SSW-555: Agile Methods for Software Development

Lean Software Development

Dr. Richard Ens
Software Engineering
School of Systems and Enterprises

How Piggly Wiggly changed manufacturing and software development

Images from Wikipedia

Today's topics

Origin of Lean

Principles of Lean

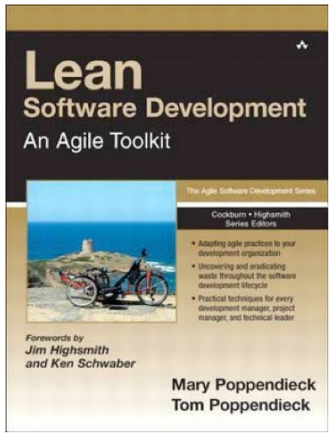
Practices of Lean

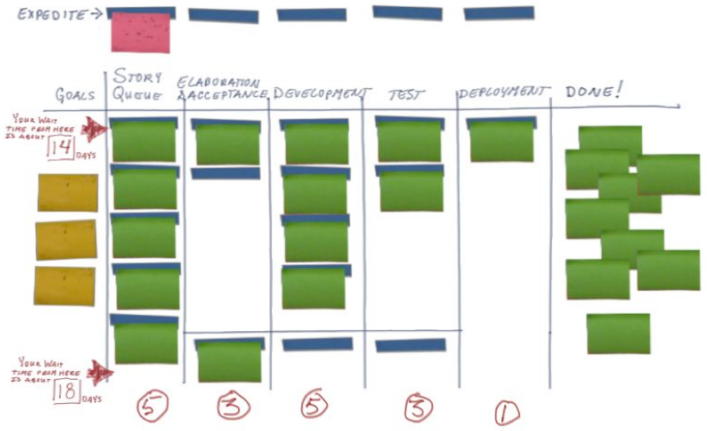
Kanban

Kanban principles

Kanban practices

Scrumban





Origins of Lean

Toyota developed a Lean Production System in the 1950s

- Response to mass manufacturing of Ford and GM
- Japan had a much smaller population and economy than the US
- Needed to be more agile, since volumes were lower
- Needed to shift quickly between different models
  - Ford/GM took many weeks to shift production to a new model
- Needed to eliminate waste

Poppendieck and others have applied principles of lean production to create Lean Software Development

Piggly Wiggly Story

Toyota visited Ford automotive plants in 1950

- Many of Ford's manufacturing methods would not be effective for Toyota
- More impressed with restocking mechanism at a local grocery store

Piggly Wiggly

- First self-service grocery store

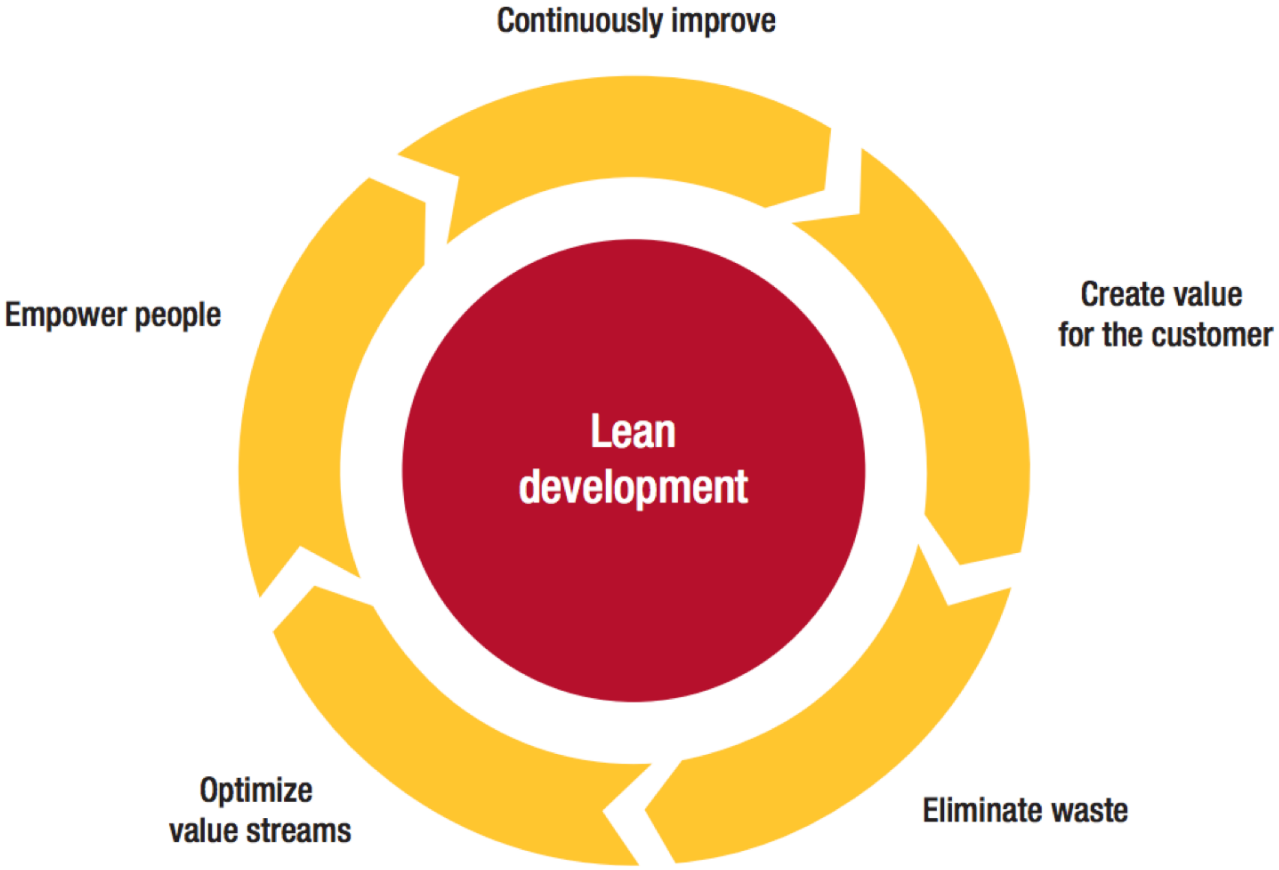
  Before this, customer asked employee behind the counter for each item

