



Lean Software Development Cutting Fat Out of Your Diet

After revolutionizing the manufacturing sector, lean principles are now poised to do the same in the software industry. Traditional waterfall development is giving way to more effective approaches that focus on empowering employees, reducing waste, and streamlining processes. This white paper gives an overview of lean software development and explains how Agile methods and test-driven development support lean objectives.

Introduction

Business leaders who shop around for a software development firm are often exposed to new concepts like “Agile” and “test-driven development” that can leave them wondering if there’s a method to this apparent madness. For example, the notion of implementing a project without first developing an approved business requirements document is shocking to people with a background in engineering or architecture. However, building software is not like building a bridge.

Smart developers understand that software is, in fact, unlike any other product and its development process must necessarily be different.

While most business leaders are not familiar with the ins and outs of software development, they may have some grasp of lean principles and how lean has been applied to manufacturing, product development, and other activities to achieve overwhelmingly positive results.¹

Most famously, lean principles have revolutionized the automotive industry. Toyota was the first to develop lean principles and implement them across its organization, resulting in a huge competitive advantage that the automaker parlayed into record sales and an enviable reputation for quality (until recently). Lean is now the rule, not the exception, for car manufacturers and their suppliers.

Computer manufacturer Dell is another notable lean success story. Dell became a dominant player in the PC market by using lean principles to radically streamline its supply chain and manufacturing processes.^{3, 4}

More recently, software development has also benefited from lean thinking, though the methods for implementing these powerful ideas are usually disguised under other names—for instance, “Agile” and “test-driven development”.

The Shift toward Lean Principles

Like many industries, the software industry has struggled to overcome a legacy of scientific management practices that are firmly entrenched in business thinking. Scientific management espouses an artificial separation of thinking (managers only) and doing (workers only) that on the surface seems logical but in practice makes a business less competitive because the real experts (the workers) have no say in process improvement.

Just as lean manufacturing—with its emphasis on eliminating waste and empowering employees—shook up the automotive industry, lean principles are revolutionizing software development. Lean developers can build software faster, better, and cheaper than competitors using traditional methods. By adopting Agile practices and test-driven development, a software firm can go a long way toward leaning out its operations and serving its customers better.

This white paper explains how lean principles can be applied to software development and describes where Agile practices and test-driven development fit into this strategy.

But why should we care if world manufacturers jettison decades of mass production to embrace lean production? Because the adoption of lean production, as it inevitably spreads beyond the auto industry, will change everything in almost every industry...

- Womack, Jones & Roos
The Machine that Changed the World²

There was a connection between lean manufacturing and agile software from the beginning in that many of the developers of the various agile methods were influenced by the ideas of lean manufacturing.

- Martin Fowler
Agile Versus Lean⁵

The idea of lean production is to expose problems as soon as they arise, so they can be corrected immediately. It may seem that lean systems are fragile, because they have no padding. But in fact, lean systems are quite robust, because they don't hide unknown, lurking problems and they don't pretend they can forecast the future.

- Mary Poppendieck
Principles of Lean Thinking⁶



Lean Software Development

As the name suggests, lean is all about doing more with less. It relies on *pull*—as in pulling products with customer orders—rather than *push*—pushing out products, stockpiling them, and then trying to get orders. Regardless of whether the products are widgets or software, the principle is the same: produce just enough to satisfy demand but no more.

A lean organization tries to use every resource to its fullest potential, wasting nothing and adding value in every process step. The key to success here is people. By empowering employees, lean companies put decision making in the hands of the experts, which improves performance and increases competitiveness.

Naturally, when lean is applied to a particular environment—for example, software development—its principles must be adapted to achieve the desired outcome.

Seven Principles of Lean Software Development

Seven key principles have been defined for lean software development, which are:⁷

1. Eliminate waste

In software development, waste can be defined as effort that either adds nothing to the final product or actually impairs it in some way. For example, the traditional approach of formulating a detailed business requirements document up front is wasteful because some of the requirements are initially unknown, some will inevitably change, and others will simply drop off the list. By following this type of design document, a development team will certainly create many unnecessary features and may irreversibly commit to the wrong software architecture.

