**Lean Method**

Introduce the Topic

* Lean development originated from lean manufacturing and Toyota production in Japan
  + Also known as the “just-in-time” manufacturing
  + Lean manufacturing is a process management philosophy that transformed the car manufacturer’s approach to building vehicles.
  + Expand here?
* Lean Software Development: An Agile Toolkit
  + 2003 book by Mary Poppendieck and Tom Poppendieck
  + Presents the lean principles and compares then to traditional agile tools
* What is it?
  + Principles
    - 1. Eliminate Waste
      * 3 general forms of waste identified by the Toyota Production System
        + Muda – meaning unproductive
        + Mura – unevenness, inconsistency
        + Muri – over burden, unreasonableness
      * 7 types of waste

1. Over production
2. Unnecessary transportation
3. Inventory
4. Motion
5. Defects
6. Over processing
7. Waiting
   * + - Everything that does not add value to the customer is waste
         * Mary and Tom Poppendieck related this as:

Unnecessary code or functionality

Starting more than can be completed

Delay in the software development process

Unclear or constantly changing requirements

Bureaucracy

Slow or ineffective communication

Partially done work

Defects and quality issues

Task switching

Avoidable process repetition (often caused by insufficient testing)

* + - * The iterative process of constantly learning found in agile is key for constantly removing waste.
      * Making improvements “little but often” creates a culture that is constantly improving and an overall learning environment.
      * In order to eliminate waste, one should be able to recognize it
        + If some activity could be bypassed or the result could be achieved without it, it is waste
        + Partially done coding eventually abandoned during the development process is waste
        + Extra processes and features not often used by customers are waste
        + Waiting for other activities, teams, processes is waste
        + Defects and lower quality are waste
        + Managerial overhead not producing real value is waste
      * A value stream mapping technique is used to identify waste
      * Second step is to point out sources of waste and eliminate them
      * Waste removal should take place iteratively until even seemingly essential processes and procedures are liquidated
    - 2. Amplify Learning
      * This is the best approach for improving a software development environment
      * Accumulation of defects should be prevented by running tests as soon as the code is written
      * Instead of adding more documentation or detailed planning, different ideas could be tried by writing code and building
      * Process of user requirements gathering could be simplified by presenting screens to the end users and getting their input
      * Learning process is sped up by usage of short iteration cycles – each one coupled with refactoring and integration testing
      * Increasing feedback via short feedback session with customers helps with determining the current phase of development and adjusting efforts for future improvements
      * During these short sessions both customer representatives and the development team learn more about the domain problem and figure out possible solutions for further development
        + Thus, the customer better understand their needs, based on the existing result of development efforts, and the developers lean how to better satisfy those needs
      * Another idea in communication and learning process with a customer is set-based development
        + Concentrates on communicating the constraints of the future solution and not the possible solutions

