STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

SSW-555: Agile Methods for Software Development

Lean Software Development

Prof. Jim Rowland
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

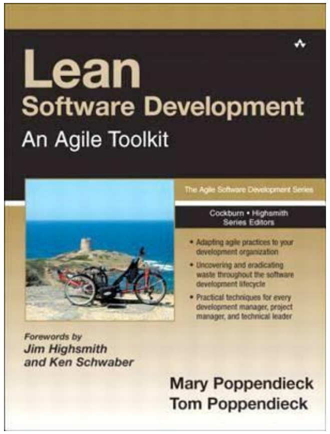
Practices of Lean

Kanban

Kanban principles

Kanban practices

Scrumban



Lean

Software Development

An Agile Toolkit

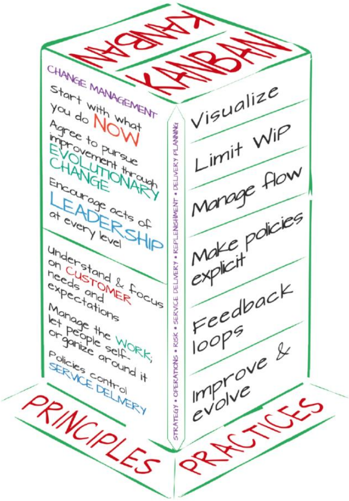
The Agile Software Development Series

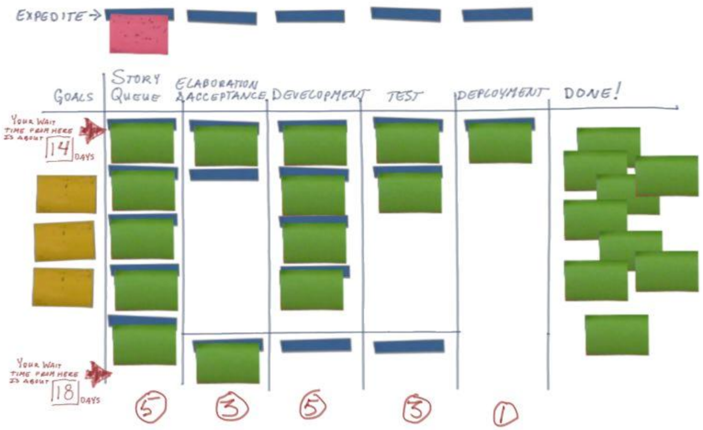
Cockburn • Highsmith
Series Editors

- Adapting agile practices to your development organization
- Uncovering and eradicating waste throughout the software development lifecycle
- Practical techniques for every development manager, project manager, and technical leader

Forewords by
Jim Highsmith
and Ken Schwaber

Mary Poppendieck
Tom Poppendieck


KANBAN
KANBAN
CHANGE MANAGEMENT
Start with what you do NOW
Agree to pursue improvement through EVOLUTIONARY CHANGE
Encourage acts of LEADERSHIP at every level
Understand & focus on CUSTOMER needs and expectations
Manage the WORK; let people self-organize around it
Policies control SERVICE DELIVERY
STRATEGY • OPERATIONS • RISK • SERVICE DELIVERY • REPLENISHMENT • DELIVERY PLANNING
Visualize
Limit WiP
Manage flow
Make policies explicit
Feedback loops
Improve & evolve
PRINCIPLES
PRACTICES