Table: Seven Types of Waste⁹

Manufacturing	Software Development
Defects	Defects
Overproduction	Unnecessary features
Transportation	Handing off work
Waiting	Waiting
Inventory	Incomplete work
Motion	Information gathering
Processing (extra steps)	Processing (extra steps)

A more efficient alternative is to adopt the Agile practice of capturing each high-priority feature in a short *user story*, a concise description of a complete, purposeful user interaction with the software. The development team

...it can be safely said that Lean views all Agile methodologies as valid supporting practices. The primary focus of Agile software development is on close customer collaboration and the rapid delivery of working software as early as possible. Lean sees that as worthwhile, but its primary focus is on the elimination of waste in the context of what the customer values.

- Hibbs, Jewett & Sullivan
The Art of Lean Software Development⁸

Partially done software has all of the evils of manufacturing inventory: It gets lost, grows obsolete, hides quality problems, and ties up money. Moreover, much of the risk of software development lies in partially done work... But far and away the biggest source of waste in software development is extra features. Only about 20 percent of the features and functions in typical custom software are used regularly.

- Mary & Tom Poppendieck
Implementing Lean Software Development¹⁰



immediately develops code to satisfy the user stories during a programming sprint that typically lasts two to three weeks. With this approach, the team always uses the latest information to rapidly develop complete units of working software and deliver them to the customer.

To learn more about Agile development, read our white paper [“Agile Software Development: A Smart Choice for Outsourced Projects”](#).

Finding and fixing software defects can also be a large source of wasted effort, so the goal of any development team should be to prevent defects from occurring at all. The best way to accomplish this goal is to use test-driven development (TDD), a process that engineers out most defects at the source.

With TDD, a programmer writes a test and then writes just enough code to satisfy the test. When each chunk of code passes its corresponding test, the programmer then refactors (optimizes) the code to improve performance. The result is that each programmer not only writes clean code but also contributes new tests to a built-in test suite that monitors the “health” of the software throughout its lifecycle, detecting any defects that are introduced in subsequent changes to the code.

To learn more about test-driven development, read our white paper [“Test-Driven Development: A Commitment to Quality”](#).

2. Learn your way to the best solution

A software project doesn't start with *We need xyz*. It actually starts with *We need to do abc: How can we accomplish this goal?*

In other words, a software project shouldn't begin with a theoretical solution (i.e. a comprehensive list of requirements). Instead, it should begin with a simple problem statement and the project team should develop a concrete solution—working software—incrementally.

Whether or not the solution is the right one depends on the customer's level of involvement in the development process. A development firm that frequently (i.e. daily) solicits feedback from its customers—learns from its customers—is more likely to deliver great software that works.

Agile development, with its short programming sprints and rapid software delivery, makes it possible for customers to learn what works for them (and what doesn't), and then guide the development team to the right solution. After each sprint, the development team reviews what they've learned—from their own experience and through customer feedback—and incorporates this knowledge into their next effort. Together, customers and developers *learn* their way to the best solution.

And finally, by recording knowledge from each project in a way that it can be easily retrieved, the development team is better equipped to tackle subsequent projects and advise its customers.

In Lean Software Development, the goal is to eliminate as many documents and handoffs as possible. The emphasis is to pair a skilled development team with a skilled customer team and give them the responsibility and authority to develop the system in small, rapid increments, driven by customer priority and feedback.

- Mary Poppendieck
Principles of Lean Thinking¹¹

One of the puzzling aspects of “waterfall” development is the idea that knowledge, in the form of “requirements,” exists prior to and separate from coding. Software development is a knowledge-creating process... An early design cannot fully anticipate the complexity encountered during implementation, nor can it take into account the ongoing feedback that comes from actually building the software.

- Mary & Tom Poppendieck
Implementing Lean Software Development¹²



3. Delay commitment

This principle is a cornerstone of iterative development approaches like Agile, and a common source of misunderstanding. Delaying commitment doesn't mean avoiding decisions. Rather, it means waiting "until the last responsible moment" to make *irreversible* decisions.¹³ Delaying commitment actually encourages good decision making by suggesting that developers wait until complete information is available before choosing one option over another.

Practically, this principle is about doing work that's being pulled by customer demand and tabling everything else. Developers who leap ahead and try to anticipate demand are taking unnecessary risks. For example, if the team locks into a specific architecture too soon, the software might be unable to accommodate future changes that would otherwise be easy to make.

With TDD, the development team delays committing to specific portions of the code until just before writing the tests, ensuring that tests and corresponding code chunks are based on the latest information. Also, by producing small units of simple, clean code, developers build flexibility into the software, allowing them to keep their options open as long as possible. When a development team writes complex code or code that contains many defects, they make the software harder to change, which guarantees that a major testing and debugging effort will be required downstream. In this scenario, maintenance will likely be more problematic and the effective life of the software may be significantly decreased.

