

# Lean Software Development

Andre van der Schyff

# The Very Beginning

The Toyota Production System (TPS)

# Toyota Production System

## A Brief History

- 1937 – Kiichiro Toyoda founds Toyota Motor Company
- 1938 – Asks his cousin, Eiji Toyoda, to oversee new factory, later renamed Toyota City
- 1950s – Eiji Toyoda visits Ford's River Rouge Plant
  - Very impressed with scale
  - Many inefficiencies
- Decides to adopt US manufacturing techniques with a focus on quality
- Collaborates with Taiichi Ohno to develop the 'Toyota Way'
  - These are the 14 principles underlying the TPS
- From Toyota: TPS is a way of "making things" that is sometimes referred to as a "lean manufacturing system"



# Toyota Production System

## The Two Founding Concepts (or Philosophies)

### Jidoka

"Automation with a Human Touch"  
Highlighting / Visualisation of Problems

- Build quality into the process
- Automatically stop work when complete or defect is found
- Alert operator of the problem
- Daily Improvements
- Required for JIT to work



### Just in Time

Productivity Improvement

- Making only "what is needed, when it is needed, and in the amount needed"
- Complete elimination of waste, inconsistencies, and unreasonable requirements
- Build in the shortest time possible
- Kanban System

# Toyota Production System

## The Kanban System

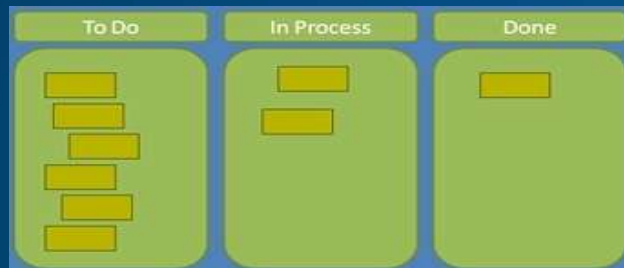
- Common Japanese Term for "Signboard" or "Billboard"
- Supermarket concept
- Kanban is a means through which JIT is achieved
- A signaling system to trigger action, typically a card
- Pull-based system
  - Push-based system requires accurate demand forecasting
  - Minimizes waste by only getting what is necessary, when it is necessary



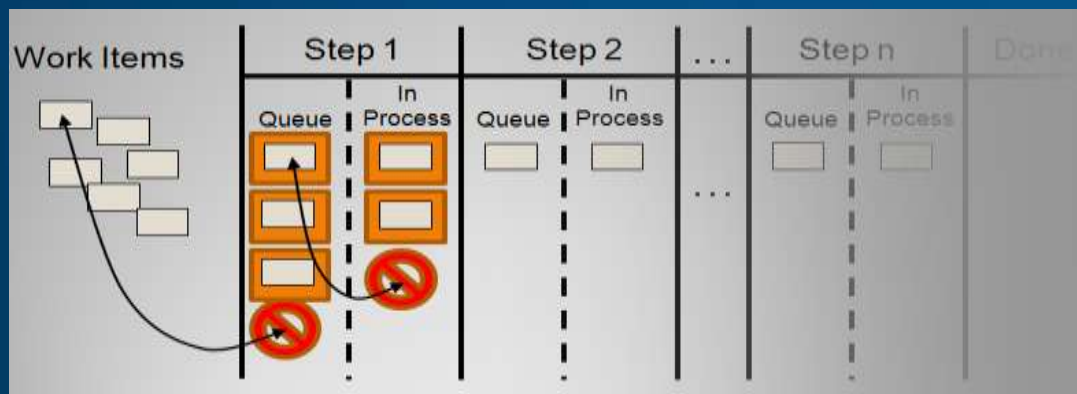
# Toyota Production System

## The Kanban System - Use in Agile

- Typically Used in Agile as a Kanban Board and Cards
- Simplest Form: ToDo, Doing, Done



- More Complex: Work Items moving to downstream Queues / WIP
  - Important element: Queue limits & WIP limits



# Toyota Production System

## The Principles - 'The Toyota Way'

### Section I: Long-Term Philosophy

1. Base your management decisions on a long-term philosophy, even at the expense of short-term financial goals.

# Toyota Production System

## The Principles - 'The Toyota Way'

### Section II: The Right Process Will Produce the Right Results

1. Create a continuous process flow to bring problems to the surface.
2. Use "pull" systems to avoid overproduction.
3. Level out the workload
4. Build a culture of stopping to fix problems, to get quality right the first time.
5. Standardized tasks and processes are the foundation for continuous improvement and employee empowerment.
6. Use visual control so no problems are hidden.
7. Use only reliable, thoroughly tested technology that serves your people and processes.