This promotes the birth of the solution via dialogue with the customer

* + - 3. Decide as late as possible
      * Better results should be achieved with an options based approach, delaying decisions as much as possible until they can be made based on facts and not on uncertain assumptions and predictions
      * The more complex a system is, the more capacity for change should be built into it
        + Enable the delay of important and crucial commitments
      * Iterative approach promotes this principle
        + The ability to adapt to changes and correct mistakes, which might be very costly if discovered after the release of the system
      * Agile software development approach can move the building of options earlier for customers
        + Delay certain crucial decisions until customers have realized their needs better
        + Also allows later adaptation to changes and the prevention of costly earlier technology bounded decisions
      * This does not mean no planning should be involved
        + Planning activities should be concentrated on the different options and adapting to the current situation as well as clarifying confusing situations by establishing patterns for rapid action
      * Evaluating different options is effective as soon as it is realized that they are not free, but provide the needed flexibility for late decision making
    - 4. Deliver as fast as possible
      * The sooner the end product is delivered without major defects, the sooner feedback can be received, and incorporated into the next iteration
      * The shorter the iterations, the better the learning and communication within the team
      * With speed, decisions can be delayed
        + Speed assures the fulfilling of the customer’s present needs and not what they required yesterday
        + Gives them the opportunity to delay making up their minds about what they really require until they gain better knowledge
      * Customers value rapid delivery of a quality product
    - 5. Empower the team
      * Managers are taught how to listen to the developers, so they can explain better what actions might be taken, as well as provide suggestions for improvements
      * Favors the aphorism “find good people and let them do their own job”, encourage progress, catching errors, and removing impediments, but not micro-managing
      * People need motivation and a higher purpose to work for
      * Developers should be given access to the customer
      * Team leader should provide support and help in difficult situations as well as ensure that skepticism does not ruin the team’s spirit
    - 6. Build quality in
      * Pair Programming
        + Seeks to avoid quality issues by applying the minds of two developers to each task
      * Benefits from the collective, combined experience of two developers instead of one
        + Often resulting in better productivity as they see solutions that on their own they might not have done
      * Improved quality, since one person can be thinking slightly ahead of the other, catching issues before they occur.
      * Test Driven Development
        + Writing tests before writing code
        + Stubbing out the code and writing automated unit tests for each of the test conditions before actually writing the code. The developer then writes the code to pass the tests.
        + Prevent quality issues from happening
      * Constant Feedback
        + By doing development in small incremental steps, through close collaboration, and by developing in small iterations
        + These agile methods provide the opportunity for constant 2-way feedback between the Product Owner and the team.
        + This feedback can valuable to ensure the right level of quality the right product.
      * Minimize time between Stages
        + Building quality into the development process is to minimize the time between development, testing and bug fixing
        + Don’t log bugs – deal with them immediately
      * Frequent Iterations
        + Continuous integration, with code integrated into the overall system, built and automatically unit tested as soon as it is checked in
        + Minimizing this gap also reduces integration waste

Regular builds and frequent iterations avoid lengthy iterations and regression testing phases.

* + - * Automation
        + Agile development methods also encourage automated regression testing.

Not just a Lean Principle but it’s a crucial aspect

* + - 7. Optimize the whole
      * Mary and Tom Poppendieck give 2 examples of the software engineering tendency to sub-optimize:
        + Vicious circle number 1

A customer wants some new features ‘yesterday’.

Developers hear: get it done fast, at all costs!

Result: sloppy changes are made to the code.

Complexity of the code base increases.

Number of defects in the code increases.

* + - * + Vicious circle number 2

Testing is overloaded with work.

Result: testing occurs a long time after the code is originally written, or testing is reduced.

Developers don’t get immediate feedback, or some things are not properly tested.

There are more defects in the code.

Testers have more work.

Feedback to developers and quality improvements are delayed further.

* + - * Lean seeks to optimize the whole value stream, not just individual functions or teams
      * The best way to organize teams is so they are complete, multi-disciplined, co-located product teams that have all the roles and skills they need to deliver a request from start to finish, without reference to other teams.
      * Putting all of this together with the better optimized workflow, the benefits or organizing in this way can be extremely significant – not only in terms of the team’s performance, but also in terms of the quality of the product, which ultimately can make your organization more competitive.
* History
  + Lean is a manufacturing & production practice that considers the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful, and thus a target for elimination
  + “value” is defined as any action or process that a customer would be willing to pay for
  + lean is centered around preserving value with less work
  + lean manufacturing is based on optimizing flow, increasing efficiency, decreasing waste, and using empirical methods to decide what matters rather than uncritically accepting pre-existing ideas
  + Toyota was a leader in implementing lean practices in the 80s

Compare to Japanese manufacturing style

* Lean manufacturing was created by Taiichi Ohno (1912-1990)
  + Father of Toyota manufacturing
    - Became known as Lean manufacturing in the United States
    - Identified 7 wastes
      * 1. Delay, waiting or time spent in a queue with no value being added  
        2. Producing more than you need  
        3. Over processing or undertaking non-value added activity  
        4. Transportation  
        5. Unnecessary movement or motion  
        6. Inventory  
        7. Production of Defects
* Wanted to eliminate waste from manufacturing:
  + Materials left over on the shop floor
  + Inventory on-hand that isn’t needed yet
  + Improperly made items that are hanging around, waiting to be fixed
  + The amount of work that is taking place

Present details about it (including code and non code based examples)

* Definition
  + Lean is a customer-centric methodology used to continuously improve any process through the elimination of waste in everything you to
  + It’s based on the ideas of “Continuous Incremental Improvement” and “Respect for People”
* Lean attacks waste mainly by shortening the time between the customer order and shipment
  + Customer is prepared to pay for operations by producers or their suppliers that transform the product in a way that is meaningful to the customer
  + Customers don’t want to pay for waste at the producer end
* 5 steps thought process for guiding the implementation of lean techniques