A development team should be mindful to keep each option open for as long as *reasonably* possible.

4. Deliver quickly

Being lean means being fast. While it's good practice to delay decisions as long as reasonably possible, implementation should be fast as lightning when a commitment is finally made.

Using Agile, a development team can break up an application into bite-sized chunks and deliver new units of working software to a customer every few weeks. Although test-driven development takes a little longer to initially execute, the benefits of producing the right code up front (and defect-free) are cumulative and TDD actually helps to speed up the overall development process. Conversely, the negative effect of defects is cumulative—this is called *technical debt*—and it slows down development.

5. Have respect for people

This is the linchpin. Without respect for people, the other six principles never amount to anything more than words because employees, not management, are the real agents of lean development.

Agile is also a people-centric development philosophy. When the authors of the Agile Manifesto stated that they valued "individuals and interactions over processes and tools"¹⁶, they obviously knew that empowered employees improve processes rather than simply follow them, and choose the best tools rather than use what they're given.

Caution: Don't equate high speed with hacking. They are worlds apart. A fast-moving development team must have excellent reflexes and a disciplined, stop-the-line culture. The reason for this is clear: You can't sustain high speed unless you build quality in.

- Mary & Tom Poppendieck
Implementing Lean Software Development¹⁴

Traditional software development has followed the same pattern as traditional U.S. car manufacturing: let defects slip through and get caught later by QA inspections. The Lean approach is to mistake-proof your code by writing tests as you code the features.

- Hibbs, Jewett & Sullivan
The Art of Lean Software Development¹⁵



At first glance, this principle might seem like a no-brainer but when you think about it, how many companies actually set goals for employees, give them adequate resources, and then get out of their way?

6. Build quality in

The traditional practice of building a product and then checking it for defects is not good enough. Following such a strict separation of functions is a lazy way to develop software. Even worse, it encourages sloppy coding by setting up the expectation that defects will be found and fixed during the testing phase (after coding is complete), when changes are more difficult and expensive to execute.

The alternative is to incorporate key aspects of quality assurance into the coding process and eliminate defects immediately after they're created. This can be accomplished with test-driven development. A development team that uses TDD prevents technical debt from piling up and overwhelming the project.

7. Optimize the top level

According to one estimate, 80% of software defects can be attributed to the system that a team follows rather than to errors made by individual programmers.¹⁸ Therefore, it makes sense to evaluate team performance, not individual performance, and try to improve the system.

Also, as tempting as it may be to individually optimize each step in the software development process, this can be counter-productive if the process as a whole becomes less effective. It's important to stay focused on the overall goal of lean software development: meeting customer needs as efficiently as possible.

Role of Architecture in Lean Development

At first glance, software architecture might seem to be at odds with lean principles like “learn your way to the best solution” and “delay commitment” but in fact, just the opposite is true.

The misconception arises because people confuse principles of *building architecture* with those of *software architecture*. Building architects draw up detailed blueprints at the beginning of a project and expect builders to follow their designs precisely. It's a prescriptive process. Software architecture, however, is more empirical.

Smart developers never design an entire software architecture in advance because they know that business requirements will surely change during the life of the project. Instead, they let the architecture evolve organically—though not accidentally—as development progresses. Guided by an experienced software architect, the developers learn their way to the best architecture and delay decisions until the last responsible moment. The lean way actually makes good architectural sense.

To learn more about software architecture, read our white paper “[The Importance of Software Architecture](#)”.

Following the adoption of TDD the number of defects reported dropped to less than three per thousand lines of code. Even this 70 percent reduction in defects understates the improvement...The team was over three times more productive with far fewer defects.

- Mike Cohn, President Mountain Goat Software¹⁷

Agility is everywhere. Ignore it and you will be lost in the current technical dialogue. Learn about it and you will be able to make intelligent decisions about whether it is right for you. You will also be able to understand many of the other practices and methodologies that are emerging. If you are a manager of technology at any level, this is your responsibility as well as a survival necessity.