# Toyota Production System

## The Principles - 'The Toyota Way'

### Section III: Add Value to the Organisation by Developing Your People

1. Grow leaders who thoroughly understand the work, live the philosophy, and teach it to others.
2. Develop exceptional people and teams who follow your company's philosophy.
3. Respect your extended network of partners and suppliers by challenging them and helping them improve.

# Toyota Production System

## The Principles - 'The Toyota Way'

### Section IV: Continuously Solving Root Problems Drives Organisational Learning

1. Go and see for yourself to thoroughly understand the situation
2. Make decisions slowly by consensus, thoroughly considering all options; implement decisions rapidly
3. Become a learning organization through relentless reflection and continuous improvement

# Transition to Software Development

Mary & Tom Poppendieck

- Started her career as process control programmer
- Manage IT department of a manufacturing plant
- Considered retirement in 1998
- Instead managed government project
  - Encountered Waterfall
- Tom Poppendieck spent 25 year career in software development
- Co-authored **Lean Software Development: An Agile Toolkit**
  - Explain how the lean principles from manufacturing offer a better approach to software development



# Lean Principles

“It has been my observation that most people get ahead during the time that others waste.” - Henry Ford

Eliminate Waste

# Principle 1: Eliminate Waste

## What is waste?

- Three types of activities exist:
  - Those that definitely create value
  - Those that create no value, but are necessary given the current state of the system
  - Those that create no value and can be eliminated
- Everything not adding value to the Customer is waste

# Principle 1: Eliminate Waste

## What is waste?

### Waste in Manufacturing

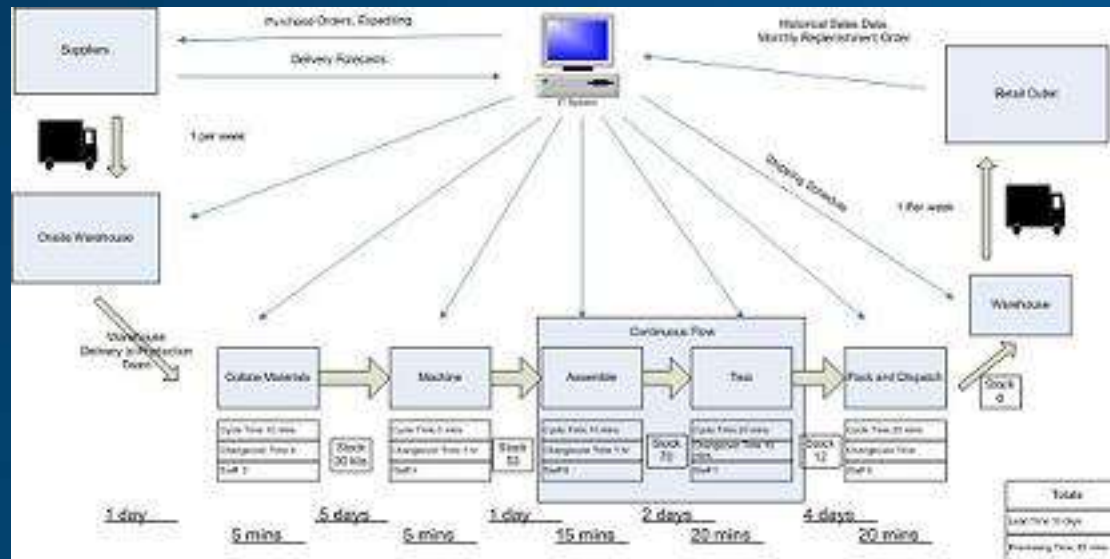
- Overproduction
- Inventory
- Extra processing steps
- Motion
- Defects
- Wait-time
- Transportation

### Waste in Software

- Extra Features
- Requirements
- Extra Steps
- Finding Information
- Bugs
- Decisions, customers etc
- Hand overs

# Principle 1: Eliminate Waste

## How? Value Stream Mapping?



- Draw each step in your process
- For each step, measure time taken, resources use, key performance indicators etc
- Map using icons
- Critique current state
- Come up with new plan and repeat

# Principle 1: Eliminate Waste

How?

