

Agile Development

SCRUM meets Web 2.0

White Paper

“The dotcom bust, the advent of Web 2.0, and the lingering recession are making the need for managing large software projects obsolete. Most businesses seek to leverage specialized solutions or applications and integrate or repurpose them to solve pieces of a larger solution.”

— Charlie Marval
Principal – ovionx

“Newer more customer-driven paradigms are moving towards a more adaptive approach that centers on business goals and customer needs”

Introduction

A camel is a horse designed by committee. IT projects devolve to a committee process. While the goals of technology and innovation as it relates to business are to solve real world business problems, 68% of all IT projects fail to achieve these goals largely due to poor requirements analysis. The percentage is even higher regarding software development. One key aspect about software development and the primary cause for budget overruns is the lack of a process by which to manage expectations. In most cases, either developers fail to manage expectations or customers provide ambiguous definitions of business problems. This leads to solutions designed around a democratic process ending in a mis-match of ideas that create something, but that something fails to meet the intended goal of solving the original business problem.

There are competing methodologies designed to address the complexities and risks associated with software development. The so-called Software Development Life Cycle (SDLC) methodologies have been found wanting because of their rigidity and their focus in architecture. SDLC describes the steps a typical software development project goes through. The methodologies describe the how to implement the SDLC. One such example is the waterfall methodology, which dictates a set of strict and disciplined steps: requirements gathering, analysis, design, implementation, and maintenance. The rigidity of these

types of methodologies is not customer-driven and in many cases precludes the customer from reprioritizing and changes in direction.

Problem

Newer more customer-driven paradigms are moving towards a more adaptive approach that centers on business goals and customer needs. Such an approach is the Agile Software Development (ASD) methodology. There are a number of models to this approach, but the two most popular are SCRUM and Extreme Programming (XP).

XP focuses on a highly iterative process whereby the primary focus is on short development cycles that include a small piece of a problem and a working solution. In between these short cycles, the customer can introduce new priorities or requirements.

While SCRUM also introduces an iterative process, its primary focus is handling larger projects by introducing a set of rules governing the roles of each team member and a tactical approach to conducting meetings and setting milestones known as Sprints.

“The dotcom bust, the advent of Web 2.0, and the lingering recession are making the need for managing large software projects obsolete”

Both approaches revolve around project stakeholders and involve gathering feedback from the people for which the software is built. Some key differences and incidentally controversy are the level of formality in documentation, level and type of participation, risks, and scalability.

Documentation: XP is extremely light on the documentation and can lead to a number of problems including, not the least, business continuity. Conversely, SCRUM's focus on documentation of functional requirements can be exhaustive. The SCRUM approach could lead to analysis paralysis. The costs involved both with the maintenance and management of documentation are high and the benefits marginal.

While the XP approach requires little documentation, a compact vision and scope document followed by practical functional requirements documentation is enough to proceed. The functional requirements documentation can still be filtered through the vision and scope document effectively enough to proceed.

Customer Participation: The XP approach mandates that a "Champion" of sorts on the customer side to be constantly involved. Proponents of SCRUM argue that this approach leads to a single point of failure where there is a monopoly on ideas or micro-management by the customer representative. The reality is that the more inclusive participation SCRUM implement is seldom practicable and there is almost invariably an ultimate decision maker.

When there is active, healthy, and intimate partnership between the developers and the customer, the issue of single point of failure is overcome by the developers relaying business continuity information to a new customer representative.

Risk Adversity: No doubt that the XP approach exposes more risk by minimizing the focus on 'The System.' While the model concentrates on adaptability to customer requirements and less on architecture, the SCRUM approach involves a much longer process. This costly and prolonged approach leads to executive anxiety and loss of interest for fears that the end product would outlive its usefulness. The XP approach avoids the "mega-review" that leads customers to do nothing at all through its low risk high reward potential.

Scalability: XP is designed to work with small groups (10-20 peers in a group) and does not work well with big projects with a large number of participants. While SCRUM has well structured techniques for handling large groups, it is precisely this that makes SCRUM cumbersome when used in the current environment:

The dotcom bust, the advent of Web 2.0, and the lingering recession are making the need for managing large software projects obsolete. Most businesses seek to leverage specialized solutions or applications and integrate or repurpose them to solve pieces of a larger project or goal.

“Large scale projects have been replaced with integration protocols and middleware that seek to connect solutions efficiently and effectively”

There is still a continuous stream of solutions flowing from the dotcom days. These solutions, bought and paid for by seemingly bottomless budgets, have become available for pennies on the dollar. The days of large scale projects have been replaced with integration protocols and middleware that seek to connect solutions efficiently and effectively.

Furthermore, the advent of Web 2.0 mashup, the concept where an application combines data or functionality from two or more external sources to create a new service, is gaining wider acceptance. Once a dream now a reality, newer more efficient protocols to exchange information means that real portability and reusability can be achieved. This, coupled with the providers' appetite to please, means this external sources are willing to twist and turn to accommodate any flexibility needed. This avoids the not-invented-here syndrome and moves to leverage existing resources.