- Only reorder goods when customers had almost depleted current stock

Toyota realized that they could use this Just In Time (JIT) strategy for manufacturing cars

Images from Wikipedia

Lean Development



Toyota's 7 Principles of Lean

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

Principles

Toyota developed Lean for manufacturing, but how can we use these same principles for software development?

1. Eliminate waste

Anything that does not add value to the customer is waste

First, need to see waste, so may need to employ Value Stream Mapping

Examples of waste:

- Unneeded features
- Delay in development process, e.g. waiting for a meeting to integrate new code

2. Amplify learning

Everyone needs to learn and apply improvements as soon as possible

Reading and refactoring code help

Short iterations provide helpful feedback from customers, so both developers and customers learn

3. Decide as late as possible

Premature decisions may need to be undone later, which creates waste

Building only what is needed now avoids premature decision-making

Still need to do some planning when known options need to be considered

Defer decisions as long as possible to collect as much useful information as possible

4. Deliver as fast as possible

Just-In-Time production can be applied in software development

Allow teams to self-organize so that they can most effectively deliver what is needed

Quick feedback to/from customers is ideal

5. Empower the team

Developers should provide their own estimates of effort

Developers should choose their own process

Developers should choose their own tools

Management should facilitate, not dictate

6. Build integrity in

Invest time and effort to build a good product rather than providing a bad customer experience

Refactor whenever bad smells are detected

Test frequently to assure quality, don't wait until the end of the development process to integrate

