ACE 303
Lean Development

Agile Center of Excellence

Lean Principles

1. Eliminate Waste
2. Build Quality In
3. Create Knowledge
4. Defer Commitment
5. Deliver Fast
6. Respect People
7. Optimize the Whole

Intent

- Goal
  - Efficient, quick, sustainable delivery of value to the customer
- Myth
  - Only works with waterfall
  - Only works in manufacturing
  - Getting more with less doesn't mean you need less people

Value is measured by a customer's willingness to pay for it

Eliminate Waste

- Waste - any part of a process that does not add value for the customer
- Maximize the amount of work not done - *Agile Manifesto*
- Categories of Waste
  - Necessary
    - Does not add value, but required to support the process
  - Unnecessary
    - Does not add value or support the process (ex. - Status Meeting that could be replaced by better tools that are kept up to date)
    - What would happen if these activities were eliminated
  - Goal is to make necessary waste, unnecessary, and eventually eliminated

7 Types of Waste

1. Partially completed work
   a. Unused or abandoned designs, code, tests
   b. Occurs when you plan or start to early, or have frequent priority changes
   c. Waste of time or resources

2. Extra Features
   a. Minimum features required to fulfill a customers needs
   b. 80% have little value

7 Types of Waste

3. Relearning - What did you have for lunch Monday, and Monday 1 week ago
   a. Too much or too little documentation
   b. Not utilizing co-workers expertise

4. Handoffs
   a. Implicit / Tacit Knowledge
      i. 50% is lost during handoffs
   b. Explicit Knowledge

7 Types of Waste

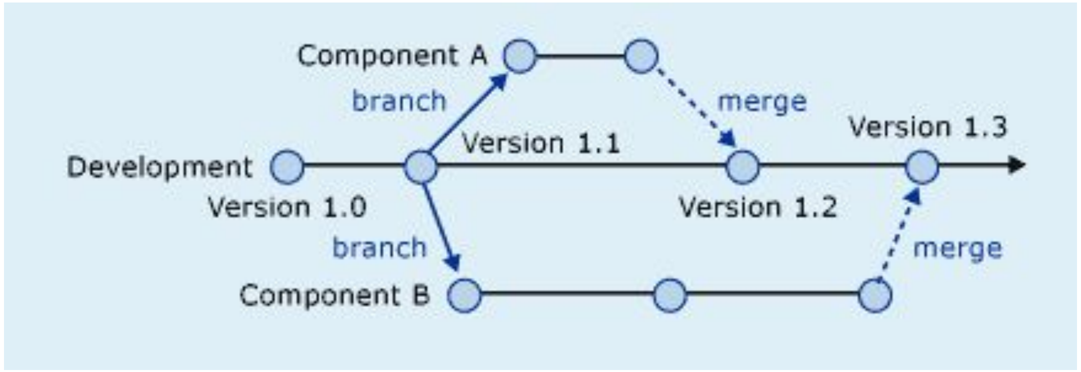
5. Task Switching - Name game
    a. Time lost due to context switching
    b. Interruptions or excessive WIP

6. Delay
    a. Dependencies
        i. Every dependency doubles the chances of a delay

7. Defects
    a. Failure Demand - Work required now because of poor work in the past
    b. Prevention is better the correction

Build Quality In

- Continuous Integration
  - Frequent (daily / multiple times a day) integration (merging) of code from separate branches back to the source code
  - Identify and resolve integration and dependency issues as quickly as possible
  - Automated build and test tools
  - Branch, develop, test locally, test in integration, merge



Build Quality In

- Refactoring
  - Improvement of code without changing its externally observable behavior
    - End User should not notice difference in outcome, other than perhaps improved performance
    - If refactoring impacts the End User, this can erode trust
  - Expectation, not exception
  - Refactor to make code less painful, but may not be perfect
  - Resources
    - Refactoring.com/catalog
    - LinkedIn Learning - Agile Software Development - Refactoring

Build Quality In - Mistake Proofing

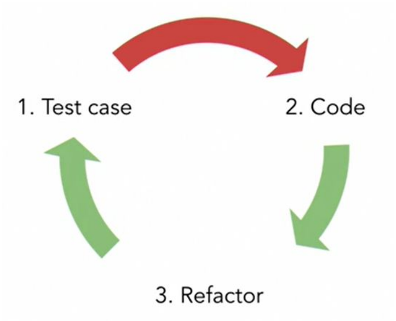
- Poka-Yoke
  - Design your process so that mistakes are impossible or at least easily detected and corrected *
- Goal of Mistake Proofing
  - Prevent mistakes when possible
  - Make mistakes obvious when they occur
  - Seek to reduce impact of mistakes when they occur

https://web.archive.org/web/20141227002617/http://facultyweb.berry.edu/jgrout/pokasoft.html

