An Introduction to Lean Software Development

by Mary Poppendieck, leanessays.com June 24th 2004

Gustaf Brandberg, CEO and co-founder of Citerus, a software consulting firm in Uppsala, Sweden, conducted this interview, featured in PNEHM! #2 2004, a newsletter on software development in Swedish, produced and distributed by Citerus.
Gustaf Brandberg: What is Lean Software Development?

Mary Poppendieck: Lean Software Development is the application of Lean Thinking to the software development process. Organizations that are truly lean have a strong competitive advantage because they respond very rapidly and in a highly disciplined manner to market demand, rather than try to predict the future. Similarly, Lean Software Development is the discipline of creating software that readily adapts to changes in its domain. There is a pendulum that swings from one extreme to the other – first there was ad-hoc software development, which could not scale to large complex systems. Then the pendulum swung far to the other side and heavy process discipline was favored, but this proved unresponsive to change in a world where change is constant. Lean Software Development provides a middle ground: high discipline along with high responsiveness to change.

Gustaf: When did you first start working according to Lean principles? From where did you get the inspiration?

Mary: I was working in a manufacturing plant making video tapes in the mid 1980's, and our Japanese competitors were selling video tapes at half of what it cost us to make them. We needed to understand how they could do this, and we discovered Just-in-Time production. We did two things: first, we provided every single worker in the plant with training in how a Just-in-Time flow works and had the workers design the details of a pull system. Then we stopped scheduling each workstation and instead sent a weekly schedule to the packing station. It worked like magic; inventory disappeared, quality improved, costs dropped, and customer response time was a week instead of a month.

Gustaf: You believe that there is no such thing as a 'best' practice. Why is that?

Mary: Frederick Winslow Taylor wrote The Principles of Scientific Management in 1911. In it, he proposed that manufacturing should be broken down into very small steps, and then industrial engineers should determine the 'one best way' to do each step. This ushered in the era of mass production, with 'experts' telling workers the 'one best way' to do their jobs.

The Toyota Production System is founded on the principles of the Scientific Method, instead of Scientific Management. The idea is that no matter how good a process is, it can always be improved, and that the workers doing the job are the best people to figure out how to do it better. In Lean Production, workers learn how to create a hypothesis, test it, analyze the results, and – if the data supports the hypothesis – make the change permanent.

Software development covers a lot of territory, and no matter how good a practice may be, it will not apply universally across all software development environments. Moreover, even where a practice does apply, it can and should always be improved upon. There are certainly underlying principles that do not change. These principles will develop into different practices in different domains, driven by the economic reality of each environment.

Gustaf: What are these core principles in Lean Software Development you are referring to?

Mary:

- **Eliminate Waste** – Do only what adds value for a customer, and do it without delay.
- **Amplify Learning –** Use frequent iterations and regular releases to provide feedback.
- **Delay Commitment** – Make decisions at the last responsible moment.
- **Deliver Fast –** The measure of the maturity of an organization is the speed at which it can repeatedly and reliably respond to customer need.
- **Empower the Team –** Assemble an expert workforce, provide technical leadership and delegate the responsibility to the workers.
- **Build Integrity In** – Have the disciplines in place to assure that a system will delight customers both upon initial delivery and over the long term.
- **See the Whole** – Use measurements and incentives focused on achieving the overall goal.

Gustaf: OK. You say I need to eliminate waste, but how do I recognize it? Where should I start looking?
Mary: Something like 45% of the features in a typical software system are never used, another 19% are rarely used. That means two thirds of the software in a typical system is waste. Eliminating this waste is the first place to look for reducing software waste. We need to focus on creating more value with less effort, and make sure that the resulting systems do not turn into legacy software.

Gustaf: If what you are saying is true, all that needless functionality is certainly a waste. How come so many features are never or rarely used?

Mary: Quite often at the beginning of a software development project, we ask customers what they want, even though they don’t really know. We make it clear to customers that they need to tell us about everything they might possibly want, and quite often we record their wish list without question. This is what we call ‘Scope’. Later, if customers want to add or change items in the ‘Scope’ we challenge them with a ‘Change Review Process’. So we reward customers for coming up with a long initial list of features, and punish them if they want to modify the list at a later stage. Is it any wonder that two thirds of the features in a system developed playing this game are rarely or never used?

Gustaf: So, I realize putting in extra features is a big source of waste. In your book, you write that another way of discovering waste is mapping your value stream. What is a value stream?

