Lean Software Development

Dr. Khubaib Amjad Alam
Tahir Farooq

Lean

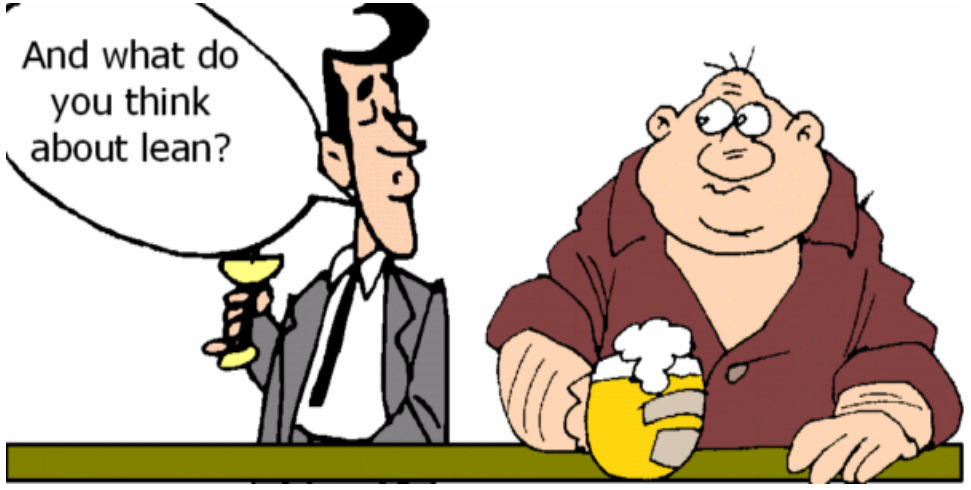
- Pioneered by Toyota over the last 50 years.
  - Enabled sustained competitive advantage.
    - #2 Car maker
    - #1 in Profitability
- Adapted for Software
  - Lockheed Martin
  - Timberline Software
  - ProWorks

Origins of Lean Software Development

- Original author : Taiichi Ohno
  - Inventor of Just-In-Time manufacturing
- “Costs do not exist to be calculated. Costs exist to be reduced.”
  - Taiichi Ohno

Lean software development

- **Lean software development** is a translation of Lean manufacturing and Lean IT principles and practices to the software development domain. Adapted from the **Toyota Production System.**
- Lean is an Agile methodology which can also be seen as a philosophy
- The core idea is to maximize **customer value** while minimizing waste. Simply, lean means creating more value for customers with fewer resources.



What is lean development?

- Implementation of lean manufacturing principles into a software development model

- **Goal :** Reduce the waste in a system and produce a higher value for the final customer

What is lean development?

- Similarities to Agile Development
- Agile Manifesto refresher: Developers shall value
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over customer negotiation
  - Responding to change over following a plan
- Based on, yet a basis for Agile projects

Basics of Lean Development

- **Muda** (Japanese terminology for waste)
- Any activity that uses of resources, but adds no value
- Examples of waste
  - Partially done work
  - Extra processes
  - Defects

Basics of Lean Development

- **Value**
- What the customer wants the product to do
- 45% of all software features go unused
- What do customers really need?

Basics of Lean Development

- **Value Stream**

- All actions required to bring project from creation to completion

- Types of actions
  - Add value
  - No value added, but unavoidable
  - No value added, avoidable

Basics of Lean Development

- **Flow:** Product is in motion at all time
- **Pull:** No product is made until the customer requests it
- **Pursuit of perfection:** After a project flows… keep improving it

Lean principles

- Eliminating Waste
  eliminate anything that does not add value to the customer

- Amplify Learning
  This principle encourages Lean teams to provide the infrastructure to properly document and retain valuable learning

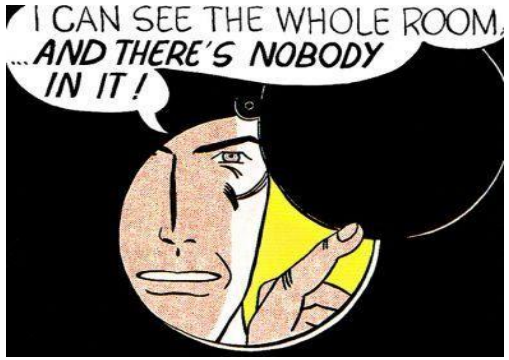
- Decide as late as possible, Defer commitment
  encourages team to demonstrate responsibility by keeping their options open and continuously collecting information, rather than making decisions without the necessary data.

- Deliver as fast as possible
  The Lean way of delivering quickly isn't working longer hours and weekends, or working recklessly for the sake of speed. Lean development is based on this concept: Build a simple solution, put it in front of customers, enhance incrementally based on customer feedback.

Lean principles

- Empower the Team

- See/optimize the Whole
  encourages Lean organizations to eliminate these sorts of vicious cycles by operating with a better understanding of capacity and the downstream impact of work.
- Two vicious cycles:
  The first is releasing sloppy code for the sake of speed.
  When testers are overloaded, it creates a long cycle time between when developers write code and when testers are able to give feedback on it.