**Step 1:** Identify what value really is

**Step 2:** Identify the waste (including things that appear important)

**Step 3:** Eliminate it!

# Principle 1: Eliminate Waste

## Where to find waste: The biggest culprits

- **Extra Features**
  - Try to develop the 20% of features that deliver 80% of the value
- **Churn**
  - From the dictionary: *agitated vigorously; in a state of turbulence;*
  - If you have requirements churn, you are specifying too early.
  - If you have test and fix cycles, you are testing too late.
- **Crossing Organisational Boundaries**
  - Creates buffers that slows down response time
- **Partially Done Work**
  - It gets lost, grows obsolete, hides quality problems, ties up money, and introduces risk

# Principle 1: Eliminate Waste

Sidetrack... what about our own practices?

- According to recent studies:
  - **TDD** teams took longer to complete their projects—15 to 35 percent longer
  - More **code coverage** in testing does not decrease post-release fixes
- 'Greenfields' development

# Principle 1: Eliminate Waste

Sidetrack... what about our own practices?

- **TDD** teams produced code that was 60 to 90 percent better in terms of defect density than non-TDD teams
- **Code coverage** is more beneficial for more complex code

# Lean Principles

“Learning is not compulsory but neither is survival.”

- W Edwards Deming

Eliminate Waste  
Create Knowledge

# Principle 2: Create Knowledge

## Two Approaches to Creating Software

### Production

- Quality is conformance to requirements
- Variable results are bad
- Iteration generates waste

### Development

- Quality is fitness for use
- Variable results are good
- Iteration generates value

# Principle 2: Create Knowledge

## Delivering a Service

- Customers will perceive quality if a system solves their changing problems
  - So, quality is determined by fitness for use
- Variation is good
- Unstructured approach is required to solve poorly understood problems
  - Try-it, Test-it, Fix-it
  - Generate information

# Principle 2: Create Knowledge

## Create Short Feedback Loops

- Tendency to instill 'discipline' in troubled projects, making things worse
- Rather use short feedback loops:
  - Test code immediately, and stop when they break
  - Write code instead of documentation to test ideas
  - Give users options instead of gathering requirements
  - Try different tools
  - Try to do everything for an immediate customer
  - And use short iterations

# Principle 2: Create Knowledge

## Iterations

- Iteration planning:
  - Highest priority / risk first
  - Fixed time-box (iterations can be variable)
  - Long enough for meaningful dev, short enough for feedback
- Team to commit to time-box
  - Gets more accurate with time
- Convergence achieved through:
  - Correct iteration length
  - Highest priority / risk first
  - Converges into solution although no set end

# Principle 2: Create Knowledge

## Synchronisation of Multiple Teams

- Continuous Integration
- Spanning Application
  - Simple app spanning all layers
  - Tests components
- Matrix
  - Develop interfaces first
  - Teams then split off

# Principle 2: Create Knowledge

## Set based development

- Point-based vs Set-based approach to problems
- Set-base development
  - Develop multiple options
  - Communicate constraints
  - Let the solution emerge

