What is Lean Software Development?

- Central concept is to take **lean production concepts from manufacturing** and apply them to software development
- In manufacturing, lean production is concerned with **eliminating waste**, **focusing** all effort on **valuable deliverables**, and **optimising** the **production process** to get from start to finish in the most efficient way.
- Very strong ideological overlap with Agile software development, has found a home within Agile software development.
- Lean Software Development is a kind of "sub-culture" within Agile Software Development.

Apart from what I used to be, what is Lean …

- Even within Lean Software Development there is variation in application:
  - Some focus on Lean principles applied to common development practices
  - some focus on workflow management
  - others focus on complementary product development processes used by Toyota and other Lean producers
- According to Jim Womack and Daniel Jones (*Lean Thinking*, 1996), there are 5 principles associated with lean thinking…

James P. Womack , Founder and Senior Advisor, Lean Enterprise Institute (jwomack@lean.org)

5 Principles of lean thinking

1. Specify **value** from the standpoint of the end customer by product family.
2. Identify all the steps in the **value stream** for each product family, eliminating every step and every action and every practice that does not create value.
3. Make the remaining value-creating steps occur in a tight and integrated sequence so the product will **flow** smoothly toward the customer.
4. As flow is introduced, let **customers pull value** from the next upstream activity.
5. As these steps lead to greater transparency, enabling managers and teams to eliminate further waste, **pursue perfection through continuous improvement**.

Examples of Lean Software Development approaches

- Of course, any approach to software development that embraces some or all of the 5 principles of lean thinking can claim to be a Lean Software Development approach…
- (Probably) the two best known (or most talked about) Lean Software Development approaches are:

1. **Lean Software Development**: **An Agile Toolkit** (**Poppendieck** X 2) – applying lean principles to software development
2. **Kanban (David Anderson) –** workflow management

7 Agile Principles for Lean Software Development (Poppendiecks)

Decide as late as possible does not mean procrastinate.

Software dev is always associated with some uncertainty.

Deliver as fast as possible does not mean rush and do sloppy work.

It is not the biggest that survives, but the fastest.

Work-out technique: managers are thought how to listen to developers

Empower the team does not mean abandon leadership

Eliminate waste does not mean throw away all documentation.

Everything not adding value to the customer is considered to be waste

1. Eliminate Waste
2. Amplify Learning
3. Decide as Late as Possible
4. Deliver as Fast as Possible
5. Empower the Team
6. Build Integrity In
7. See the Whole

Understand the broader landscape – other vendors / suppliers / interactions

See the whole does not mean ignore the details.

Amplify learning does not mean keep on changing your mind.

Software development is a continuous learning process – champions short iterations and effective customer engagement sessions

The system's separate components work well together as a whole with balance between flexibility, maintainability, efficiency, and responsiveness

Build integrity in does not mean big, upfront design.

Red text is guidance on the application of the principle, and in particular on interpretations to avoid.

Pros & Cons of Lean S/W Dev

Similar but slightly different to the general Pros & Cons of mainstream agile over traditional approaches

Pros	Cons
Facilitates flexibility	Requires a v motivated team
Encourages an SPI focus	Strong discipline required
Focus on value-add	More power = more responsibility

Note: As with other agile software development, there is a **power shift** away from management and towards individual developers…

Note: What is *waste* in any case? Are there waste identification challenges? Wasteful *when*?

Note: To what extent can we industrialise output in software development.?

Lean is Not a Synonym for Agile

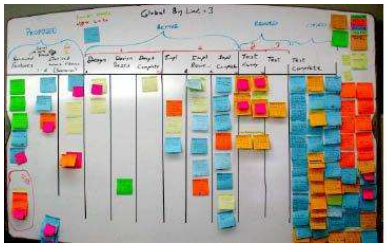
- Agile methods: Scrum, Extreme Programming (XP), Dynamic Systems Development Model (DSDM), Crystal, etc.
- Agile and lean are similar and related philosophies
  - Different roots, common goals
- Lean has heavy emphasis on the "whole" and "end-to-end" process (order to cash)
  - Agile methods focus on flexibility and change
- … no common agreement

Kanban

- Originally the Japanese word Kanban is two words kan and ban; kan means **visual** and ban means **card**.
- Kanban software development originated by **David Anderson**

Kanban

- Visual Card….