- Gary Pollice Agile Software Development¹⁹



Success Stories

Here's what a few software companies have achieved by applying lean principles to their development process:

- **Salesforce.com** has improved time to market of major software releases by 61% and boosted productivity across their R&D organization by 38% since adopting Agile development.²⁰
- **BT Adastral**, the largest telecommunications company in the UK, completed its first major lean software project 50% sooner than expected and incorporated many product changes along the way. The product yielded 80% ROI in the first year.²¹
- **PatientKeeper**, specializing in software for the healthcare industry, puts out weekly maintenance releases, monthly new feature releases, and quarterly new application releases. This company completes 45 development cycles in the time it takes their competitors to do 1 cycle.²²
- **Timberline Software** (now part of The Sage Group), serving the construction and real estate market, estimates that improvements in quality, costs, and time to market were all greater than 25% as a result of switching to lean software development.²³

...Lean techniques (which have dramatically increased manufacturing success for over 50 years) are now being applied to software development and validating the successes of Agile. The Lean principles and the mindset of Lean thinking have proved remarkably applicable to improving the productivity and quality of just about any endeavour...However, only in the last few years have the Lean principles and techniques been applied to software development.

- Hibbs, Jewett & Sullivan
The Art of Lean Software
Development²⁴



References

1. Daft, Richard L. *Organization Theory and Design, 9th Edition*, (Mason, Ohio: Thomson Higher Education, 2007), 255-57.
2. Womack, James P., Daniel T. Jones, and Daniel Roos. *The Machine that Changed the World*, (New York: Free Press, 2007), 10.
3. Ireland, R. Duane, Robert E. Hoskisson, and Michael A. Hitt. *Understanding Business Strategy: Concepts and Cases, Second Edition*, (Mason, Ohio: South-Western Cengage Learning, 2009), C-51.
4. Pritchard, Stephen. "Inside Dell's Lean Machine", allbusiness.com, December 1, 2002, <http://www.allbusiness.com/management/960195-1.html>, accessed June 22, 2010.
5. Fowler, Martin. "Agile Versus Lean", martinowler.com, June 26, 2008, <http://martinowler.com/bliki/AgileVersusLean.html>, accessed May 25, 2010.
6. Poppendieck, Mary. "Principles of Lean Thinking", Poppendieck LLC, 2002.
7. Poppendieck, Mary and Tom. *Implementing Lean Software Development: From Concept to Cash*, (Boston: Addison-Wesley, 2006), 23.
8. Hibbs, Curt, Steve Jewett, and Mike Sullivan. *The Art of Lean Software Development*, (Sebastopol, California: O'Reilly Media, 2009), 23.
9. Poppendieck, "Principles of Lean Thinking".
10. Poppendieck, *Implementing Lean Software Development*, 24.
11. Poppendieck, "Principles of Lean Thinking".
12. Poppendieck, *Implementing Lean Software Development*, 29-30.
13. Poppendieck, *Implementing Lean Software Development*, 32.
14. Poppendieck, *Implementing Lean Software Development*, 35.
15. Hibbs, *The Art of Lean Software Development*, 20.
16. "Manifesto for Agile Software Development", <http://agilemanifesto.org/>, accessed May 6, 2010.
17. Poppendieck, *Implementing Lean Software Development*, 28.
18. Poppendieck, Mary. "Lean Software Development", C++ Magazine, Methodology Issue, Fall, 2003.
19. Pollice, Gary. "Agile Software Development: A Tour of its Origins and Authors", ibm.com, March 15, 2007, <http://www.ibm.com/developerworks/rational/library/mar07/pollice/>, accessed May 25, 2010.
20. "Agile Development Meets Cloud Computing for Extraordinary Results at Salesforce.com", Salesforce.com, 2008.



21. Grant, Ian. "BT Adastral Moves to Lean Software Development", computerweekly.com, April 20, 2010, <http://www.computerweekly.com/Articles/2010/04/20/240971/BT-Adastral-moves-to-lean-software-development.htm>, accessed June 22, 2010.
22. Poppendieck, Mary. "A History of Lean: From Manufacturing to Software Development", slide 22, Poppendieck LLC, September, 2005.
23. *The Lean Office: Collected Practices and Cases*, (New York: Productivity Press, 2005), 15.
24. Hibbs, *The Art of Lean Software Development*, 2.



Architech Solutions
3 Church Street
Suite 602
Toronto, Ontario
M5E 1M2

Phone: (416) 607-5618
Fax: (416) 352-1768

Architech Solutions is a Toronto-based technology consulting and software development firm. We design and build powerful, user-centred systems that work. We're agile, disciplined, and passionate about delivering for our clients.

To book a free on-site Discovery Workshop led by our team of consultants, e-mail info@architech.ca or visit us at www.architech.ca

