Lean Software & Kanban

Software Engineering II
WS 2017/18

Christoph Matthies
Christoph.matthies@hpi.de

Enterprise Platform and Integration Concepts
Lean Software’s Main Idea

Reduce Waste

Waste
- Anything not delivered to the customer
- Artifacts that do not deliver a business value

Lean software inspired by
- Toyota’s “lean manufacturing” industrial production
- Just-in-time production
Lean Software’s Principles

1. **Eliminate Waste**
   - Anything not delivering business value
   - E.g. requirements documents, partially done work, rarely used features (bloat), bugs, task switching, waiting

2. **Amplify learning**
   - “Try-it, test-it, fix-it” rather than “do it right the first time”
   - Short iteration cycles

3. **Decide as late as possible**
   - Avoid up front design decisions, make choices when information is available

4. **Deliver as fast as possible**
   - Working system at every iteration, fast feedback cycle
Lean Software’s Principles

5. **Empower the team**
   - Motivate the team, self-organization
   - “Find good people and let them do their own job”

6. **Build integrity in**
   - Maintain the consistency of a system’s design
   - E.g. through refactoring, automated tests, complete build system

7. **See the whole**
   - Focus on overall progress of the project
   - Strong common sense
Lean Software Summary

- **Idea:** Software development can benefit from industrial production recipes
- However, software has no production, only design
- “Lean” can be seen as more philosophy than method
- Reminder to look out for waste of any kind
Kanban

(看板)
At full capacity, there is little throughput (flow is constricted by bottlenecks)
Kanban’s Main Idea

- Kanban: literally “signboard” or “billboard” in Japanese
- Inspired by Toyota
- Visual process-management approach (“Kanban boards”)
- “Stop Starting, Start Finishing”
- Ensure just-in-time production
Core Kanban Practices

1. **Limit work in progress (Kanban Limits)**
   - Limit amount of tickets per column
   - Focus on most productive task for the project
   - Pull work from previous columns
   - Reduce context switching (waste)

2. **Visualize**
   - Shared Kanban-Board with process steps as columns
   - Requirements (tasks, user stories,...) travel as notes from left to right.

3. **Manage flow**
   - Measure length of queue, average cycle time and throughput
   - Identify bottlenecks and allow planning
4. **Make policies explicit**
   - Create explicit shared understanding of rules and assumptions
   - E.g. what columns mean, Definition of Done, which ticket to pull next

5. **Implement feedback loops**
   - Process of continuous improvement (“kaizen” in Japanese)
   - Don’t wait for feedback, build it into the process

6. **Improve collaboratively, evolve experimentally**
   - Try things out, evaluate
Flow: from backlog to delivery

Kanban limits

4 4

Pull
Push vs. Pull Systems

- How a team handles the inventory & scheduling of work items
- **Push production**: based on forecasted demand, schedule
- **Pull production**: based on consumed products, take only what is needed, process immediately

- **Scrum** Sprint Planning: **Push**
  - Forecasted demand (business needs)
  - Estimated capacity of team

- **Kanban**: **Pull**
  - No need for planning, no queue to push into
  - Stories worked on based on actual demand and actual capacity

It can be argued that Scrum is a pull-based system as well, with work pulled in larger batches (the Sprint Backlog).
Subdivisions by subject / swim lanes
One day in Kanban Land

(A comic by Henrik Kniberg)
Metrics — Cumulative Flow Diagram

Lead Time: Time from ticket being placed on board to ticket delivered

Cycle Time: Time from starting work on ticket to finishing
Discussion

- Main differences between Scrum and Kanban?
- What could be the biggest challenges when employing Kanban?
- What would be a "Kanban Master's" tasks? Is this a necessary role?
- What domains/contexts is...
  - Kanban better suited for?
  - Scrum better suited for?
References