Build Quality In - Mistake Proofing

- When to implement Mistake Proofing
  - Development Process
    - Automation - ensure repetitive steps are performed consistently
    - Test Driven Development (TDD)

      (applies to Unit Testing - tactical vs. functional)
      - Write failing test case
      - Write code to pass the test
      - Refactor as needed
  - User Experience
    - Examples
      - Data validation before proceeding in UI
      - Electrical outlet on fits one way

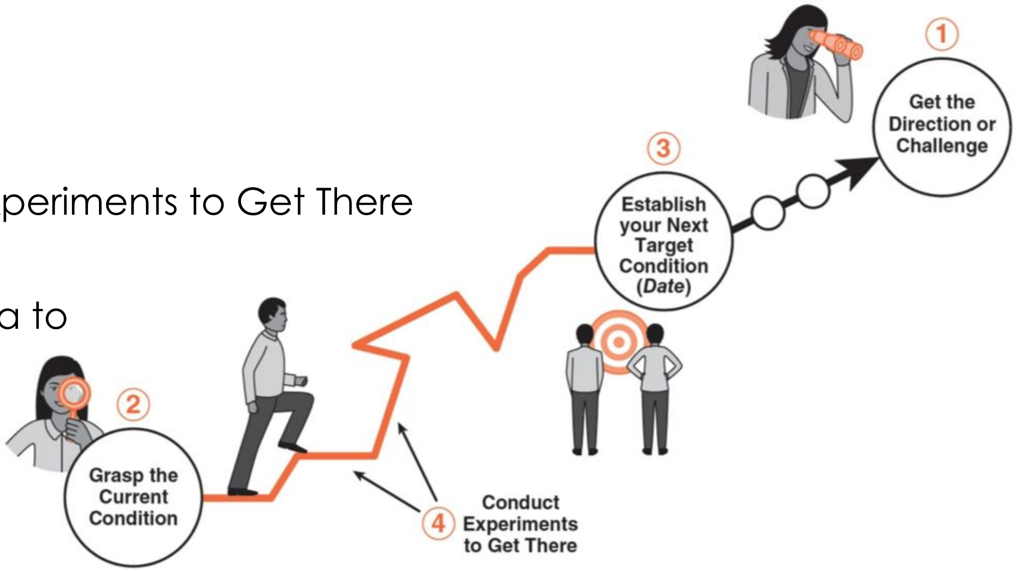


Quality Cadences

- Frequent planning sessions to align on future priority
- Daily meetings for coordinating current WIP
- Frequent reviews to assess completed functionality for fitness of use
- Regular retrospectives for analysis and improvement
- Frequent constructive feedback
  - Expectation vs. Experience

Create Knowledge - Kata

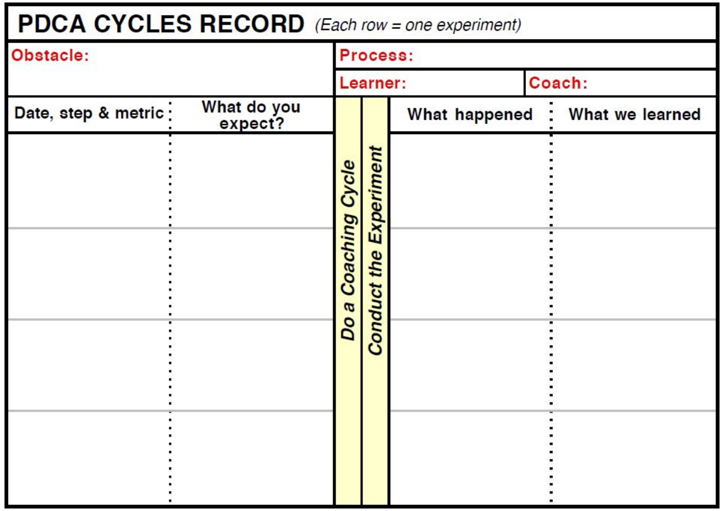
- Kata
  - Japanese for a pattern practiced to learn a skill
- Lean Kata *
  - Improvement Kata
    - Determine a Vision or Goal
    - Grasp the Current Condition
    - Define the Next Target
    - Execute Small, Quick, Iterative Experiments to Get There
  - Coaching Kata
    - Leaders teach Improvement Kata to everyone in the organization


1
Get the Direction or Challenge
3
Establish your Next Target Condition (Date)
2
Grasp the Current Condition
4
Conduct Experiments to Get There

* (Adapted from Rother, 2010, and Shook, 2008.)