Develop releases that provide value to the customer

7. See the whole

All staff need to be committed to the whole product

Don't isolate developers from the customer

Lean practices

Value stream mapping

Set-based development

Pull systems

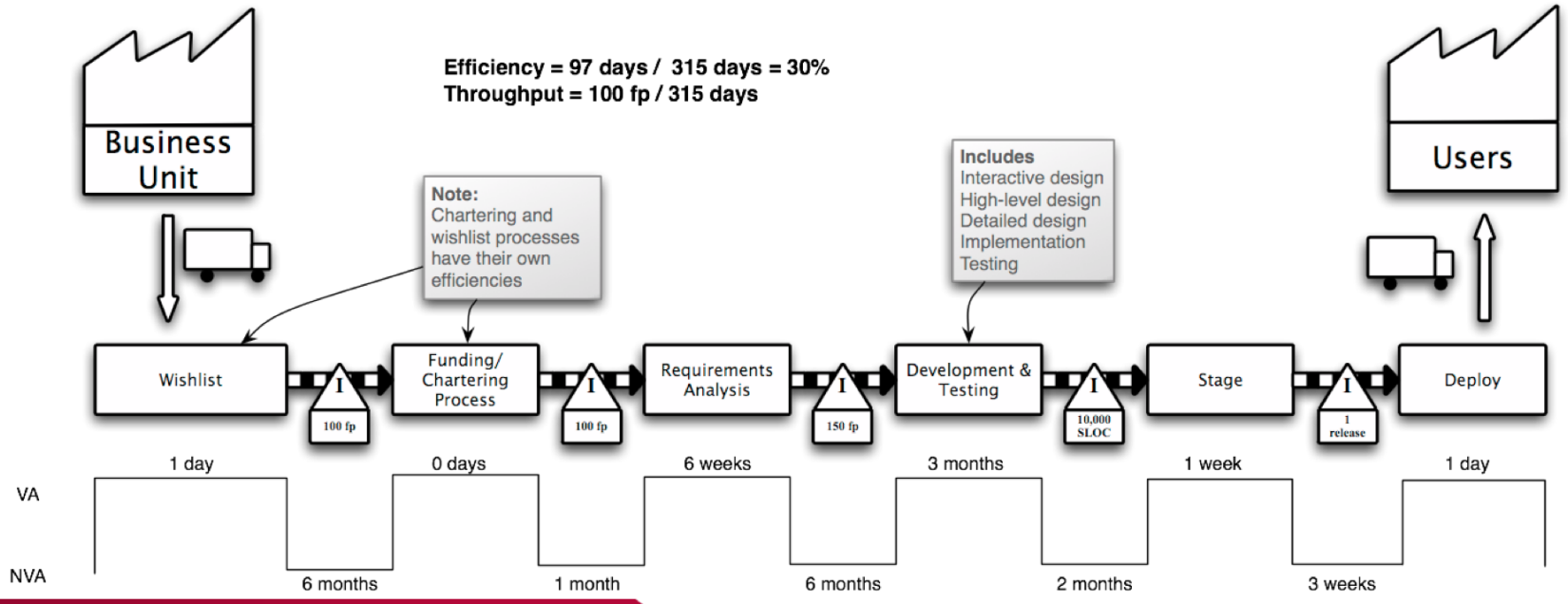
These principles were developed for manufacturing and adapted for software development

Value Stream Mapping

Diagram the flow of goods and/or information:

- Current map – shows the current situation
- Future map – shows the desired situation

Identify waste that should be removed from the current map and changes needed to establish the future map

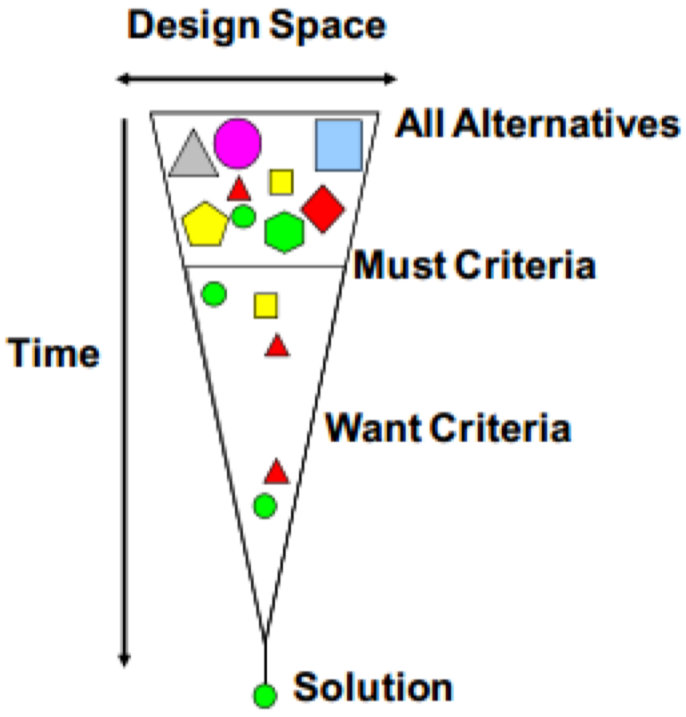


Set based development

Instead of choosing one design, consider several designs that will satisfy the customer requirements

Invest some time exploring all the alternatives, perhaps even implementing prototypes

Eliminate alternatives as you gain experience and feedback



Source: http://lean-consulting.co/training.html

Lean Summary: 7 Principles of Lean

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

Principles

Kanban *is one method to apply the Lean Principles*

Kanban Method

Anything that slows the software delivery pipeline is wasteful

Kanban helps to identify waste and optimize processes

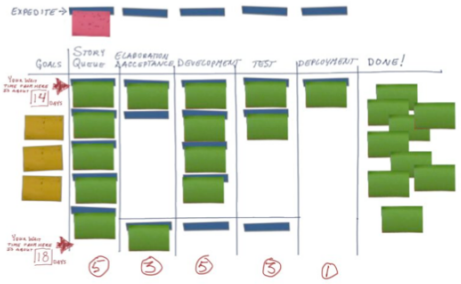
"Kanban is the science of not trying to do too much at once"