1. Identify Value

* Define **value** from the customers perspective and express value in terms of a specific product or service
* Value is determined by the customers who want to buy the right product with the right capabilities at the right price
* This means that the product must be “right” every time
  + From design to manufacture, from deliver to error free operation
* Work on making their processes right by eliminating waste
* Some questions to ask what the customer value
  + What is the problem that impacts the customer
  + What is the problem that the team is going to take action on
  + Why is the project so important that the organization should address it
  + Why is the project being done

1. Map the Value Stream

* **Map** all of the steps, value added and non-value added that bring a product or service to the customer
* eliminate whenever possible those steps that do not create value
* detailed process flow diagram should be generated for each product or product category
* Some Review Questions:
  + Did the team identify the waste in the process
  + Does the team understand how the whole process works
  + Did the team manage to complete a detailed process flow diagram at this stage

1. Create Flow

* The continuous **flow** of products, services and information from end to end through the process
* Make the value creating steps occur in tight sequence so the product will flow smoothly toward the customer
* Specific process waste is identified at each stage of process flow and is eliminated
* Lean enterprise implements 5S – a tool developed for reducing the slack hidden in manufacturing process
  + Sort: sorting or segregating through the contents of the workplace and removing all unnecessary items
  + Straighten: putting or arranging the necessary items in their place and providing easy access by clear identification
  + Shine: cleaning everything, keeping it clean and using cleaning to inspect the workplace and equipment for defects
  + Standardize: creating visual controls and guidelines for keeping the workplace organized, orderly and clean, in other words, maintain the shine
  + Sustain: instituting training and discipline to ensure that everyone flows the 5S standards

1. Establish Pull

* Nothing is done by the upstream process until the downstream customer signals the need
* Actual demands **pulls** product/service through the value stream
* As flow is introduced, let customers pull value from the next upstream activity
* This step can be directed toward either removing excess capacity (inventory) or increasing the rate of pull
* A pull system, which on the production side is making a product at the same rate at which it is being sold
* On the supply side, a pull system is flowing resources into a production process by replacing only what has been consumed

1. Seek Perfection

* The completed elimination of waste so all activities create value for the customer by breakthrough and **continuous improvement** projects
* As value is specified, value streams are identified, wasted steps are removed, and flow and pull are introduced, begin the process again and continue it until a state of perfection is reached in which perfect value is created with no waste
* Common mistakes in implementing lean
  + Lean must never be seen as a tool for headcount reduction or mindless cost cutting
    - It’s purpose is to create value through eliminating waste
  + Avoid from the beginning is the impulse to implement individual lean tools without seeking to understand the system in which they fit
    - This is hard to avoid since many tools deliver immediate payoffs
    - Ultimately all lean workers must understand the “why” behind the tools, or their value will be lost
  + Lean beginners should limit the scope of their initial project
    - It’s better to insure success, be sure that they have deep knowledge and a gemba attitude (Note: gemba = “the real place” in Japanese)

Provide Pointers to additional material on the topic for interested readers

Example of Lean Method

Why should software developers care about this topic?

* Lean offers insight into strategies for improving our software process
  + Ex: understanding the source of waste in IT, we can begin to identify it and then eliminate it
* Lean principles provide a philosophical foundation for scaling agile approaches
* Provides techniques for identifying waste
* Helps calculate overall time efficiency of what we are doing
* Lean developers can build software better and cheaper than competitors using traditional methods
* Emphasis on:
  + Building only what is needed
  + Eliminating anything that does not add value
  + Stopping production if something goes wrong

Other information

References

* <https://en.wikipedia.org/wiki/Lean_software_development>
* http://www.allaboutagile.com/category/lean-development/
* <http://www.netobjectives.com/blogs/differences-between-lean-manufacturing-and-lean-software-development>
* <http://www.dummies.com/how-to/content/lean-for-dummies-cheat-sheet.html>
* <http://www.lean.org/WhatsLean/Principles.cfm>
* <http://www.lean.org/WhatsLean/CommonLeanQuestions.cfm>
* <http://www.isixsigma.com/methodology/lean-methodology/learning-think-lean-six-steps-review-points/>