Principle 1: Eliminate Waste

- Remove all wastes that add no value to project
  - Waste in code development
  - Waste in project management
  - Waste in workforce potential

Eliminate waste

- ***Waste is anything that does not add customer value***
- Customers wouldn’t choose to pay for it.
- Waste is anything that has been started but is not being used in production.
- Waste is anything that delays development or keeps people waiting.
- Waste is any extra features that are not needed now.
- Waste is making the wrong thing or making the thing wrong.

- Waste is an intrinsic part of a system, some types of waste are a result of beneficial aspects of your process.
- Handoffs (handing over work to next person) in software development generally cause waste.

Seven types of waste in software development

- Work-in-progress
- Over-engineering
- Hands off
- Task switching
- Delays
- Relearning the process
- Defects

Work-in-progress

- Most common form of waste: Work which has been partially or fully completed but not yet released to the client.

Over-engineering

- Doing more than was required to achieve a certain outcome.
- You spend so much time to address a problem which was never actually a problem.
- E.g. Over-emphasis on validation checks.

Task switching

- The battle of responsiveness vs efficiency
- Swapping from one task to another, then having to swap back to original work.
- E.g sudden issues raised by clients, and developers are forced to put-aside their current work and resolve that query first.

Delays

- Waiting between the end of one process and another.
- Indirect waste, with an impact on elapsed time.
- E.g If u have a shared tester working between variety of projects, and you have a developer who finished his taks, and task is ready for testing but shared tester is not available. Now there is a delay..

Relearning the process

- Needing to get back up to speed on a piece of system which has not been looked at for a while.
- Similar to task switching
- If you leave working on a particular technology or feature for some weeks, switching back to that technology or feature will require some relearning.

Defects

- Issues in the code itself:
- Process not being followed
- Specifications misunderstood or not being followed
- Insufficient or incorrect specifications
- Indirect effect of new code
- Human errors

- You may spend huge time and budget resources to deal with the defects

Principle 2: Amplify Learning

- Learning is not compulsory... neither is survival.
- Full knowledge is impossible, need to learn as project is developed
- The customer should be learning too!

Principle 2: Amplify Learning

- Use the scientific method
  - Create hypothesis
  - Conduct rapid tests of possible solutions
  - Compare results
  - Implement the best solution

Principle 3: Decide as late as possible, Delay Commitment

- Decisions are often made on incomplete or incorrect information
- Delay until the last responsible moment
- Benefits:
  - More knowledge for decision
  - Leave options open

- **Pilots**

"In Pilot training, we learned that when we had to make a decision, we should first decide when the decision should be made, then when the time comes, make the decision based on the available information."

- **Military**
- "One of the most important thing I taught young recruits is that when they were threatened, they should decide on the timebox for a response, and not respond until the end of the timebox."

Principle 4: Empower the Team

- Make decisions as “low” as possible
- Shift from complete documentation to goals and guidelines
  - Developers closest to code know the problem best
  - Developers can get feedback for personal improvement
  - Developers must be experienced in domain
- “The best executive is one who has sense enough to pick good people to do what he wants done, and self-restraint enough to keep from meddling with them while they do it.”
  - Theodore Roosevelt

Principle 5: Deliver Fast

- Customers change their mind.
  - Solution: Deliver so fast, they don’t have time to!
- Short release cycles
  - About two weeks
- Requirements:
  - Don’t overload developers with requests
  - No partially completed work

Principle 5: Deliver Fast

- Doesn’t fast delivery lead to hasty, buggy code?
  - Don’t confuse speed with rushing to get something done on time!
  - Speed vs Schedule
- Flowing processes
  - Development
  - Knowledge gathering

Rapid Delivery is the Competitive Advantage of:

Manufacturing

Order Processing

Shipping

Supply Chain

Aircraft Turnaround

Product Development

Principle 6: Build Integrity In

- Perceived Integrity
  - Does exactly what the customer wants, even if they don’t know it yet
  - Again, continual interaction with the customer
- Conceptual Integrity
  - The product works smoothly and functions well

Principle 7: See the whole

- Optimize the entire product, not just parts
- Short delivery cycles can lead to optimized parts, sub optimized whole
- Don’t worry about scope!
  - By focusing on what customers want, scope will handle itself
  - Don’t have customer overestimate what the product will do

Vicious Cycle #1:

1. A customer wants some new features yesterday
2. Developers hear: Get it done fast, at all costs!
3. Result: Sloppy changes are made to the code base
4. Result: Complexity of code base increases
5. Result: Number of defects in code base increases
6. Result: Exponential increase in time to add features

Vicious Cycle #2:

1. Testing is overloaded with work
2. Result: Testing occurs long after coding
3. Result: Developers don't get immediate feedback
4. Result: Developers create more defects
5. Result: Testing has more work. Systems have more defects.
6. Result: Feedback to developers if delayed further. Repeat cycle.

Recap

- Lean development is focused on removing waste from a system and improving value for the customer
- Principles of Lean software development
  - Eliminate Waste
  - Amplify Learning
  - Delay Commitment
  - Empower the Team
  - Deliver Fast
  - Build Integrity in
  - See the whole