Stephen Palmer, 2012

Stop starting and start finishing

Kanban helps the team to prioritize work

Kanban focuses on process improvement



Kanban boards

Visual display of items at each stage of the process

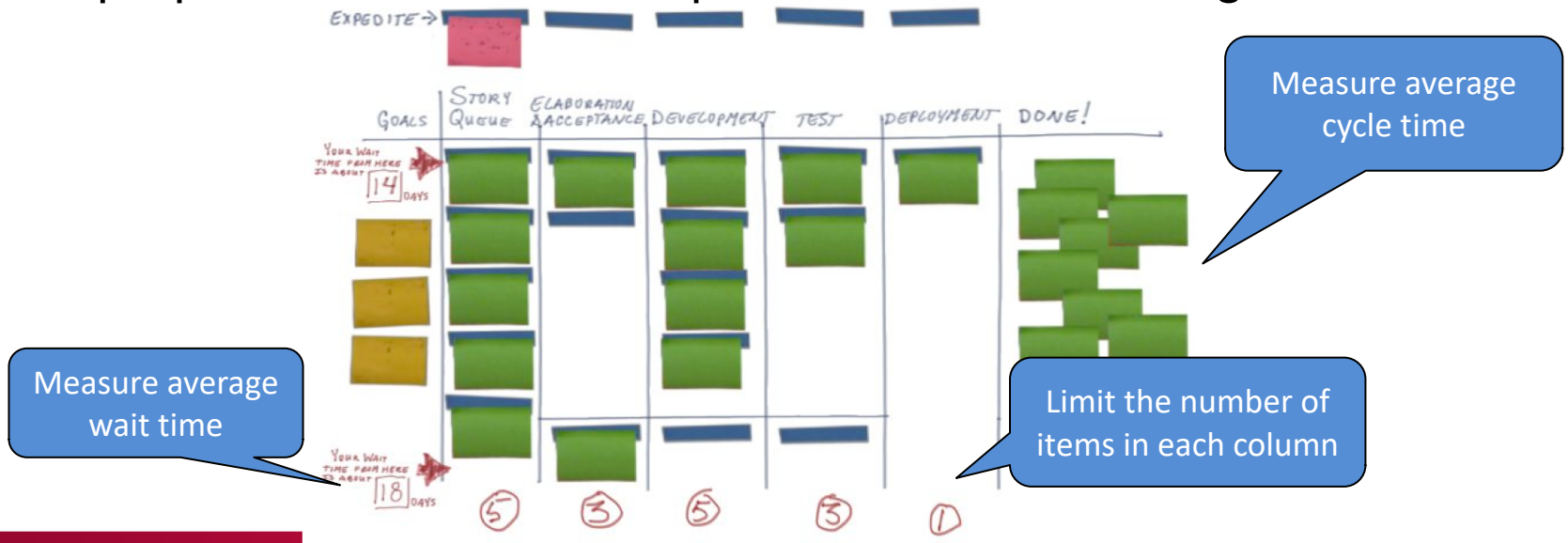
- ***Pull*** a task through the flow when capacity is available
- ***Don't push*** a task through the flow on demand

Prioritize to limit the number of items in each queue at any time

Focus on ***flow of value***: delivering items with little value quickly doesn't help

