because of its integration capabilities. Over time, they decided that they could achieve better integration if they used all IBM software. Initially, however, they were able to see the value of one of IBM’s tools before they decided to implement anything else.

Continuous improvement: A retrospective

Continuous improvement is an integral part of agile development. To ensure that the new process and tools were working for the teams and to understand what problems they might be encountering, we held a retrospective with each team two weeks after we trained them on the process and tools. This helped us identify areas where additional coaching was needed. One example was writing stories. We were asked several times to help teams with their stories and we saw that there was a pattern there, so we added more coaching about stories to our overall coaching and mentoring processes.

Conclusion

Implementing agile development in a large enterprise takes a disciplined, incremental approach. Trying to implement it all at once when you have different development platforms, different kinds of resources and globally distributed teams is a recipe for failure. However, if you follow a customized approach that tailors agile practices to the needs of each team, introduce tools that support development's goals but also sustain executive buy-in, and invest in training and mentoring to help instill good agile habits, even the largest of companies can reap the benefits of agility at enterprise scale.

© Copyright IBM Corporation 2012
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)