**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? (http://www.allaboutagile.com/7-key-principles-of-lean-software-development-2/)
  + Principles
    - 1. Eliminate Waste
      * 3 general forms of waste identified by the Toyota Production System
        + Muda – meaing 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.
* Term lean software development originated in a book by the same name, written by Mary Poppendieck and Tom Poppendieck
  + Book presents the traditional lean principles in a modified form as well as a set of 22 tools and compares the tools to agile practices
* Lean development originated from lean manufacturing and the Toyota Production System in Japan
* 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
* Eliminate Waste
  + Learn to see waste
  + Learn to reduce waste
    - Reduce management activities
    - Minimize tracking by creating a smooth flowing work system
  + Rethink Authorization systems
  + Retrain your brain to see waste – “Why am I really doing this”
  + Map your value stream to identify inefficiencies

Compare to Japanese manufacturing style

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

Provide Pointers to additional material on the topic for interested readers

Example of Lean Method

Why should software developers care about this topic?

Other information

References

* <https://en.wikipedia.org/wiki/Lean_software_development>