# Lean Principles

“Quality is not an act, it is a habit.” - Aristotle

Eliminate Waste  
Create Knowledge  
Build Quality In

# Principle 3: Build Quality In

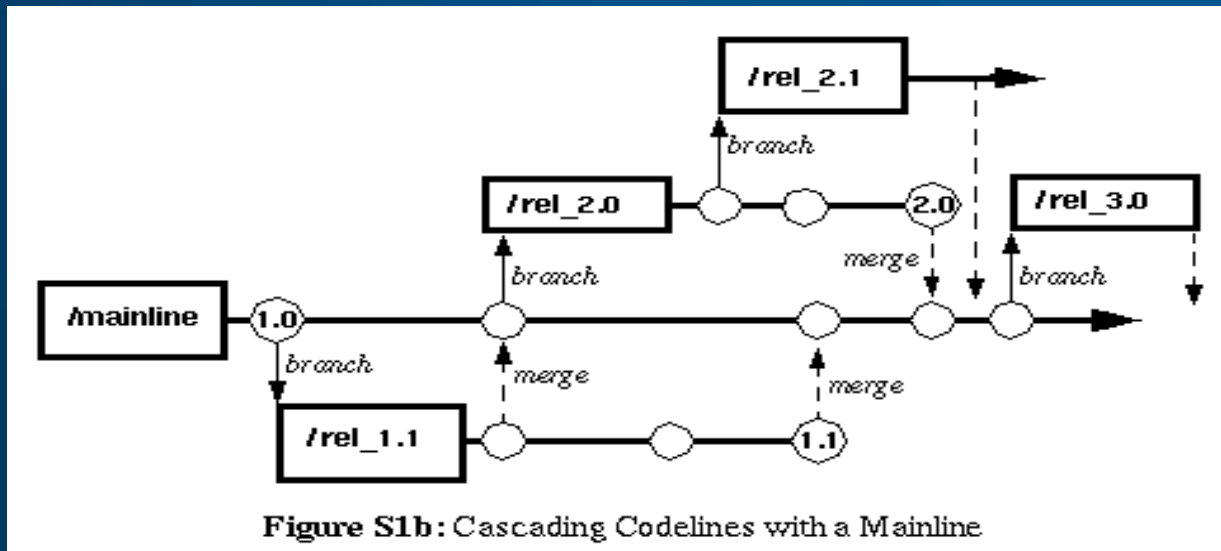
## Goal: Build Integrity In

- Perceived Integrity:
  - System as a whole achieves balance of function, usability, reliability and economy
  - Intuitive
- Conceptual Integrity:
  - Central concepts work together / Effective Architecture
  - Required to achieve perceived integrity
  - Not seen by customer
- Achieve through good information flow from customer to development team and within team itself
  - Consider current and future scenarios
  - Good environment for communication
  - Increase everyone's domain knowledge

# Principle 3: Build Quality In

## Technical Practices

- Refactoring
- Test Driven Development
- Continuous Integration
  - Also automate testing
- Nested Synchronisation
  - Mainline Pattern(s): Branch off and bring back to central line instead of cascading



# Lean Principles

“Delay is preferable to error.” - Thomas Jefferson

Eliminate Waste  
Create Knowledge  
Build Quality In  
Decide As Late As Possible

# Principle 4: Decide as late as possible

## Maintain Options

- Prefer breadth-first over depth-first
- Develop multiple solutions
- Keep your options open as long as practical, but no longer
  - Balancing act
- Avoid detailed discussion of things not in development or nearby in planning
  - Maintain Flow
- Not always necessary - develop instinct with experience

# Principle 4: Decide as late as possible

## Dreyfus Model of Skills Acquisition

Level	Input	Orientation
Novice	Context Free Rules / Recipes	Self
Advanced Beginner	Applies Guidelines to Adapt Rules	Other
Competent	Applies Processes to Respond to Unexpected	Process
Proficient	Responds to Nuances / Combines Processes	System
Expert	Uses intuition, grasps whole situation	Larger System

# Principle 4: Decide as late as possible

## Break Dependencies

- Dependencies force you into commitments
- Refactor
- Architecture should support addition of features

# Principle 4: Decide as late as possible

Schedule irreversible decisions at the last possible moment

- This is when you have the most information
- Avoid making critical decision early in the process
- Try to make any decision reversible
- Remember planning is not the same as commitment
  - Companies need to plan, but stay flexible

# Lean Principles

“You can't just ask customers what they want and then try to give that to them. By the time you get it built, they'll want something new.” - Steve Jobs

Eliminate Waste  
Create Knowledge  
Build Quality In  
Decide As Late As Possible  
**Deliver As Fast As Possible**

# Principle 5: Deliver as fast as possible

## The advantages

- Cost advantage over competitors
  - Requires you to eliminate waste
- Low defect rate: Speed impossible without quality
  - Requires reflexes and stop-the-line culture
  - Not Hacking
- Develops customer understanding: Able to take experimental approach

# Principle 5: Deliver as fast as possible

## Reduce cycle time

- Steady rate of arrival: Prevent queues
- Steady rate of service
- Slack: Run at 80-90%
- Bottom line: Limit work to capacity
  - Make iterations reliable and repeatable

# Lean Principles

“Leaders don't create followers, they create more leaders.” - Tom Peters

Eliminate Waste  
Create Knowledge  
Build Quality In  
Decide As Late As Possible  
Deliver As Fast As Possible  
Respect People

# Principle 6: Respect People

## Origins

- Three out of four cornerstones of TPS concern people:
  - **Entrepreneurial Leader:** Develop great leaders
  - **Expert Technical Workforce:** Nurture Expertise
  - **Responsibility-Based Control and Planning:** Give teams general plans and reasonable goals, and trust them to self-organise

# Principle 6: Respect People

## Empower the team

- People are not resources
  - Need more than list of tasks and deadlines
- Effective teams are motivated by:
  - Common goal
  - Effective leaders
- And thrive on:
  - Pride
  - Commitment
  - Trust
  - Applause
- Group or Team?

