A False Measure of Success

"I'd rather have an ounce of cure over this 200 pounds of prevention"

Richard K. Cheng
Level One Consulting
rcheng@l1consulting.com

Abstract

Several years ago, I was working at a major US financial institution on their web development projects. When I started with this company, I was initially brought in as a consultant to perform web development. Over the course of the next 5 years, I had converted over to a full time employee with the company and served a variety of roles including developer, team lead, architect, and project manager of web projects.

This experience report examines the changes in infrastructure, workflow and processes during my time with this company and the results of these changes. The companies' web development teams started with a software process that was not perfect, but did satisfy the teams' internal business customer. Customers were pleased with the rapid turn-around time that the technology teams provided. Any issues or defects that occurred were within acceptable tolerances for the customer's business needs. However, the management teams of the technology division mandated implementation of processes to ensure stability, redundancy, and uptime. Employee goals and financial bonuses were updated to measure qualities such as 99.99% uptime, full redundancy and zero defects. But as the development teams came closer to their technology goals, the cost of projects increased and project turnaround time decreased. Defining "What quality level is acceptable" and "What cost is acceptable" was shifted from the Business teams to the Technology teams. Customers were no longer pleased with the performance of their technology counterparts, thus creating A False Measure of Success.

1. Business goals

At this financial institution, the business goal of these particular web development teams was to disseminate stock market data to a variety of target audiences. This included institutional customers, regulators, and home investors. As customer demand for different data products was discovered, the business units stressed the need to quickly deliver information to their customers.

2. The workflow

In typical fashion, the business teams used analysts to represent their interests. A business analyst worked with the technical leads to plan the features for each release. Requirements and documentation were sparse and communication was informal using phone calls, face-to-face discussions, and emails. The process for each release involved iterative designs on the data and the web sites.

The business units would then try to throw in as many additional features as possible, with the idea that the development would be time-boxed to be about 5 days for the development team. Everything that could be done in those five days was the workload for that release.

Once development was completed, developers met with testers to explain and demonstrate the new functionality. Once testers began testing for a release the process was time-boxed to about two days.

Once testing was completed, the lead developer created a set of instructions for installing the release into production. The lead developer then worked with the operations personnel to perform the file transfers and installation procedures needed for the release.

Once the system was released into production, the business, operations, and development teams monitored the production site for the next couple of days. There were usually some fixes that would need take place after the release, but everyone was aware of this and in the majority of cases, these were minor and within the business teams' tolerance levels.

www.manaraa.com

Based on this workflow, the standard project release was about a two week process (see **Table 1**).

Table 1. Process Workflow

Process Steps	Duration
• Working with the business units to discuss business needs	1 to 2 days
• Development	5 days
• Testing	2 days
• Production Deployment	½ day
• Post Production Updates	2 days

3. Not a Perfect Process

The process, though not perfect, met the needs of the business teams and their customers. The process allowed for the business units to create a very quick turnaround from business vision to production. Since the business units worked very closely with the development team during the development phase, the final product most often met business needs. In essence, it was a very Lean process [1].

Despite the benefits that our team experienced using this process, we also experienced limitations when attempting larger projects and problems with testing and documentation. The biggest issues with the process are described in more detail in the next sections.

3.1. Ability to handle larger projects

Since the development work was usually time boxed to five days, issues would arise when projects required more time. In some cases, the development team would try to squeeze all of the work in five days. Other times, the work was broken into what could be done in five days. Still other times, the releases could be extended. But when using any of these alternatives, the process was not a natural fit and side effects included poor testing, "heroic programming", and inadequate functionality in interim releases that did not yet fulfill business needs.

3.2. Testing

Since the testing was time-boxed, it wasn't so much a case of whether the testing was complete, rather a *"time's up, time to ship"* mentality. This resulted in incomplete testing at times.

3.3. Defects

Due to the rapid development and short test time, errors did occur in production. Most often, errors were within business tolerance levels. There was an expectation that each release would have a certain amount of acceptable defects.