Finally, the projections for the current economic environment are grim. At least in the near future, organizations are just not willing to commit to large scale projects where they can't realize quick and palpable ROI.

The need for instant gratification, coupled with the need to having to constantly reinvent themselves in an environment that is picking winners and losers on a daily basis, simply does not allow for extensive planning and commitment to large scale projects.

Solution

Most organizations resort to the supermarket approach where they pick and choose elements of different approaches and tailor a solution that best fits. The XP approach fits most small to mid-size organizations. However, there are several shortcomings that can be addressed by borrowing elements of the SCRUM model with some additional modifications. Let's take a brief look at "backlog of items" and "business continuity."

There is no clear way in the XP model to address backlog of items that result from the continuous, successive releases. As work develops on each of the features, there will be items that simply take longer and miss the deadline. Meanwhile, items that were scheduled in a release have to be pushed back creating a backlog. This backlog if unchecked could impact business goals and deadlines that parts of the solution were suppose to address. Implementing the Sprint or milestone approach prescribed under the SCRUM model ensures the approach to the backlog employs some form of checkpointing.

This checkpointing, preferably every 30 days, should include all stakeholders' and end-users' feedback. These meetings should be solely focused on reprioritization and reevaluation of all features queued to be worked on.

“It is imperative that any agile development model addresses business continuity”

It is imperative that any agile development model addresses business continuity. How does the model ensure continued progress when faced with the inevitable loss of a representative on either side, customer or developers?

Documenting each feature with a vision statement serves this purpose. An effective vision statement would not have to be extensive, simple is better, but it does have to include a simple statement with the following components:

- ❖ Target users/customers
- ❖ What they need to be able to do
- ❖ A solution name
- ❖ Description of the solution
- ❖ Description of what the solution has to be able to provide:

"For [target users/customers], who need to [what they need to be able to do], the [project name] is [description of the project]. Unlike our current solution, this solution will [description of what the solution has to be able to provide]."

Each feature in the functional requirements document should be filtered against this statement to determine if the feature falls within or outside the vision/scope of the solution.

A simplified version of functional requirements documentation should concentrate on breaking down requirements in terms of three elements:

- ❖ Functional requirements (what it must do)
- ❖ Non-functional requirements (how it must do what it does)
- ❖ Assumptions (from both customer or developer)

Benefits

Leads to a cycle of continuous improvement

This modified approach to XP provides with a greater level of flexibility. More importantly it makes collecting data about the effectiveness of an implementation more manageable and leads to a cycle of continuous improvement.

Increased level of control

Bigger is not better when it comes to controlling budget and resources. Smaller working pieces make cost analysis easier to manage and control.

For more information

For more information about ovionx products and services, call us toll free at 1-888-OVIONX-NET (1-888-684-6696) in the US. Outside the US, please call 1-813-886-2554. To access information using the World Wide Web, go to: www.ovionx.com.

Synopsis

Organizations continue to struggle in their attempts to manage software development but the rewards are great for those who succeed. Moving to shorter development life cycles using the Extreme Programming (XP) model as an agile development approach leads to a more manageable and controllable process and leads to a cycle of continuous improvement.

The opportunities to leverage existing solutions inherited from the dotcom bust and the shift in paradigm to Web 2.0 mashup, is making software development models like the waterfall and SCRUM unnecessary and cumbersome for the management of large projects.

A modified version of XP is a powerful approach especially in the current economic environment where companies are finding it necessary to integrate and automate their business processes in order to stay competitive. This is critical to greater business profitability. The advantage of taking smaller steps means businesses can be flexible and adaptable in a changing marketplace.

References

Krigsman, Michael. 2008. "Study: 68 percent of IT projects fail."
<http://blogs.zdnet.com/projectfailures/?p=1175> (Accessed May 27, 2009).

Wieggers, Karl. Software requirements: Practical techniques for gathering and managing requirements throughout the product development cycle. Microsoft Press, Washington. 2003.

Schwaber, K.. "SCRUM Development Process: Advanced Development Methods".
<http://jeffsutherland.com/oops/schwapub.pdf>. (Accessed October 2, 2009)

Wikipedia. 2009. "Agile software development."
http://en.wikipedia.org/wiki/Agile_software_development (Accessed Oct 4, 2009).

About ovionx

ovionx is a specialist company that focuses on helping our clients find and take advantage of business opportunities through our agile development approach and deep experience we do extraordinary work for B2B clients to achieve operational efficiencies and compete more effectively.

ovionx's offerings range from enterprise application integration, supply chain automation, customer relationship solutions, and anything else that helps to take your organization to the next level and become a leader in your industry.

© 2009 ovionx corporation. All rights reserved. This case study is for informational purposes only. OVIONX MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. ovionx, the ovionx logo, are either registered trademarks or trademarks of ovionx corporation in the United States and/or other countries. All other trademarks are property of their respective owners.