Lante Case Study

SpreeRide, Lante, and Agile Methods

Business and Technology in Collaboration

A Successful Experience

Agile Methods is a software development approach that depends on a close collaboration between technology and business teams. This method creates software solutions that are inherently adaptable, dependable, and much less costly than solutions developed using the traditional "waterfall" approach. It is an emerging practice that shows great promise for eliminating problems such as cost overruns or unmet deadlines that are all too common using the traditional approach. This case study describes how, despite fluctuating functional requirements, Lante successfully used Agile Methods to implement a software solution for SpreeRide Corporation. This solution is in production today as their patent pending MC^2, a comprehensive, accountable, integrated marketing system.

Agile Methods vs. Traditional Development Methodologies

Although the development method presented here might be considered unusual or even extreme, today's business environment is evolving in a direction that demands a very adaptive kind of software development practice. Traditional development methods often create antagonism between project stakeholders due to a lack of communication. Agile Methods fosters a collaborative rather than combative atmosphere because the stakeholders must meet face-to-face each week to discuss the project. Because traditional methods usually insulate the client from development activity, an additional effort to manage customer expectations is usually required. With Agile Methods, expectation management is a built-in feature, and the client is able to see business value implemented on a daily basis. With the traditional process, functional requirements are fixed at the end of the requirements phase and remain so for the course of the project. Any requirement changes usually involve costly and time-consuming revisions to the code, forcing the whole project to go off schedule. With Agile Methods, requirements are fixed for a week's time, allowing the project to change at the same velocity as the business. Traditional development methods do not usually produce a functioning system until the very end of the development phase, and client support for a project often becomes stale, especially after a great deal of money has been spent with nothing concrete to show for the investment. With Agile Methods, the growing functionality of the system is demonstrated every two weeks, thereby justifying continued support for the project.

Because the Lante technology team rigorously applied Agile Methods, SpreeRide was able to change objectives in the middle of development to meet new business demands without seriously disrupting the progress of the developers.

The Need for a New Approach

SpreeRide developed a well-founded business plan and identified all of the functional requirements necessary for a system to implement that plan. However, given the volatile

nature of the current business environment, SpeeRide needed to be able to change the requirements of the system quickly to meet evolving market opportunities. Knowing that changing business requirements did not work well with the traditional waterfall cycle of development, SpreeRide engaged Lante to develop the system using Agile Methods.

A Different Way of Working

In accord with Agile Methods practice, the business team headed by the CTO met with Lante's technology team at the beginning of each week for "planning game" sessions to determine the work that needed to be done for the week. Initially, the system functions had been prioritized according to current business needs using **M**O**SC**O**W** lists (an acronym for **Must** Have, **Should** Have, **Could** Have, and **Won't** Have). Under the terms of the engagement, only the functions in the "Must Have" list were to be implemented during course of the project. It was understood that once the Must Have functions were working, the client would reprioritize the remaining functions or request new functions during the weekly planning game.

At the conclusion of the planning game, a pair of Lante developers would take responsibility for implementing a function using a "test first" approach. A test that described a specific behavior would be composed and then code would be written to implement that behavior in the program. This method of incrementally writing tests and code through a series of iterations provided the developers with a reliable way to consistently produce quality work.

Demonstrable Results

One important objective of Agile Methods development is to always work towards providing demonstrable results. Although at first a fair amount of "backstage" work—such as setting up the development environment—needed to be accomplished, at least one visible function was implemented each week so that progress could be demonstrated to the client. As each week progressed, fewer backstage efforts were required and more attention could be given to developing visible functionality.

The use of Agile Methods also ensured consistent progress for Lante because pair programming was required. If anyone were unable to come to work, he or she could do so without restraining the progress toward implementing a function because the other partner would team with another developer to continue the work. Pair programming also allowed for mentorship and promoted high team morale throughout the project.

The Fifth Week Challenge

Five weeks into development, SpreeRide's objective changed from creating a demonstration system for a specific client to creating a generic version that could be adapted for immediate demonstration on many different clients. This required a change in the look and feel of the demo, a change in workflow, and a change in the deployed solution. In response, Lante developers repurposed the existing code to meet the new objective and created an impressive working demo that SpreeRide was able to show to prospective clients without a serious impact on the development timeline.