3.4. Lack of documentation

There was very little in terms of formal documentation. Thus, if the management teams ever asked for documentation or if we did need to reference documentation, there was not much in place.

4. Technology Goals

New executive management wanted more control of their technology investments and new technology goals were defined that measured the *"effectiveness"* of technology teams and team members. The emphasis was product stability and fault tolerances. Events such as Y2K followed by the tragedy of September 11th and Sarbanes-Oxley all helped to place great importance on measures such as:

- 99.99% Uptime
- Full redundancy
- Disaster recovery
- Zero defect

The Operations and Testing departments were given strong executive support to implement processes to address these issues. The idea was to create standards across all systems, ranging from mission critical markets systems through to more specialized web projects.

As a result of these new technology goals *all* servers and application tiers had to have a redundant onsite and offsite system with warm or hot failovers. This required considerable investment in equipment, new licensing and upgrading of existing licenses to support enterprise wide solutions. Costs were also incurred for the resources that implemented the changes, including operations resources and the developers needed to re-engineer systems to support the changes.

For the zero defects policy, yearly bonuses were paid to members of technical teams, partially based upon defect rate (as well as uptime). These yearly bonus ranged anywhere from 5% to 45% of the employee's yearly salary, thus employees were very motivated to release stable code. However, this meant that fewer risks were taken and changes did not occur nearly as quickly as they once had. The business units

www.manaraa.com

may have been pushing for faster development, but the technology divisions had other goals to meet and they did not want a decrease in their perceived effectiveness and bonuses.

5. The new process

In order to meet the technology goals, the project process was updated to mitigate risk. It was commonly thought that a more traditional waterfall approach with appropriate document artifacts, which were lacking in the previous process, was needed. Thus the following workflow was established:

5.1. Requirements

The business units gathered requirements and were required to produce a requirements document. This document contained all of the functionality needed for the release.

5.2. Project planning

Based upon the requirements, the technical lead/project manager generated a task based Gantt chart for the release. Using level of effort estimates, the Gantt chart dictated the total project timeline. Technology manager's goals were also based upon being able to accurately estimate project plan dates, which resulted in more conservative timelines and longer project cycles.

5.3. Development

During the development cycle, the development team still worked closely with business analysts. Tasks were assigned based upon the resource assignment on project plan Gantt charts.

5.4. Testing

In terms of planning, the general rule was that testing would take 33% of the project time. Thus testing was still time-boxed. Testers now based their testing on the documentation, combined with help from the developers and business analysts. During the testing phase, there was a daily meeting involving all team members (business analysts, developers, testers and operations) to discuss the status of the testing.

5.5. System Deployment

Releases were built with an one-click installation process with automated rollback. The installation required quite a bit of development work, particularly for automated rollback. The systems in question were four tier systems that spread across multiple servers, locations, and database systems that affected multiple datasources. Each release took about a day or two to design, test, and implement. The developers shipped the install package to operations personnel. The operations personnel then performed the installation on production systems.

5.6. Post production

To ensure production system stability and security, a corporate policy was mandated where access to production systems was limited to operations employees. This introduced a new problem when the system failed since the developers' best equipped to track down data or code issues could not access the system directly. Without access to these backend systems, the developers could not act quickly to address errors. This caused delays in resolving errors or system malfunctions.

5.7. Project timeline

The timeline in Table 2 roughly typified the new timeline for a project similar in scope to that in **Table 1**. Extra time was consciously added to the schedule to ensure system stability and comprehensive testing.

Table 2. New Process Workflow

Process Step	Duration
• Requirements and Project Planning	2 weeks
• Development	2 weeks
• Testing	1 week
• Deployment	2 days

6. The results

Over time, all of these procedures did get implemented and changed the approach to system development. System stability did increase, and for the most part the technology goals were met. From the technology division's point of view, the changes were a success.

