Lean Agile PM: Applying Agile and Lean Practices to Managing Projects

by Steve Blais

Accommodating change in the software development process is the nemesis of all development projects. The generally accepted practice in linear methodologies to accommodate change is to freeze the accumulated requirements at a specific point in time. Change is not prevented nor are new requirements prohibited. The freeze simply slows down the change process because all changes have to be justified to management before acceptance.

The good thing about a linear method is that necessary and appropriate changes get made with oversight and are documented in regulated environments. The bad thing is that it takes time. Both the customer and the development team often view the process as bureaucratic, time-consuming, additional overhead, and simply unnecessary.

The project manager's focus is on minimizing the impact of the changes on the project in progress to keep the schedule and resource targets on course. Project management tools are great for pointing out the impact to the project when a change is made – add requirements that cause a task to extend a few days, and look at what it does to the rest of the project! However, a project management tool *cannot* show you the impact of *not* making the change: what is the affect on the business or on the ultimate success of the product?

In today's dynamic business world, we need to transform our project planning focus from an expanding control change process to an adaptive process that embraces change: a process that responds to business requests whenever they are made, a process that is flexible enough to change even the process itself. In a word, we need an *agile* development process.

Agile software development methods have introduced a number of practices that can contribute to a software development project's flexibility and responsiveness. These practices are not limited to use in Scrum, Extreme Programming (XP), Feature Driven Development (FDD), Microsoft® Solutions Framework (MSF), and other branded Agile methods and processes. All development processes can gain from the implementation of any of these methods' core Agile practices.

Iterate Incrementally

Allowing change to be introduced during project execution without affecting the project is the benefit of one Agile practice – the iterative approach. This approach breaks the overall product into smaller pieces: for example, a stack of index cards that relate user stories that define the desired functionality, a list of features that are grouped together in a set, a selection of requirements from a backlog list of the required solution elements.

Each iteration of the software development life cycle (requirements definition, analysis, design, code, and test) executes the same phases many times, making multiple smaller deliveries. Each successive iteration:

- Follows the same development process as the first.
- Delivers something more of value for the customer to review and approve before the next iteration commences.
- Allows change to be introduced at the end of each iteration.
- May be a separate task within the overall project or a project in itself.

The customer and the development team collaborate to select each iteration’s specific deliverable. The amount of functionality each deliverable provides is determined by using another Agile practice: the time box.

With time boxing, the project manager asks, “How much of the solution can get done within this time frame” (how many user stories, features, requirements)? The time frames are typically short – two to four weeks – and, when added together, provide a more realistic estimate for the entire solution.

The customer prioritizes, the developers estimate, and a small portion of the entire solution is developed. In a few weeks, the customer sees the results. In this way, the customer can better understand and react to what they are getting than if the entire product was described in a requirements document that was approved several months earlier.

Each incremental iteration brings the product one step closer to the whole. And the customer has the option at the end of each iteration to continue with the current progress, introduce changes, or even stop the project. Any option is a positive outcome for the project.

Bring Everyone Together

In another core Agile practice – include the customer early and often in the software development process – the customer is represented on the development team. The representative is designated as the “On-site Customer,” the “Domain Expert,” or the “Product Owner,” depending on the brand of Agile.

Whether the representative is a user who sits with the development team or a small team of users who are on call to answer questions and provide direction, both the development team and the customer realize that they are on the same team with the same goal: solving the business problem.

Including the customer has caused some resistance to Agile approaches. Organizations are not usually inclined to allocate resources to a full-time development team. The Agile point of view is that the customer must understand that the quality of the delivered solution is increased in proportion to the time the customer spends with the solution team.

Through more frequent association with the customer, the development team learns the language and the issues behind the problem being solved and is better able to apply the correct solution. The customer learns the difficulties and tradeoffs that are intrinsic in software development. Both learn more about the problem domain and the resulting solution. And, even if nothing but propinquity is at work, increased collaboration and a more agile and flexible solution are gained.

Keep It Simple

Perhaps the core practice that underlies the whole Agile process is "do the simplest thing that will work." Agile processes support writing only enough software to solve the business problem and only what will be used today. This implies just enough requirements, just enough design, just enough testing: a lean concept wherein everything is *just in time*.

Agile methods also endorse the practice of refactoring – taking the time to simplify code and design to eliminate anything unneeded or redundant.

A simpler product and a simpler process make changes to both also simpler. Keeping the product and process lean and flexible allows the project manager to embrace change rather than fear it.

Conclusion

The fastest solution and the solution with the highest quality is one that simply solves the business problem. Agile practices are all about returning the focus of software development to solving business problems by using computer technology. Focusing on solving the business problem in the simplest way makes the process responsive, flexible, and ultimately, agile.

Project management's responsibility is to continually ask, "Does this activity move us closer to the solution of the problem?" Eliminating those activities that elicit a negative or even a questionable response and emphasizing those activities that move the project in the right direction make the overall project more agile.

Steve Blais, PMP, is a highly experienced project consultant and educator with more than forty years in the field of computing. Steve currently works with companies to create and improve their business analyst processes and is also the author of IIL's Business Analysis courses including his forthcoming book*, The Beginning and End of Software Engineering: A Guide for the Business Analyst*.