Lean Software Development

Expect Zero Defects

- Mark Windholtz, www.objectwind.com

$59.5 Billion wasted! [i] The U.S. gross domestic product loses 0.6% a year because of software defects. *Currently, over half of all errors are not found until "downstream" [...] or during post-sale software use*. This occurs even though vendors already spend 80% of development costs on defect removal.

The software industry is becoming increasingly interested in removing defects after they are created. A focus on defect removal sounds like a good idea. What if the defects are not created in the first place? What would that be worth? Is it realistic to suggest that software can be built without defects? Teams using Lean Software Development are demonstrating that it can be done. To see what we mean by Lean, let's start with what is not Lean.

"Scientific" Project Management[ii]

Henry Ford built his early automobiles using "scientific" management techniques. Every production step was optimized for return on investment. Cars were shipped directly from the factory to the dealer without testing them. The focus was on efficient, not quality production. As the auto industry grew factories added testing and repair stations at the end of the line to catch defects before the customers did. It was a helpful innovation. Quality assurance became focused on that final phase. When defects were found components were disassembled and fixed. The cost of this re-work was high.

Parallels can be found in how most of the software industry builds software today. Programs are either released to the customer for testing, or they are tested as the last step before release to the customer. And much of the current focus of quality assurance is on finding defects at the end. Is it reassuring that we are following the same path as the automotive industry? Certainly if we do a good job of testing at the end, many of the defects will be caught.

Testing cars at the end does find the defects. But since a mass-production factory produces in large batches, many cars are built with the same defect before the defect is detected at the end. Consequently many cars need to be re-worked because of the same defect. When cycle time between error, detection, & correction is long it causes repeated errors to occur before correction. Now that is expensive!

Could this be why 80% of development costs are spent on defect removal? If we wait until the end of a month-long cycle to test, programmers will introduce similar kinds of defects across the system. Repeated defects in software occur when communication with the customer breaks down causing multiple wrong features to be built; when wrong assumptions are propagated; or when

programmers don't share information among themselves and multiple programmers make the same mistake. These defects show the limitations of long cycle times.

Lean Manufacturing

In the 1950's a revolutionary concept was introduced. The Toyota Motor Company began testing at every station. They also reduced batch sizes so that cycle time was reduced, defects appeared quickly, and were corrected quickly. When cycle time is short, the number of components assembled with the same defect is reduced. This means less rework and lower costs.

Still today most software is built in long cycles from three months to many years and often not tested until the final step of the process. Lean Software Development provides a different vision. Quick cycles, rapid, continuous, automated testing, & close interaction with the customer provide the software that the business needs when it is needed.

Lean Software Development[iii]

The software industry finds itself in a situation today similar to automobile manufactures in 1970's. Defects are sapping resources and angering customers. Defects and long cycle times are reducing our ability to respond to new business opportunities. The most heavily marketed solution is to add more bulky process steps and to do more testing at the end. This approach lengthens cycle times. Meanwhile the Lean approach is delivering successful projects using processes with unlikely names like: Scrum, DSDM, and Extreme Programming (XP). Lean Software Development reduces defects and cycle times while delivering a steady stream of incremental business value.

Cycle time for software development is measured in the number of days needed between feature specification and production delivery. This is called: *Software In Process* (SIP)[iv]. A shorter cycle indicates a healthier project. A Lean project that deploys to production every 2-weeks has a SIP of 10 working days. Some Lean projects even deploy nightly.

Going to production every 2-weeks may raise fears of introducing new defects. Let's deal with that. In order to achieve short cycle time we need to eliminate defects. Doesn't that sound pretty straightforward? Start by expecting zero defect software production.

Expect Zero Defects

If you expect zero defects a fundamental change in thinking occurs. Do I mean to say that no single defect will ever occur? No. I mean that we treat every defect as something really, really bad, and figure out how to prevent that class of defect from occurring again. Lean Manufacturing production lines allow each assembly worker to stop the line if a defect appears. Paradoxically, lean production lines almost never stop. In "Scientifically Managed" production lines, only the manager can stop the line, yet they often crash because defects are not reported creating bigger problems downstream. Expect zero defects and you realize that the way you schedule, test, program, and release software all needs to change to accomplish your goal. And since 80% of cur-

rent software project dollars are spent removing defects the goal of Zero Defects becomes financially justifiable.

Extreme Programming has a practice of Test-First coding[v] that is fundamental in achieving the goal of Zero Defects. Test-First coding requires that an automated unit-test is written before the production code and the test-to-code cycle time be 5-15 minutes long. Additionally, automated acceptance testing insures that the correct business value is delivered. Acceptance tests are built directly from the requirements before the production code is written. All tests are built one at a time immediately before each piece of production code. Moving the tests to the front of the process gains efficiencies in providing a tight disciplineand direction to the team.

Increasing The Business Payback of Projects

Finally, there are two ways to increase the payback on a software project: **lower the cost** or **increase the value**. Lower costs can be accomplished with offshore development. Understandably, corporate executives are unhappy with paying high dollar rates for poor quality software built locally. Instead they reduce labor costs by moving projects offshore. Offshore programming gets low-cost, & talented programmers to produce the same poor quality software as before. Reducing cost increases payback.

An alternative way to increase payback is to increase project value. This can be realized by savings of time to market, closer understanding of the real customer requirements, and customer goodwill by providing defect free applications. Lean Software Development provides the business case for this approach. Extreme Programming provides the most popular way to implement Lean Software Development.

A closing thought. In the 1980's Japanese Lean Auto Manufacturersestablished engineering and production facilities in the U.S. Did they move development off-(Japanese)-shore for lower U.S. labor rates? No way! They now build cars in Ohio and Michigan, in order to be closer to the customer, to ship quicker and to understand more deeply. They moved closer to customers to produce more profitably.

The same occurs in companies that adopt Lean Software Development. Offshore may be less expensive, but when software is business critical, locally built Lean Software Development with short cycle times and low defects are the cost effective option.

References:

[i] The Economic Impacts of Inadequate Infrastructure for Software Testing

A report by the National Institute of Standards and Technology

http://www.rti.org/

[ii] The Machine That Changed the World: The Story of Lean Production
by James P. Womack, et al, 1991

[iii] *Lean Development: A Toolkit for Software Development Managers*

Mary & Tom Poppendieck (Draft), June 2003

www.poppendieck.com/ld.htm

[iv] *Software-In-Process, A New/Old Project Metric*

Kent Beck, Three Rivers Institute

http://www.threeriversinstitute.org/

[v] *Test-Driven Development, by Example*

Kent Beck, June 2003

The Goal, A Process Of Ongoing Improvement

E. M. Goldratt, 1984