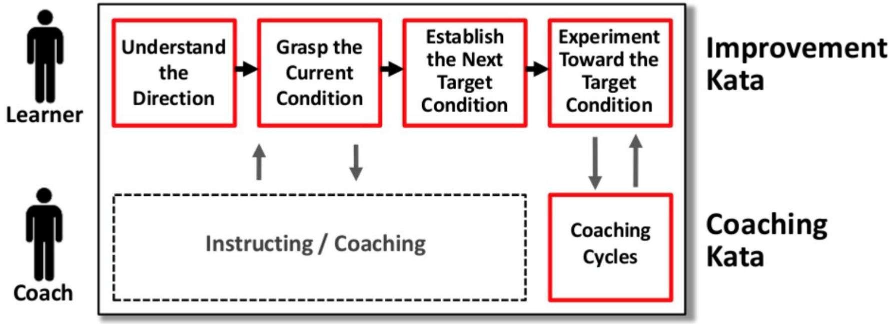
5 Coaching Kata Questions

1. What are we trying to achieve
(Goal)
1. Where are we now
(In Relation To Goal)
2. What obstacles is currently in our way
(Immediate / Short Term)
3. What's our next step and what do we expect
(Measurable Results)
4. When can we see what we've learned from taking that step
(Observable, Actionable Output)



PDCA CYCLES RECORD *(Each row = one experiment)*

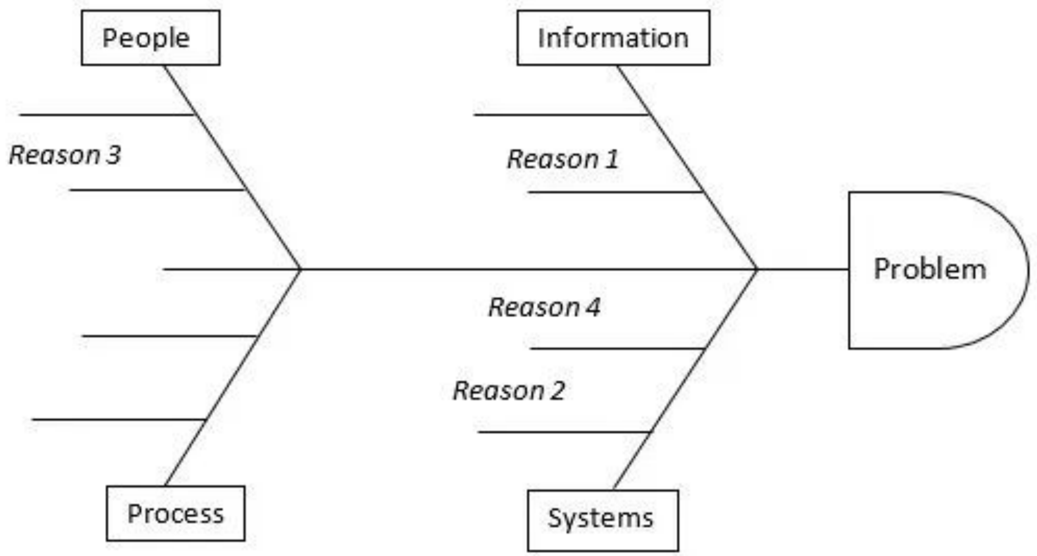
Obstacle:		Process:			
		Learner:		Coach:	
Date, step & metric	What do you expect?	*Do a Coaching Cycle*	*Conduct the Experiment*	What happened	What we learned



* (Adapted from Rother, 2010, and Shook, 2008.)