Kanban – Origins & Workflow

- It is claimed that many of the practices and heuristics have been seen on other Agile teams before but they were first described as **a cohesive whole** by David Anderson.
- David's innovation was to explicitly **limit the work in progress**.
- This had been done by other Agile teams before but in Kanban there is a **well-known limit on the number of work items** which may be worked on at one time.
- The limit is usually **quite low**, often the limit is approximately the same as the number of developers on the team or slightly less.

http://translated.by/you/10-things-to-know-about-kanban-software-development/original/

What is Kanban?

- Kanban is a method for managing the creation of products with an **emphasis on continual delivery** while not overburdening the development team.
- Like scrum, Kanban is a process designed to help teams work together more effectively.
- Kanban is based on 3 basic principles:
  1. Visualize what you do today (workflow): seeing all the items in context of each other can be very informative
  2. Limit the amount of work in progress (WIP): this helps balance the flow-based approach so teams dont start and commit to too much work at once
  3. Enhance flow: when something is finished, the next highest thing from the backlog is pulled into play

Kanban

- Kanban derives more directly from Lean Thinking and Lean software development then many of the previous Agile techniques.
- Kanban teams focus more on work flow, the **time it takes to get work from one end of the pile to the other**.
- In the **extreme, Kanban** teams could **dispense with work estimating** (which do not add value), **iteration planning meeting** (planning is an ongoing process), fixed release dates (they release when it is ready) and **scheduled retrospectives** (they have a stop-the-line approach to address a problem when it is seen) – though this could have **negative knock on effects(?)**, e.g. fixed price contracts with penalty clauses for late delivery.
- In Lean systems the cards are used to pass information, to signal when limits are reached, to signal an out-of-stock situation or some other trigger for action.
- **Kanban has come to mean more than it literally meant** (visual card), and now the word is used as the name of a method rather than just an artefact from applying the method.

http://translated.by/you/10-things-to-know-about-kanban-software-development/original/

Kanban – in practice?

- Kanban was **in vogue in certain quarters** until recently (and may still be)...
- **How much** is it **used**? Who knows?
- Like earlier attempts to apply concepts from manufacturing to software development, it may ultimately get merged in with other techniques. **Software development != manufacturing.**

Kanban – Critical Review

- The **jury is still out** on Kanban – as a philosophy it makes a lot of sense. However, there are a some peculiarities of software development that may be overlooked:
  - **Software development is not manufacturing** =>
    - Being absolutely delivery focused (i.e. the impending/imminent delivery) **could overlook some** of the maintenance **concerns of software products** (e.g. Do cars, once shipped, undergo anywhere like the same **amount of upgrades** and **extensions** and **variability of execution environment** that a software product does?)
    - Is the **software lifecycle shorter** than some traditional manufactured products? A car may be still be operational in 25 years, can the same be said for most software products?
  - **Software development is software development =>**
    - **Tacit knowledge** that needs to be carefully managed (esp. in larger organisations or organisations with significant personnel growth/churn)
    - **Version control, seamless product upgrading** and **configuration management** concerns
    - We continually build our product?

Kanban – Critical Review

- In the case of Kanban, agile coach Allen Kelly (www.allankelly.net) points out that **depending on your point of view**, Kanban might be:

  a. The first second generation Agile development method
  b. A collection of common heuristics
  c. Dangerously unAgile
  d. None of the above (a-c)
  e. All of above (a-c)
  f. Just another marketing term

- So – it is even a matter of opinion as to what exactly Kanban is ... ☺