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

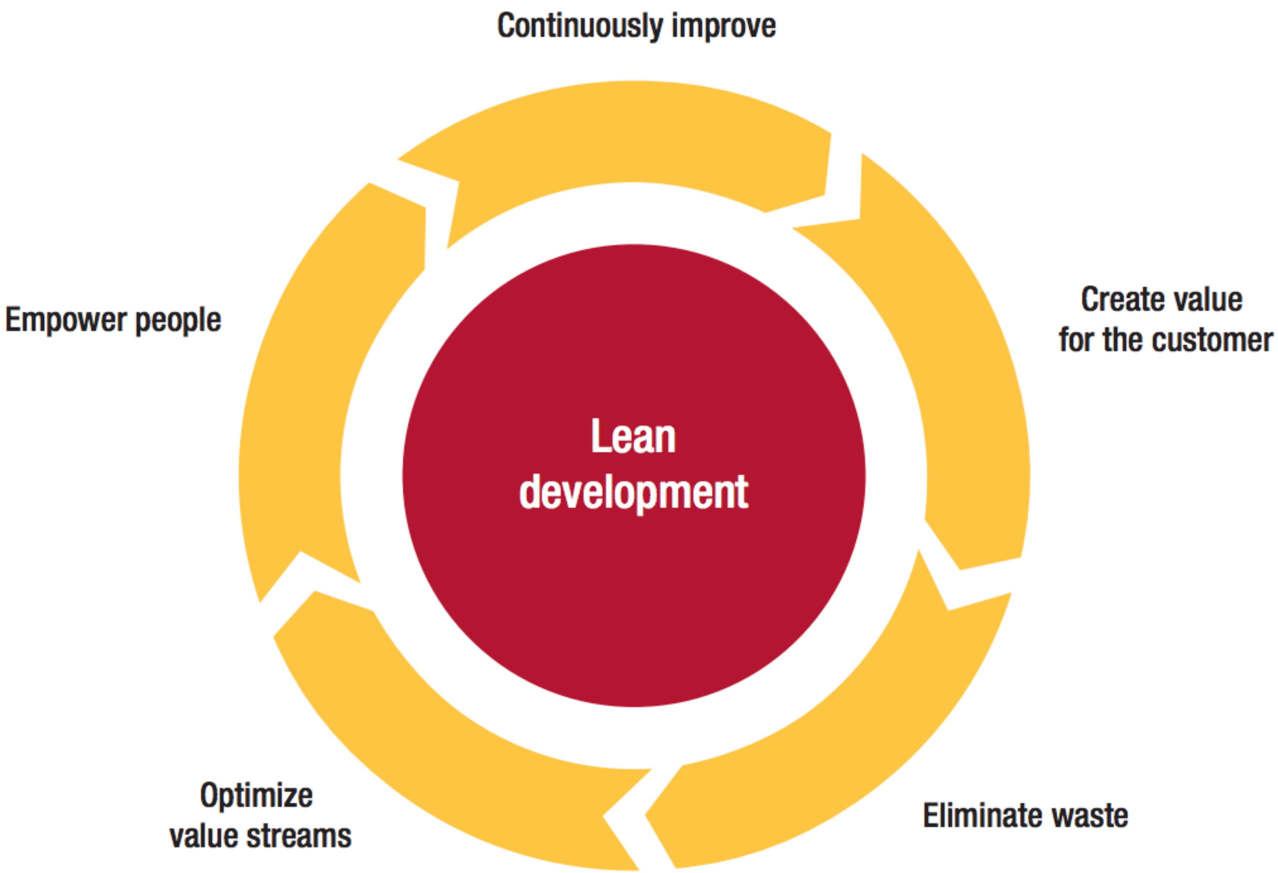
  Before this, customer asked employee behind the counter for each item

