Introduction To Kanban In Software Engineering

# Why Something New

* Pendulum from waterfall to agile swung too far in some cases, and in the wrong direction in other cases
* Artificial time boxes instead of natural rhythm
* Batches of WIP still too large. Cost of inventory in knowledge system: knowledge transfer, relearning, quality issues, hiding systemic problems, lack of feedback, building up broken parts on each other, etc
* “Stop starting and start finishing”
* “Frozen” iteration / sprint vs. real-time changing priorities of customer

# Lean Thinking

* Whole value stream, not just “development” team
* Expose problems @ systemic level
* Reduce lead time
* Increase quality
* List principles of lean (probably use Poppendieck’s list)
* Flow of value
* Defect prevention
* Inventory in product development is logical / knowledge, not physical parts
* Optimize the whole, not the individual step or resource utilization
* Principled & foundations of agile are found in Lean

# Lean Tools

* Value stream maps
* Heijunka
* PDCA
* 5 why’s
* Zero defect (andon / jidoka)
* Pull systems

# Pull Systems: Kanban

* Recognize / formalize capacity of system to get things done
* A push system schedules work. A pull system authorizes work based on capacity to get work done
* Balancing demand against capacity
* Balancing capacity against demand
* Kanban is process control, not process for adding value to the WIP
* Facilitate continuous improvement by exposing systemic issues
* Grocery store example of origin of Kanban
* Work to protect current WIP from being ejected from system or stalled due to higher priorities, while giving customer ample opportunity to re-prioritize backlog

# How To Get Started

1. Visualize your process
2. Limit WIP
3. Pull, Don’t Push
4. Monitor & Improve

# Step 1: Visualize your process

* Facilitates “whole team” mentality
* Start with As much of the value stream as you can influence, expand from there
* Start with scrum / xp / whatever task board, expand from there
* Start by showing current WIP for entire system
* Creates open, transparent, end to end process
* Facilitate communication and collaboration
* Your system is different. Kanban won’t impose a process on you
* Handling new development vs. maintenance

# Step 2: Limit WIP

* Work to capacity of the system
* Identifying queues and queue sizing
* Limit WIP by: resources, stage/step, cycle times, risk, work item size, etc
* Get things done and get feedback faster
* Be productive
* Scrum/XP limit WIP by schedule. Kanban limits WIP by capacity

# Step 3: Pull, don’t push

* A kanban is a signal that capacity is available, and work should be done
* Limited WIP facilitates pull mechanics
* Modeling kanban signals: implicit vs. implicit
* Run a demo of a pull system to help people see and understand it

# Step 4: Monitor & Improve

* The mechanics of pull are necessary but not sufficient
* Identify bottlenecks and resource starvation
* Leading indicators and proactive issue prevention, instead of reactive issue handling
* What to do when the system stalls or a resource does not have any kanban to work from
* Expose systemic issues in bright light, forcing issues to be dealt with

# Metrics & Beyond

* Cycle time, lead time, WIP limits / totals
* Focus on lead time, not production rate
* Kanban & cadence: decoupling planning, work, retrospectives, delivery. Allow them to find their own rhythm for the customer/team/process
* Heijunka & designing the work to reduce variation
* Branch per feature
* Release per feature
* CI per branch
* Cumulative flow diagrams