# Principle 6: Respect People

## Achieving the goal

- Train team leaders
  - Do not impose anything, they know better in the area
  - Teach how to set direction, align people, and enable motivation
  - Create safety
- Move responsibility and decisions to lowest possible level
  - Be an enabler and facilitator, not a driver
- Foster pride in workmanship
  - Reward the team
  - Let people decide how and when to meet goals

# Lean Principles

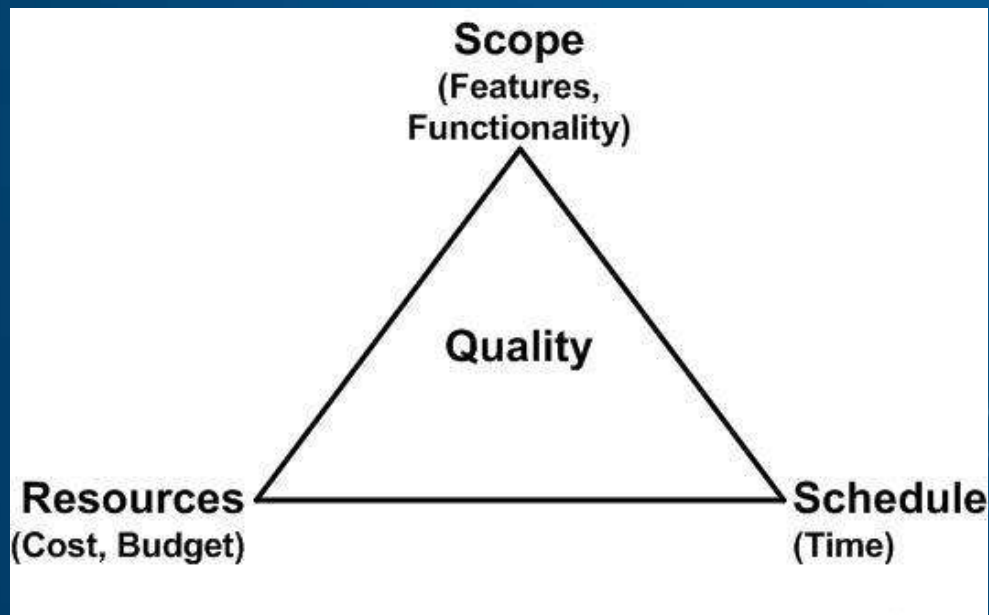
“Change is vital, improvement the logical form of change.” - James Cash Penney

Eliminate Waste  
Create Knowledge  
Build Quality In  
Decide As Late As Possible  
Deliver As Fast As Possible  
Respect People  
Improve The Process

# Principle 7: Improve the Process

## See the Whole

- Inefficiencies occur across organisational boundaries
  - Single management system: 20-30% cost savings
- Tend to break process into components
  - Measure each component
  - Optimise each component
  - Waste emerge in between
- Typical solution: Add more measurements



# Principle 7: Improve the Process

## Measure Up

- Solution: Decrease the number of measurements
  - Find right high-level measurement for low-level metrics
  - Establish basis for making trade-offs
- Examples:
  - Measure development process by cycle time
  - Measure team performance by business value
  - Measure project success by ROI

# Relationship to Agile

Is Lean and Agile the same thing?

# The Agile Manifesto

Similar, but not the same

- Early and continuous delivery of valuable software
- Deliver working software frequently
- Self-organising teams
- Reflect often on how to become more effective
- Should be able to maintain constant pace indefinitely
- Working software is primary measure of progress
- Attention to technical excellence
- Simplicity is essential
- Build projects around motivated individuals
- Welcome changing requirements
- Business people and developers must work together
- Face to face conversation



Less Compatible?

# XP Principles

## A Step Further Away

### Fundamental

- Rapid Feedback
- Quality Work
- Assume Simplicity
- Embracing Change
- Incremental Change


### Additional

- Small Initial Investment
- Concrete Experiments
- Teach Learning
- Play to Win
- Honest Measurement
- Open Honest Comms
- Accepted Responsibility
- Local Adaption
- Work with Instincts
- Travel Light

Less Compatible?



Less Compatible?



# Some Expert Opinions

Martin Fowler

"... lean and agile are deeply intertwined in the software world. You can't really talk about them being alternatives, if you are doing agile you are doing lean and vice-versa. Agile was always meant as a very broad concept, a core set of values and principles that was shared by processes that look superficially different. You don't do agile *or* lean you do agile *and* lean. The only question is how explicitly you use ideas that draw directly from lean manufacturing."