- Reorder goods only 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

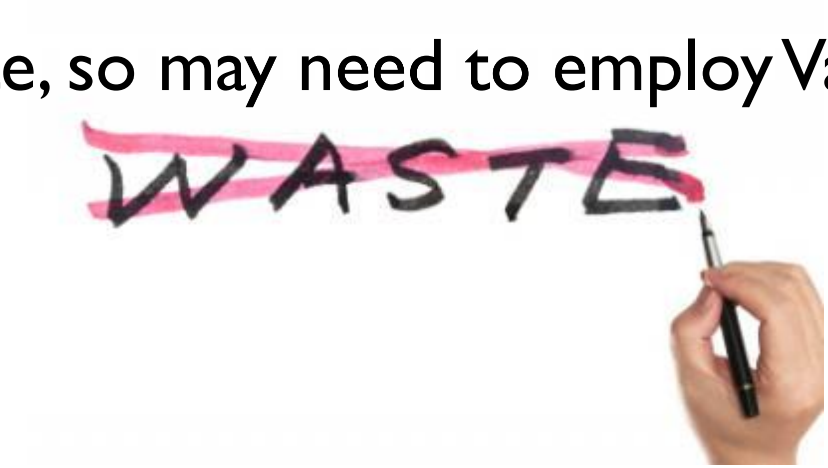
Principles

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

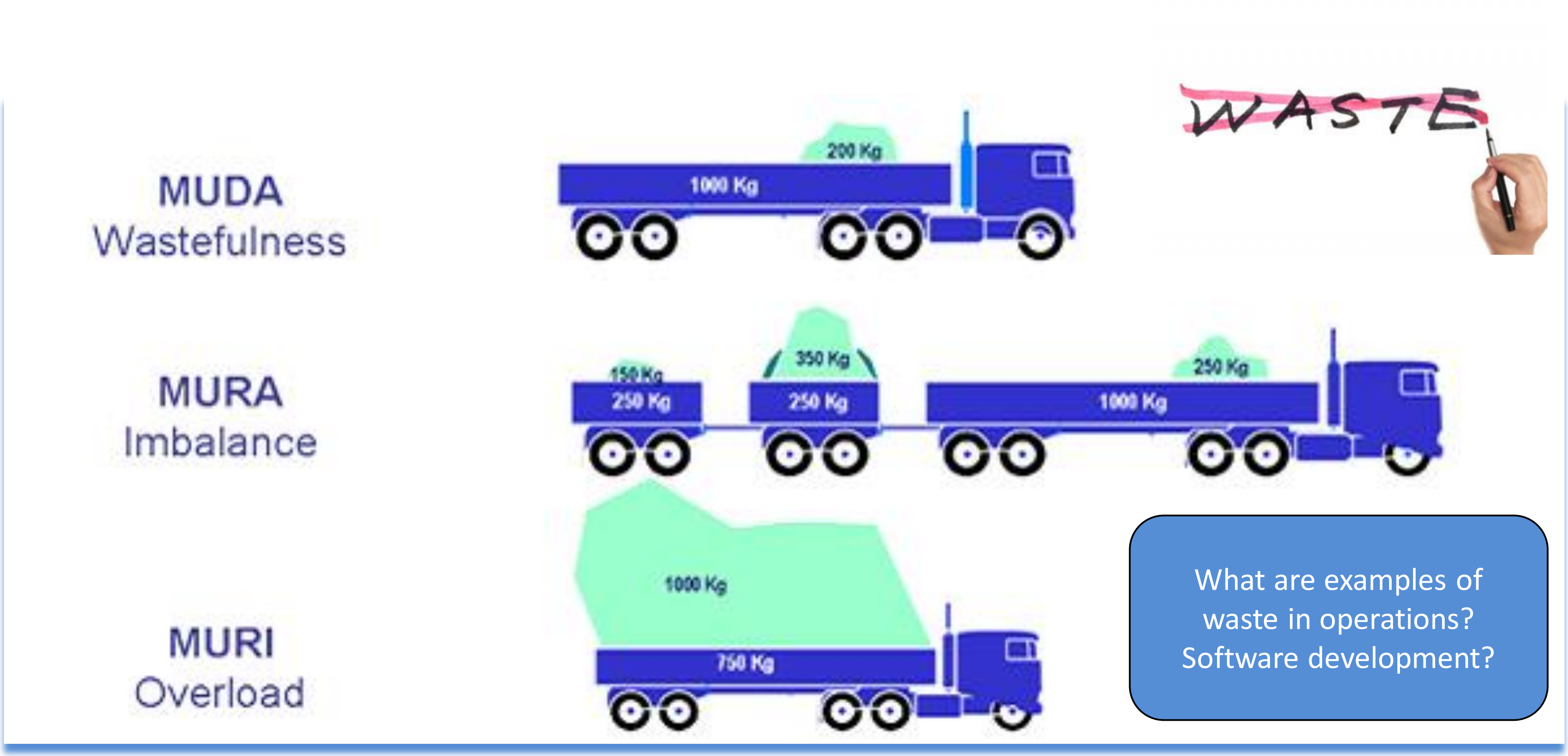
First, need to identify waste, so may need to employ Value Stream Mapping



Examples of waste:

- Unneeded features
- Delay in development process, e.g. waiting for a meeting to integrate new code
- Delays caused by manual processes

Three Types of Waste



What are examples of waste in operations? Software development?

