

INT Professional Services:

Using Agile Unified Process to Reduce Cost and Risk

What are Agile Methods?

Early in 2001, in Snowbird, Utah, a group of software developers frustrated with existing heavyweight software methodologies met to find a better way. They discovered that many of the tools and techniques that they had made them successful were also being used by others. They named these “agile” techniques because of their ability to allow projects to quickly adapt to change.

Since then, agile methods have been adopted on projects by companies and organizations such as Symantec, Chrysler, The Chicago Mercantile Exchange, and Microsoft [1,2,3]. The growing popularity of agile methods is driven by the dramatic improvements in both productivity and quality that companies see after adopting them.

Why Use Agile Methods?

Out of necessity, software development methods used for fixed cost projects strive for stability in an attempt to achieve predictability. They usually prescribe a four phase process in which requirements are gathered and written into a specification; a system design is generated; the system is implemented according to the design; and the final product is tested and deployed. The problem with these methods is that change is extremely costly. Any mistakes made in the requirements or design phases will result in unstable or useless software. Mistakes made in the construction phase are often repeated over and over, because there is no automated failsafe to prevent them. The emphasis on stability also makes it extremely difficult for customers to adapt to changing market conditions. If a competitor releases a new product with an unexpected feature, or the needs of the users change, the only choice is to scrap the requirements and design. Most of the work done up to that point must be heavily modified and tested or simply thrown away.

Agile methods are so named because they *embrace change*. They accept that most software projects are ever changing, and prescribe techniques to mitigate the risks inherent in software development. By breaking the projects into small iterations in which requirements, design, development, and testing occur continuously, agile methods are able to work well in situations where other methods fail. Some indicators of these situations are:

- The project is reasonably small (2-12 people).
- Customers and developers can communicate with each other.
- The requirements may change during the project.
- The problem scope is not completely defined.
- The problem is complex.

Agile methods work by combining complementary and interdependent techniques to greatly reduce the amount of overhead necessary in a fixed cost project. The resulting product more accurately represents the customer's needs, rather than simply what they were able to specify at the beginning of the project.

Agile Unified Process Overview

Although other methods like XP and Evo are possible, the agile method that INT usually employs is a highly specialized version of IBM's Unified Process. While a small amount of customization can be performed on a per project basis, most projects follow the same four phase procedure for each release of the product.

Inception - Customers meet with INT developers to discuss the scope of the project

1. Initial requirements are gathered either through a face-to-face meeting or by specification. It is not necessary to define all the requirements at this time.
2. Requirements are decomposed into individual features (usually described with a use case).
3. Features are entered into INT's task management system.

Elaboration – INT assigns estimates and a development plan is created.

1. INT examines each feature and generates a rough estimate of it's time to implement.
2. Customers use the estimate to assign a priority to the features. It is not necessary to assign a priority to every feature right away.
3. Customers collaborate with INT developers to create a development plan. This plan describes what form the deliverables will take (source code, binaries, tests, documentation, etc...) and how they will be delivered.

With only one exception (addressed below) requirements can be added, modified or removed at any time, even in later phases. INT will then adjust it's estimates and customers can re-prioritize as they see fit.

Construction – Work begins and continues to completion.

Development progresses in small increments called *iterations*. An iteration consists a series of steps performed over a short (usually two week) timetable. The steps include:

1. Picking the features to implement in this iteration.
2. Resolving any outstanding bugs (½ day).
3. Doing a rough design of the components for this iteration (½ day).
4. Implementing the features for this iteration. Includes:
 - Acceptance tests
 - Refactoring
 - Unit tests
 - Documentation (if necessary)
5. Delivering the product to allow for customer feedback. Customers are free to change the requirements or priorities based on information gained from this iteration.

This process can continue for as long as the customer wants. By adding more features, the product can be extended and improved. By removing features, earlier deadlines can be met.

Transition – Product is deployed or moves into support phase.

At this point the product can be deployed for use. If future releases are required, the customer can begin the process again with more requirements. If the product requires no future releases, it can be moved into a maintenance state with longer iterations performed either by INT or the customer.

Other points to note about this process...

- Acceptance tests are delivered with the product, and customers are encouraged to run them upon delivery to ensure that environment or deployment issues have not introduced bugs. Customers are also encouraged to change acceptance tests and send them back, should they discover a bug or missing feature.
- The only time customers are not able to change the scope or priority of features is when they are currently being implemented. If a feature included in the current iteration must be changed, the code will be branched from the last point of delivery and the iteration will be restarted. In practice, this occurs very infrequently.

Agile Development Tools

Agile developers depend a great deal on tools and task automation. These tools need not be expensive or complex, but they do need to integrate well with the day-to-day tasks of an agile development team. Here are some of the classes of tools used, with some specific suggestions based on INT's experience.

Developer tools:

- Build tool (Ant, NAnt)
- Development Environment (Eclipse, JBuilder, VisualStudio, IntelliJ)
- Documentation (NDoc, JavaDoc)
- Testing Framework (JUnit, NUnit)
- GUI Testing Framework (Carabiner, JFCUnit)
- Static Analysis (Findbugs, JDepend)
- Code Coverage (Emma)

Project Management Tools:

- Task Management (XPlanner)
- Bug Tracking (Bugzilla, Track+)
- Source Control (CVS, SourceSafe, Subversion)
- Project Documentation (JSPWiki)
- Continuous Integration Build System (Anthill, Cruise Control)

A common practice is to have a single project server that manages the task management, source control, bug tracking, project documentation, and build system. This server must be accessible by all the developers on the team, and if possible, by the customers (via Virtual Private Network).

Benefits and Drawbacks of Agile Methods

Benefits:

- Automated testing dramatically lowers defect rates and improves software design for better maintainability.
- Frequent releases increase feedback, lower total risk, and mitigate risk of failure by front-loading potential problems.
- Frequent releases also allow customers to stop the project at any time, for whatever reason, and still have a working, stable, and clean (albeit incomplete) version of the system.

- Focus on working, tested software lowers costs and helps eliminate 'bloatware' caused by useless or misunderstood features. The resulting design is simple, straightforward and easy to maintain.
- Iterative development cycle provides high project visibility and control, allowing customers to react quickly to changing requirements or market conditions.
- Constant feedback greatly improves the usefulness of the system by ensuring that features are implemented the way the *customer* wants them, not just how the developer chose to implement them.

Drawbacks:

- Agile methods require highly skilled and disciplined developers.
- Agile methods require frequent communication between developers and customers. For customers that are unable to provide this feedback, agile methods may be inappropriate.
- Agile methods require knowledge of, and access to, highly specialized development tools.

Related Articles:

[The New Methodology, Martin Fowler](#)
[XP: The Competitive Edge, Robert C. Martin](#)

References:

- [1] <http://www.objectmentor.com/processImprovement/xpCaseStudies>
- [2] <http://www.eweek.com/article2/0,1895,1885883,00.asp>
- [3] http://www.cio.com/archive/081504/challenge_competition.html

About INT:

INT provides high performance software components and development services for applications requiring visualization and analysis of complex data. INT products span a variety of environments, including C++, Java, .NET, and C (X/Motif). INT services span the full cycle of a software project, from solution design, to development, to application support.

By utilizing INT components and services, our customers stay focused on the strategic activities that make them successful.



High Performance Graphics Tools

2901 Wilcrest Dr., Suite 300
Houston, Texas 77042
www.int.com