As a task completes, pull it to available spot in next column

Move people to work on different queues to eliminate backlog



3 steps to successful Kanban

1. Visualize the workflow

   Identify all of the steps in your process

   Measure work flowing through normally

2. Limit Work In Progress (WIP)

   Identify **bottlenecks**

3. Manage lead time

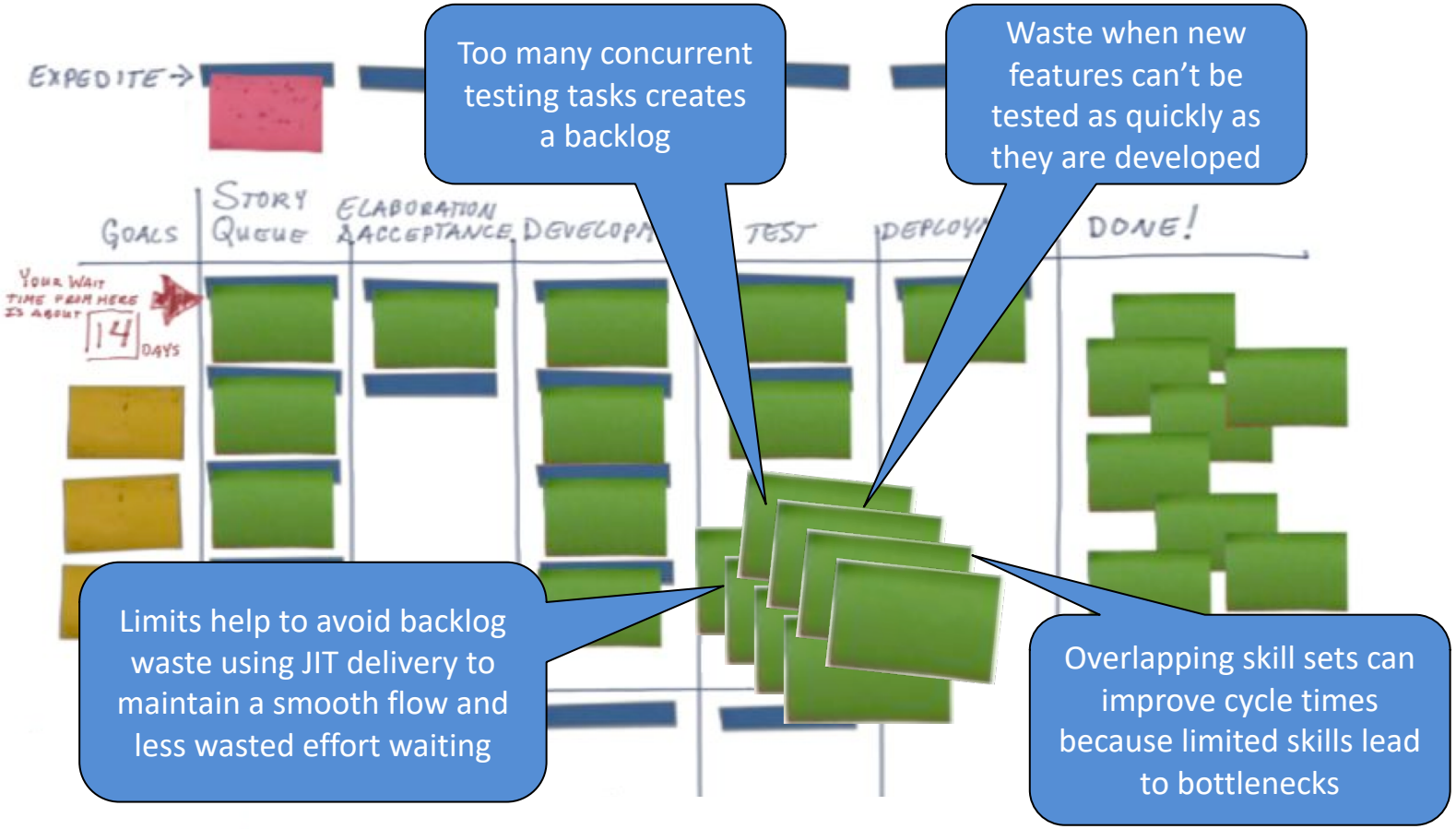
Lead time is the average time to pull an item from beginning of the process to the end

Cycle time is the time actually spent working on the item

Lead time includes wait time between stages

Wait time implies waste

Why limit WIP?



Focus on overall flow of work through the process rather than individual team member utilization

Kanban key principles

Principles

Foster leadership at all levels of the organization	Encourage all members of the organization to act as leaders, not just the bosses
Start with what you do now	Overlay Kanban on current process to evolve current process
Pursue incremental changes to existing process	Frequent, small changes are more effective than large changes
Respect current methodologies and roles	Keep what already works, but fix what's broken

Kanban practices

Visualize Workflow

Practices

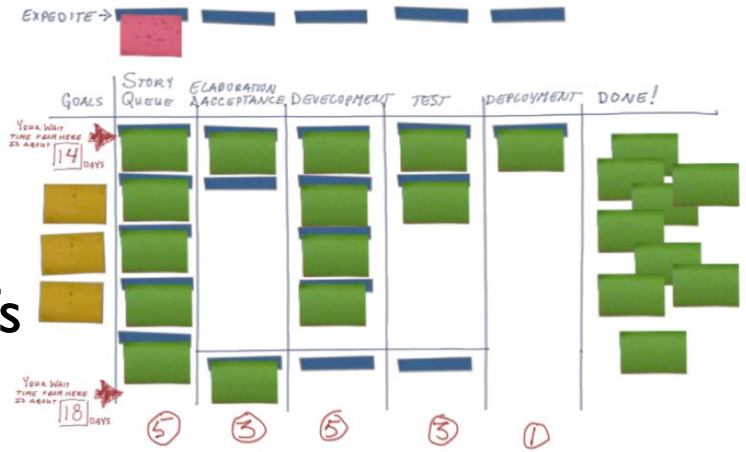
Must understand the current workflow to identify an optimal workflow