Create Knowledge - A3

- Background
  - Importance
  - Context
- Current State
  - Problem Statement
  - Process Mapping
- Set Target / Goals
  - SMART Goals / Desired Result
  - Success Metrics
- Root Cause Analysis
  - Fishbone Diagram
  - 5 Whys
  - Data (Pareto, Scatter Diagram)



Create Knowledge - A3

- Recommendations
  - Cost / Benefits / Feasibility
  - List of Options / Actions Items
  - Assign Responsible Stakeholders
- Plan
  - Key Actions in Sequence
  - People / Support / Resources
  - How to Measure Success
- Follow Up
  - Lessons Learned
  - Trend Analysis

Defer Commitment

- Commit as late as possible
  - Last Responsible Moment = Cost of Further Delay > Value Gained by Deferring Decision
  - Decisions should be made when the right amount of information is available
- Solutions
  - Incremental Decision Making
  - Change-Tolerant Designs
  - Set Based Designs
    - Pursue multiple options / designs concurrently
    - Review frequently
    - Finalize decisions as late as possible

Deliver Fast

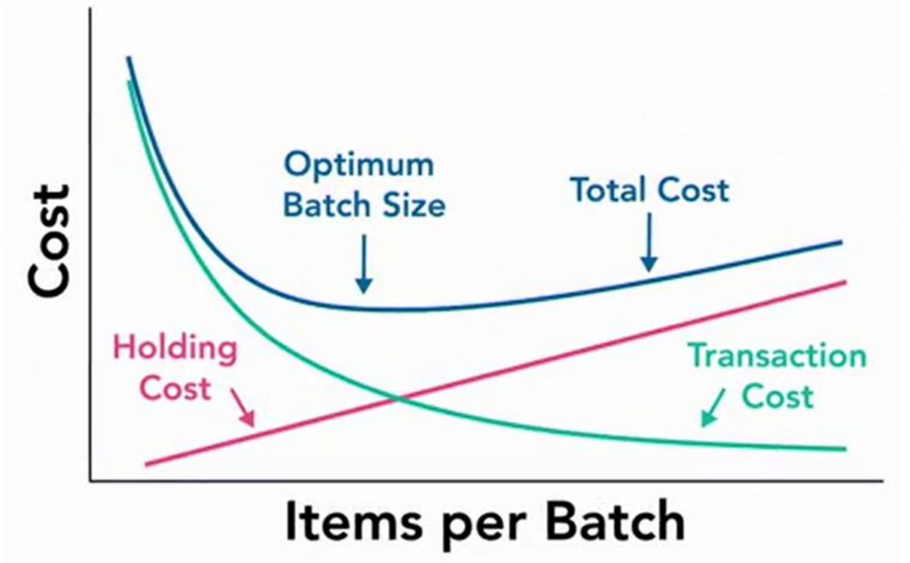
- Implement a Kanban Board
  - Visualize your work
    - Workflow steps represented by individual columns
    - Work artifacts represented by cards moving across columns
  - Limit Work In Process (WIP)
  - Monitor and manage flow
  - Make policies explicit
  - Implement regular feedback loops
  - Improve collaboratively

Deliver Fast

- Reduce WIP
  - Board-level / team WIP limit
  - Per-person WIP limit
  - Per-work-type limit
  - Per-column WIP limit

Deliver Fast

- Delivery frequent, small batches
  - Holding cost
    - Money not realized until delivered
    - Cost to manage batched work
    - Delays due to re-work or last minute testing
  - Transaction cost
    - Actual work required for delivery
    - Monetary cost? Labor cost?
    - Cost to validate deployment with customer
    - Cost to customer to consume / validate new release



Deliver Fast

- Optimize Flow Efficiency
  - Minimize waiting time between work activities
- Optimize Resource Efficiency
- Deliver frequent, small batches
  - Holding cost
    - Money not realized until delivered
    - Cost to manage batched work
    - Delays due to re-work or last minute testing
  - Transaction cost
    - Actual work required for delivery
    - Monetary cost? Labor cost?
    - Cost to validate deployment with customer
    - Cost to customer to consume / validate new release

Respect People

- Respect the Employee
  - Engage in Self-Direction
    - Kanban
    - Andon - highlight issues or exceptions
    - Dashboard
  - Foster Psychological Safety
    - Assume good intent
    - Replace blame with curiosity
    - Ensure all personality types can contribute
    - Reset opinion of conflict
    - Ask feedback

Optimize the Whole

- Focus on the whole process, not just the development cycle
  - Including
    - Ideation
    - Development
    - Delivery
    - Maintenance
    - Operation

Optimize the Whole

- Value Stream = End-to-End Process
  - Boundary - First Step and Last Step
  - Each step in between
  - Time spent in each step (Value Added Time)
    - Time work is active vs. waiting
  - How work moves from one step to the next
  - Time spent to move work to next step (Non-Value Added Time)