A Change in Direction

The first phase of the project delivered within 1% of budget, and because Agile Methods had been proven effective, a new statement of work was created to continue development

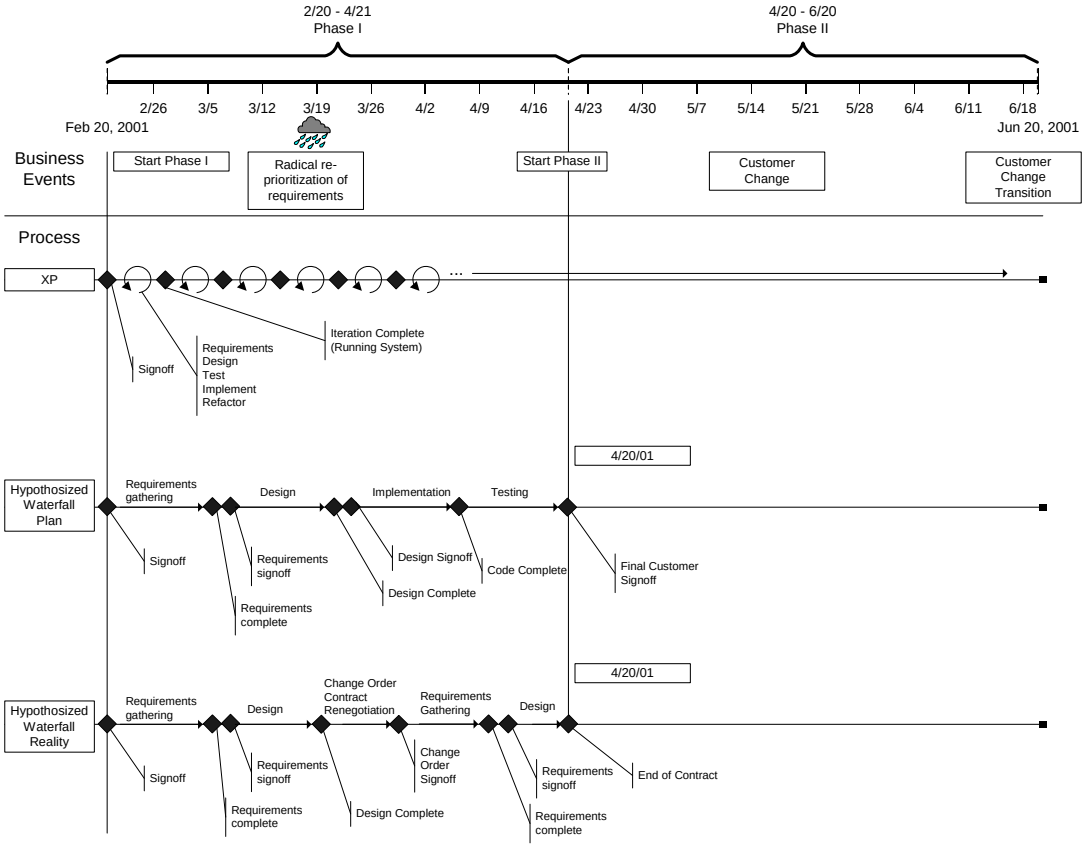
with this approach. The next goal was to develop a self-contained sales tool that would run on a laptop computer. This was a completely fixed cost, fixed time frame, optional scope engagement designed to meet SpreeRide's next goal by a specific date. This change in direction did not hinder the development team because they were only required to focus on the new set of functions each week.

A Change in Leadership

SpreeRide's CTO was the original leader of the business team, acting as both the technical point person for SpreeRide and the client representative. Towards the end of the second phase of the project, the CTO turned his role over to SpreeRide's COO. This change in leadership also meant a shift in business priorities. The impact to the development team was minimized by the weekly planning game sessions that gave the team an opportunity to provide feedback to the COO about the impact the new priorities would have on the system. The client continued to enjoy a clear view of the development activity via stand-up meetings each morning and bi-weekly steering committee meetings with other executives. The client's expectations were managed on a daily—almost hourly—basis because everybody was aware of the actual development progress.

Conclusion

The diagram below provides a vivid picture of Agile Methods in contrast with the traditional waterfall approach.



The traditional waterfall approach to development requires a team structure that fluctuates per phase, is more hierarchical, and requires more management due to the layering and coordination of various competencies. In a traditional approach, the project moves through a requirements phase, followed by a design, an implementation, and finally a testing phase.

In a waterfall approach, the requirements team attempts to identify the functional requirements for the system within the first two to three weeks of a new project. The client is then asked to confirm that all requirements have been gathered. Because the requirements documentation is often long and very detailed, the client might blindly approve the requirements, potentially jeopardizing the project, or might take an inordinate amount of time sifting through the details, effectively stalling the project. Once requirements have been approved, the design process can commence. Substantial changes to the functional requirements during the design phase have a disruptive effect on the project, triggering a change request process that forces everyone to start all over again.

For example, had SpreeRide's change in objectives occurred during the design phase, a change order calling for new requirements and new implementation timelines would have been required, a process taking several days to weeks. If SpreeRide had been using the traditional development method, the project would most likely have ended prematurely because a working demonstration for prospective clients could not have been constructed in time. Essentially, SpreeRide would have walked away carrying no more than a stack of requirements documentation and any design documentation completed up to that point.

The successful use of Agile Methods in Lante's engagement with SpreeRide illustrates how a development team can produce a quality solution while continuing to be responsive to changes in functional requirements. This new way of building systems provides functionality each step of the way, allowing businesses to gather real-time feedback and to factor that feedback into the development cycle on a weekly basis.