Source: http://lean-management.pl/wcm/muda-muri-mura/

Seven Wastes of Lean Software

1. Partially done work
   Untested or undocumented code
2. Extra features
   Features not needed or wanted by the customer
3. Relearning
   Forgetting about decisions or not using all relevant knowledge
4. Handoffs
   Giving or getting information or work from others

Seven Wastes of Lean Software

5. Delays
   Waiting for someone or something
6. Task switching
   Multi-tasking on different tasks
7. Defects
   Anything not done correctly
   Bugs in code, process, features

2. Amplify learning

Everyone needs to learn and apply improvements as quickly 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

Kanban

- Pull systems (rather than push)

Explicit processes

Continuous improvement

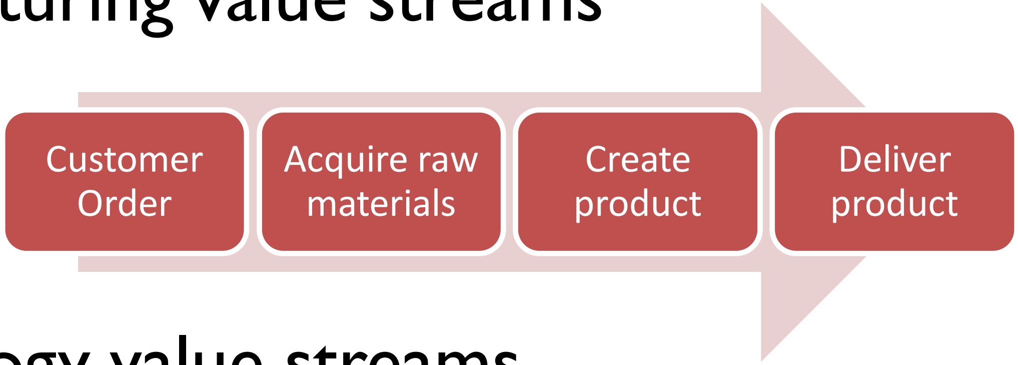
These principles were developed for manufacturing and adapted for software development