Include a column for each stage in the workflow

- Separate columns don't imply handoffs between people, just different tasks

Use a card for each incoming work request

Move cards from column to column as tasks are completed

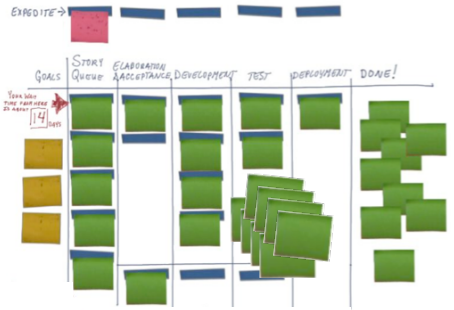


Kanban practices

Practices

Limit Work In Progress (WIP)

- Start new work only when time becomes available
- Limits amount of work that is impacted by changing priorities
- Restricts the flow of work to slowest step
  - Helps to identify and address bottlenecks
- Reduces cycle time and increases value delivered



Kanban practices

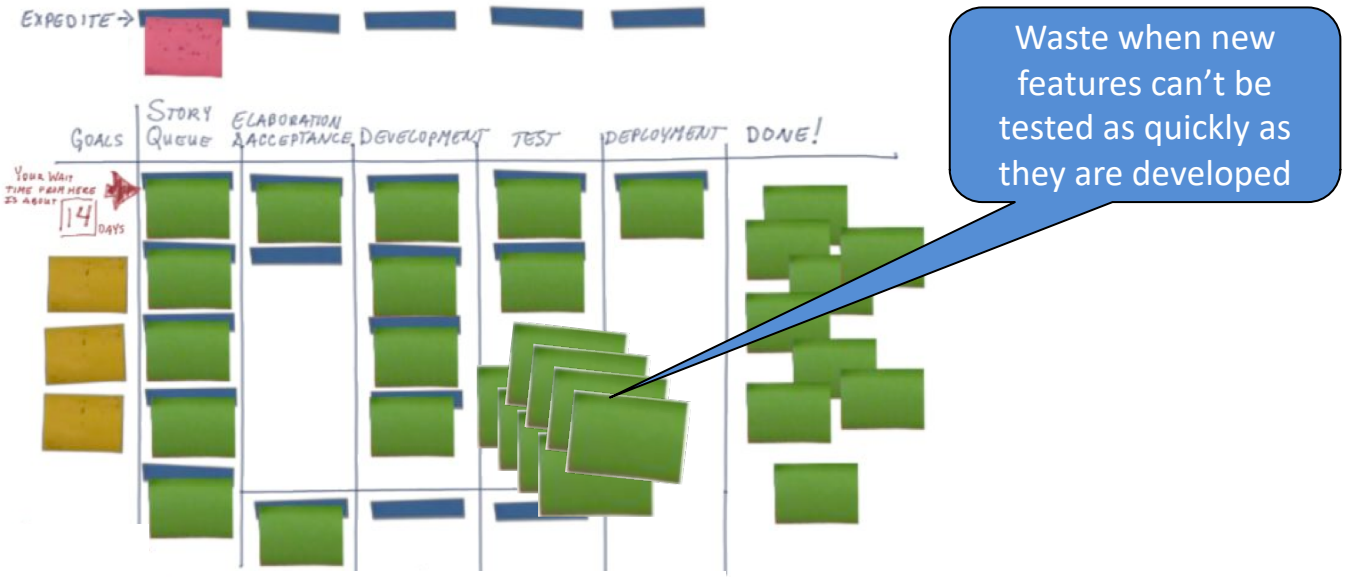
Practices

Manage Flow

Use flow to identify problems

Where do tasks stall?

Review the process and eliminate waste



Kanban practices

Practices

- Make process policies explicit
  - Process must be defined, published, and socialized
  - Get everyone on board, e.g. definition of done
  - Can't improve what you don't understand
- Improve collaboratively and continuously
  - Make small, incremental, evolutionary changes
  - Implement feedback loops and collect data

Kanban advantages

Easy to implement on top of many methods

Applies to many different types of organizations

- Especially helpful for managing frequent changes
- Software development
- Product development
- Customer support
- Manufacturing

Visually control the process

Focus on continuous delivery of value

Scrum vs Kanban Method

Scrum and Kanban may be complementary or competitors

Scrum focuses on *project management*

- What should we build?
- When will it be ready?
- Does it meet the customer's needs?