However, when talking with the business units, it was clear that they were extremely unhappy. The business unit's costs had increased an order of magnitude and they had very little to show for the

www.manaraa.com

additional cost. Furthermore, the project turnaround time had approximately tripled, yet each project release still had roughly the same number of features. The primary goal of the business units to quickly place products on the market was not met, and to make matters worse, costs kept rising. There was not much the business units could do to change either problem because the changes were mandated at executive levels and technology leaders were charged with carrying them out. In order to reduce costs and improve efficiency, the business units began to consider outsourcing a portion or even all of their projects.

Even the system stability did not offset the costs. Business tolerances for these web projects were such that partial outages of up to a couple of days or even a full outage of a couple of hours were acceptable. I recall a quote from one of the executive vice presidents, who had stated, *"I'd rather have an ounce of cure over this 200 pounds of prevention."*

7. Responsibility split

From a technology standpoint, much of what was accomplished was truly outstanding. The architectures implemented for redundancy and failovers was extremely sophisticated and innovative and the defect and failure rates did drop significantly.

However, the main issue was losing focus on the business needs. The business units did not care about much of this extra work, yet they were the units footing the bill and had little say in how their money was spent.

In Mike Cohn's Certified ScrumMaster seminar, he presents the following in terms of responsibility split:

Responsibility Split [2]

- *Business determines*
  - *When a release is needed*
  - *What functionality it must contain*
  - *What quality level is acceptable*
  - *What cost is acceptable*
- *Development determines*
  - *How long it will take to develop each piece*
  - *How much they can commit to each [release][3]*

What had taken place was the quality level and acceptable cost were taken away from the business unit's realm and placed in the realm of the technologists. These technologists did not always take the ROI into account when implementing projects and infrastructure to satisfy the goals handed down from their management chain.

8. The Agile Perspective

In retrospect, much of what was accomplished was pretty sound. The original workflow and process, though they did meet business goals, did have drawbacks. It was undisciplined and did rely too much on heroic and cowboy programming. Thus a new methodology was a good idea, as long as it did not lose focus of the business goals. However, the heft of a traditional waterfall approach was not the methodology best suited to accomplish the business goals of these web projects.

Although on the surface the new methodology seemed like a traditional waterfall approach, in practice it was not far from an Agile Scrum process. There were already Scrum-like [4] aspects in place such as:

- *Defining the product backlog*
- *Daily meetings*
- *Project managers who were already unknowingly filling many of the ScrumMaster roles*
- *Defined business analyst on the team acting as the Product Owner*
- *Time boxed projects*
- *Self organizing teams*

With some fine tuning, this could have been converted to Scrum projects. This fine tuning includes:

- *Scrum training for all team members*
- *Train the Project managers as Scrum Masters*
- *Reworking the requirement phase to reduce waste and adapt and "last responsible moment" philosophy [5]*
- *Developing the code such that it was constantly ready to implement.*

By running the a more Agile methodology, such as Scrum, the process could have been tweaked to deliver a coherent structure to meet the technology stability, reliability and availability goals while still satisfying the business needs.

In terms of the infrastructure and architecture, what was lost was the business voice in technology decisions. The technologists and business units should have been working together to not only ensure that the technologist understand the business needs, but also to educate the business customer as to their available options, the cost of their options, and what each of these options deliver in terms of value to their projects. Once an educated business unit understands their options, they should have at least a strong voice in what the acceptable cost and quality levels are for their products.

9. References

www.manaraa.com

[1] Poppendieck, Mary, and Poppendieck, Tom, *Lean Software Development*, Addison-Wesley, 2003.

[2] Cohn, Mike, "Certified ScrumMaster Seminar", Mountain Goat Software, 5/10/2005.

[3] Cohn, Mike, "Certified ScrumMaster Seminar", Mountain Goat Software, 5/10/2005. For the purposes of this paper, the paper author replaced the word "sprint" with the word "project".

[4] Schwaber, Ken, *Agile Project Management with Scrum*, Microsoft Press, One Microsoft Way, Redmond, Washington, 2004.

[5] Poppendieck, Mary, and Poppendieck, Tom, *Lean Software Development*, Addison-Wesley, 2003, pp. 57-60.

www.manaraa.com