Mary: In the physical product world, a value stream is the flow of a product from raw material to final use. For instance, a Cola can starts out as bauxite, is reduced to alumina, smelted into aluminum ingots, rolled into sheets, rolled again into thin strips, stamped into disks, formed into cans, cleaned and painted, filled with Cola and sealed, packaged and put on a pallet, warehoused at a distributor, sent to a retail store, put on a shelf, purchased by a consumer, stored in a refrigerator, and finally the Cola is consumed. At every step, a huge pile of inventory builds up waiting for the next step, which is days or weeks away. This value stream takes 319 days, of which only 3 hours (less than .04%) are spent actually making the product.

In the software world, a value stream starts when a business unit decides that better information would help it increase revenue or reduce cost. This is the start of the value stream. The value stream ends when the deployed software starts generating the extra revenue or reducing costs. Inventory in the software development value stream is partially done work: requirements that are not analyzed and designed, designs that are not coded, code that is not tested and integrated, features that are not deployed, and deployed features that are not saving money or reducing costs. When the software value stream has as little of this partially done work as possible, risks are reduced and productivity is greatly improved.

Gustaf: Another principle is delaying commitment. Isn't the project schedule in danger of slipping when no one dares make a decision?

Mary: A military officer who was about to retire once said: ‘The most important thing I did in my career was to teach young leaders that whenever they saw a threat, their first job was to determine the timebox for their response. Their second job was to hold off making a decision until the end of the timebox, so that they could make it based on the best possible data.’

Our natural tendency is to make decisions and get them over with. However, it is far better to determine the timebox for every decision, and then make the decision at the end of the timebox, because then we can make decisions based on the best possible data. In Lean Software Development, decisions are not avoided; they are scheduled and made at the last responsible moment. This assures that all decisions are made in a timely manner, yet they are made with as much information as possible to help make the best decision possible.

Gustaf: What do you mean by 'integrity'? Why is it so important to maintain it?

Mary: Integrity is a level above quality; it is the thing that makes people want to use a product. For instance, there are many high quality search engines, but only one Google. There is something about Google that attracts people to use it every day. That is integrity. Software is never an end in itself, it is always a means to an end. It gets used when it provides the best means to the end. To maintain integrity, software needs regular updates to make sure that it continues to provide the best way for users to achieve their goals, even as the goals and the technology changes. Otherwise the software becomes irrelevant at best, or legacy at worst.

Gustaf: Why have not all organizations been successful in applying Lean Thinking? What are the most frequent pitfalls?

Mary: At its core, Lean Thinking means creating a learning environment for workers in order to increase the flow of value. All too often, practices from successful Lean companies are adopted, but the heart of Lean Thinking is lost. If people, learning and value are not the central focus of a Lean initiative, it will not be particularly successful.

Last week I explained to a president of a small software company how to implement responsibility-based planning and control: schedule releases and iterations, make sure that the development team agrees at the start on what features will be included, and then leave it to the team to meet their commitment. This made the president of the company uneasy. He was already using iterations, but he wasn't quite ready to give up the responsibility for planning and tracking tasks to the development team.

Think of an emergency response team, firefighters or paramedics for example. They are trained in emergency scenarios that establish patterns for responding to anything they are likely to encounter. When an emergency occurs, there is no time for decisions to go up the chain of command and back down; emergency responders are expected to use their own judgment to deal with an emergency as they confront it. Similarly in a Lean organization, everyone is trained and equipped to do their job, and the organization is structured so that it is clear what needs to be accomplished. But it is the workers who make decisions about what to do, track their own work, and assume the responsibility for meeting their goals.

Gustaf: This sounds a lot like Scrum to me. In Scrum, the team is responsible for managing itself. How are Agile methodologies such as Scrum and Extreme Programming related to Lean Software Development?

Mary: Both Scrum and Extreme Programming (XP) are examples of Lean Thinking applied to developing software. Scrum is an excellent approach to responsibility-based planning and control. XP is a tremendous set of disciplines that enable rapid, repeatable, reliable delivery of code. I particularly like XP's focus on testing, continuous integration and refactoring. I think of refactoring as constantly improving the code base – it's sort of like applying kaizen (the Japanese word for continuous improvement) to a software system.

Gustaf: Where can I learn more about Lean Software Development?

Mary: A good start would be my book: Lean Software Development: An Agile Toolkit. Other recommended reading is Lean Thinking: Banish Waste and Create Wealth in Your Corporation, 2nd edition (by Jones & Womack), Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency (DeMarco) and Product Development for the Lean Enterprise (Kennedy). Also,check my own website and articles on Lean Development at Agile Alliance.