Kanban helps the team to prioritize work

Scrum vs Kanban

Scrum vs Kanban Method

Kanban	Scrum
No prescribed roles	Product Owner, Scrum Master, Developers
Continuous delivery	Time boxed sprints
Pull work through system	Pull work through in batches
Changes can be made at any time	Define sprint, then don't allow changes
Measure cycle time	Measure velocity
Ideal for high variability	Ideal for batch deliveries

Scrumban: Scrum + Kanban

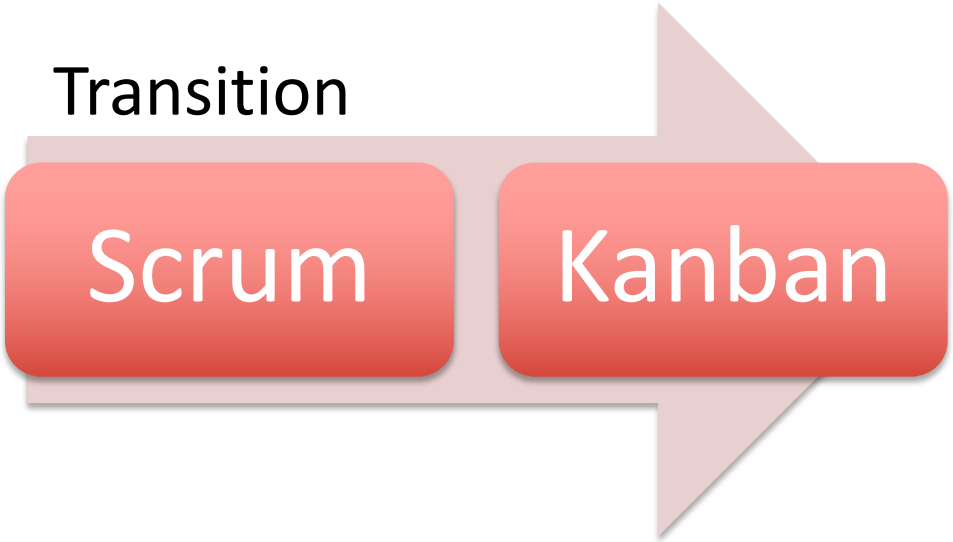
- Scrum features:
  - Scrum roles
    - Product Owner, Scrum Master, Developers
  - Scrum meetings
    - sprint planning, stand up, sprint review, sprint retrospective
  - Time-boxed deliverables
- Kanban features:
  - Just in time planning
  - Work In Progress limits

Scrumban alternatives

Include Kanban techniques in your Scrum process

$$Scrumban = Scrum + Kanban$$

Transition from Scrum to Kanban



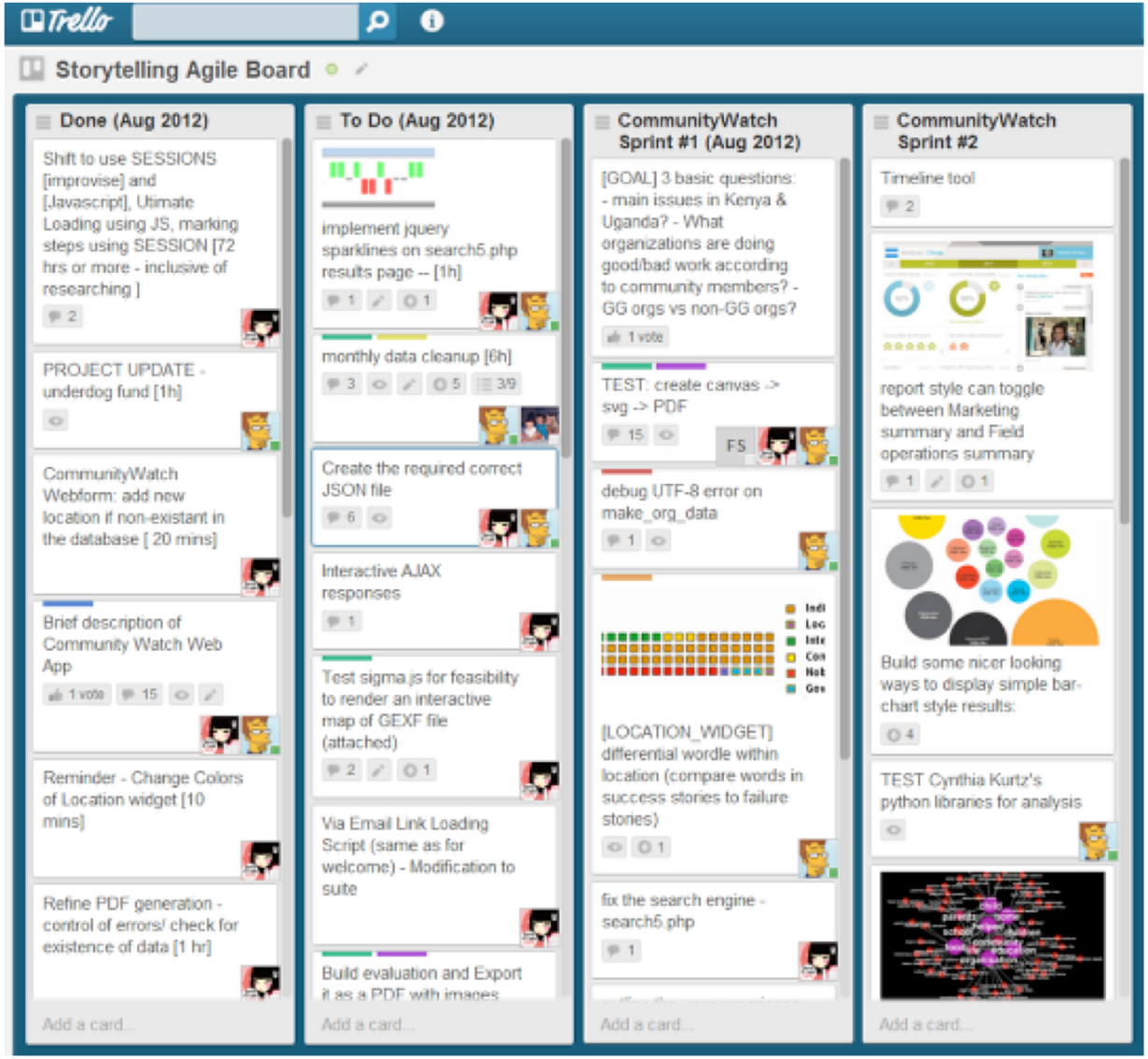
Why Scrumban?