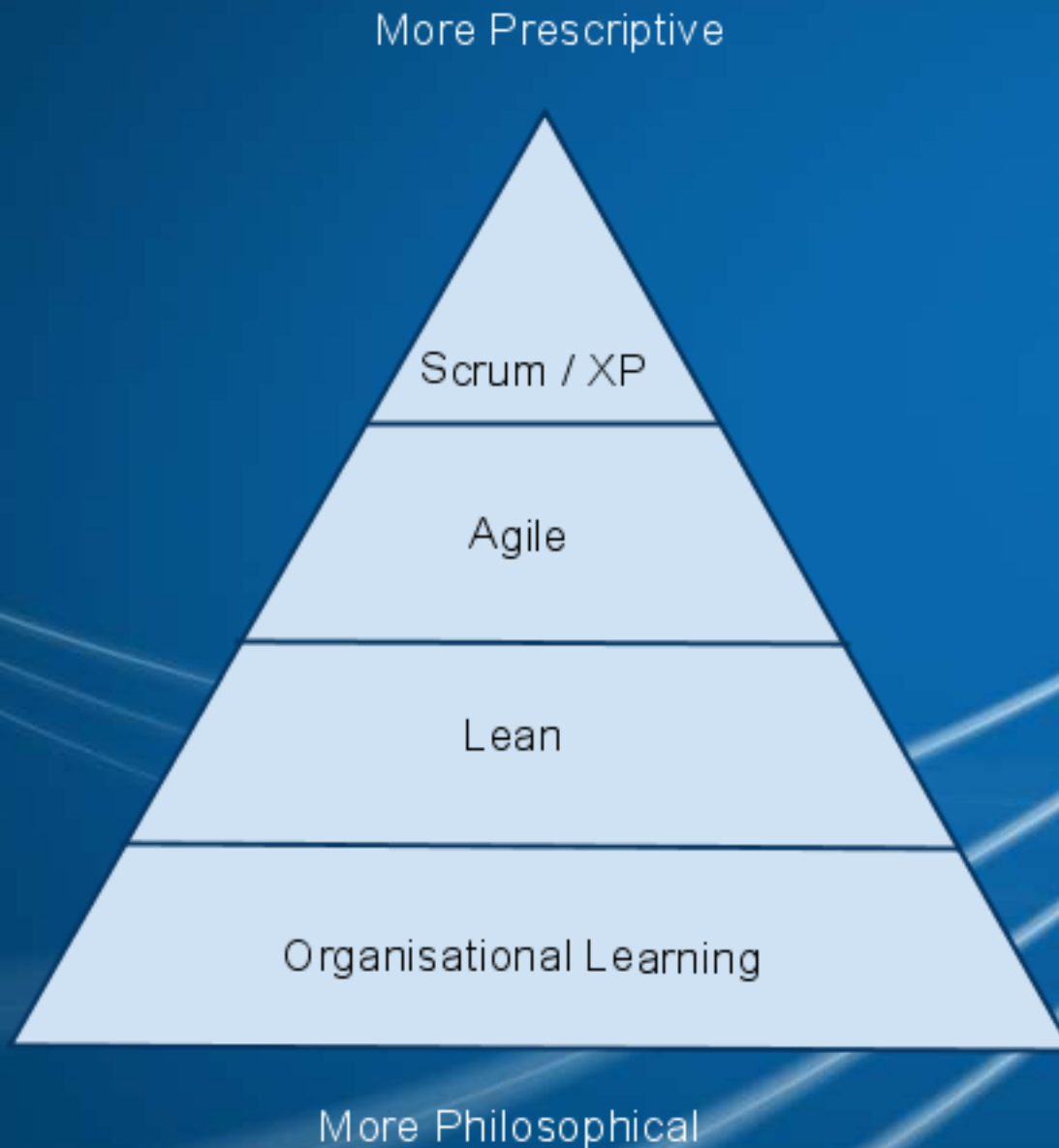
# Some Expert Opinions

Kent Beck

“I think that Agile and Lean are strongly related, but that they are two different ideas. Lean aims to achieve efficiency through eliminating waste and respecting people. Agility is a by-product in lean as rapid cycles are required to identify and eliminate waste. Agile software development aims to meet the evolving needs of customers through the early and continuous deployment of valuable software. The values, principles, and practices of the two approaches are different, even though complementary.”

In response to a post by Ryan Martens: Agile and Lean Development: An Oxymoron?

# One Common View



The background is a solid blue color with several white, curved, glowing lines that sweep across the top and middle of the frame, creating a sense of motion and depth.

Comments / Questions?