Scrum planning may be inefficient and wasteful

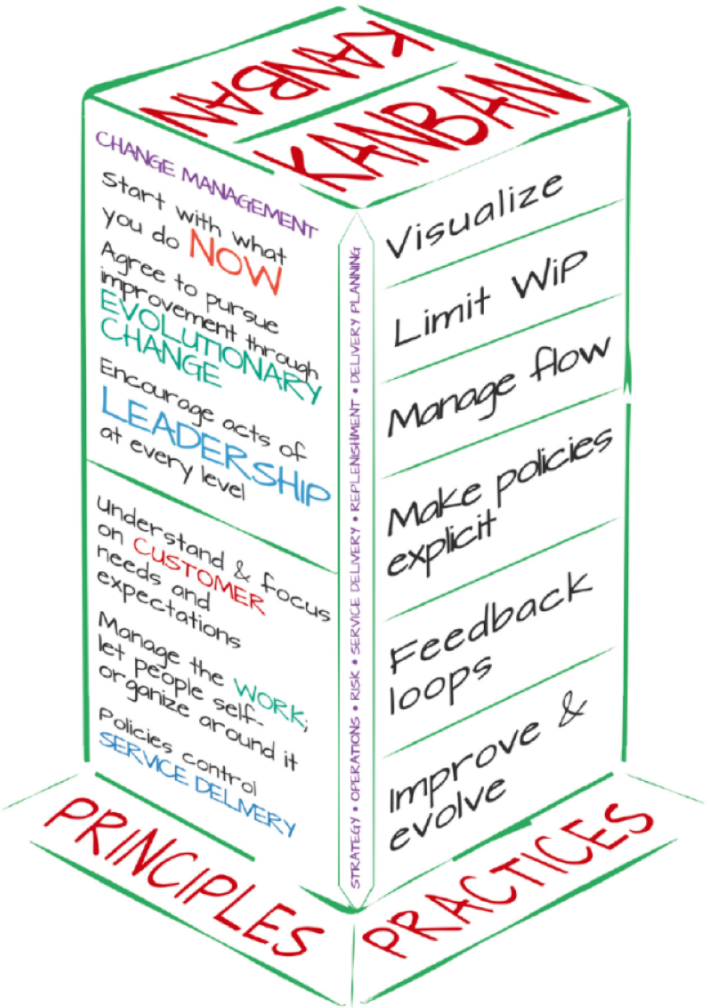
- Why estimate effort for user stories not delivered?
- Planning may take up too much time
  - Build code rather than talking about building code

May need more frequent releases than Scrum supports

Trello.com Kanban Boards



Kanban Summary



http://leankanban.com/project/what-is-km/

Questions?