Lean Value Streams

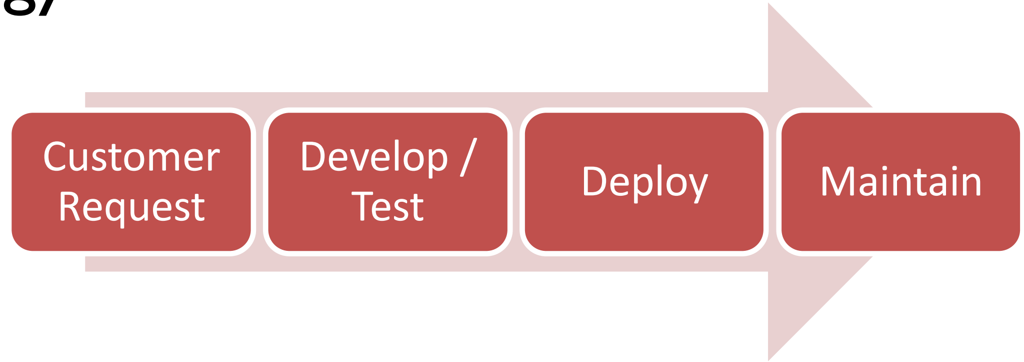
Value streams

Sequence of steps taken to deliver value to a customer

Manufacturing value streams



Technology value streams

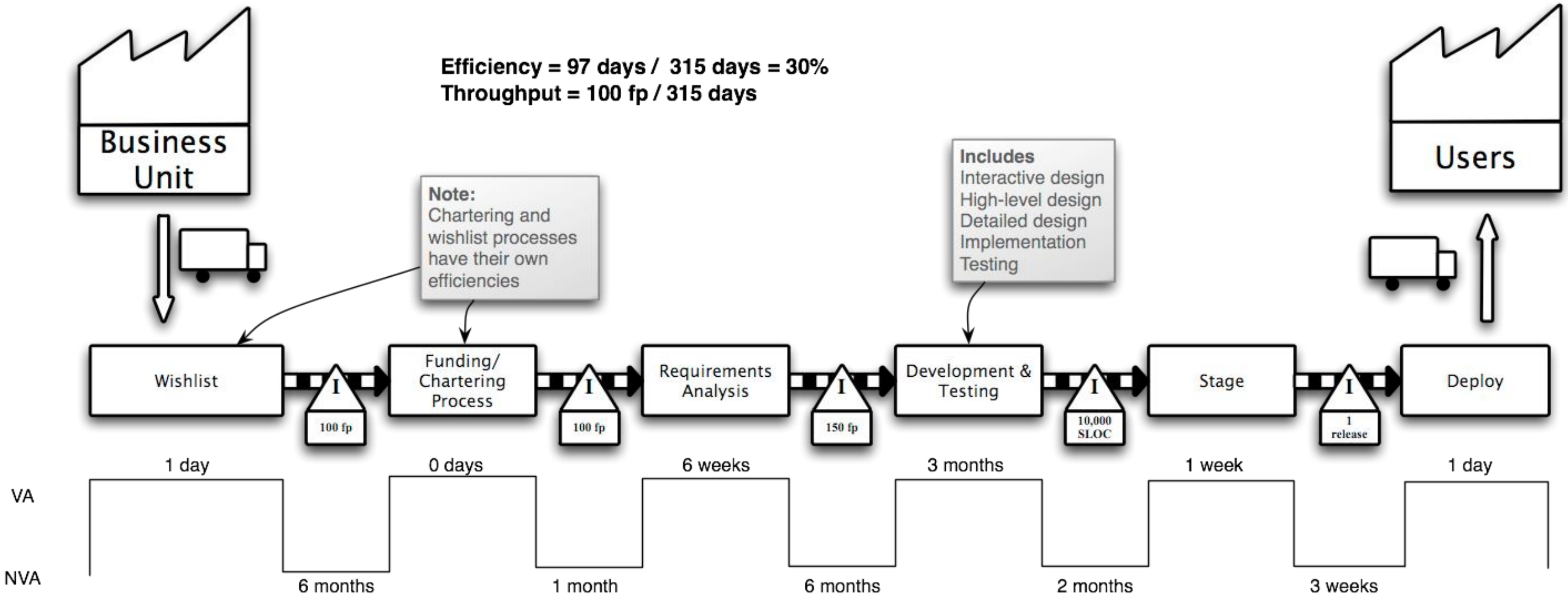


Value Stream Mapping

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



Efficiency = 97 days / 315 days = 30%
Throughput = 100 fp / 315 days

Business Unit

Users

Note:
Chartering and wishlist processes have their own efficiencies

Includes
Interactive design
High-level design
Detailed design
Implementation
Testing

Wishlist → I 100 fp → Funding/ Chartering Process → I 100 fp → Requirements Analysis → I 150 fp → Development & Testing → I 10,000 SLOC → Stage → I 1 release → Deploy

VA: 1 day, 0 days, 6 weeks, 3 months, 1 week, 1 day

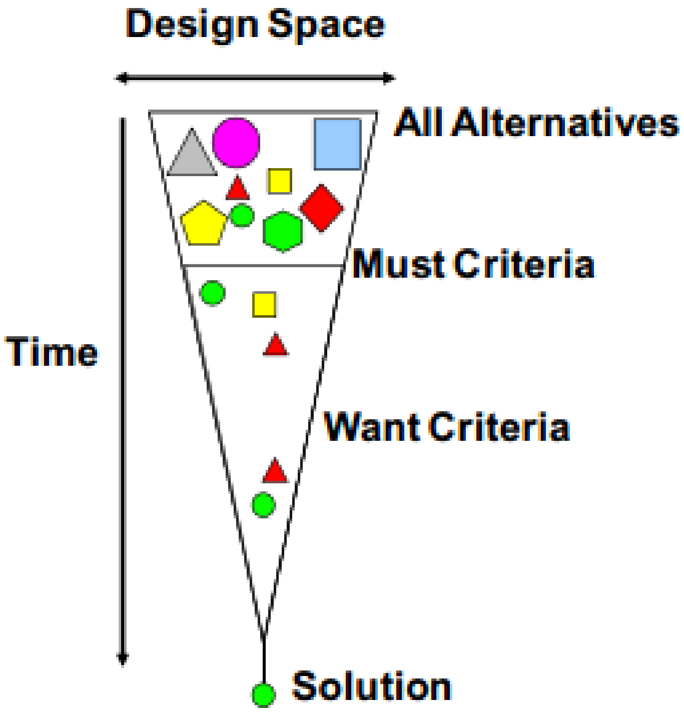
NVA: 6 months, 1 month, 6 months, 2 months, 3 weeks

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

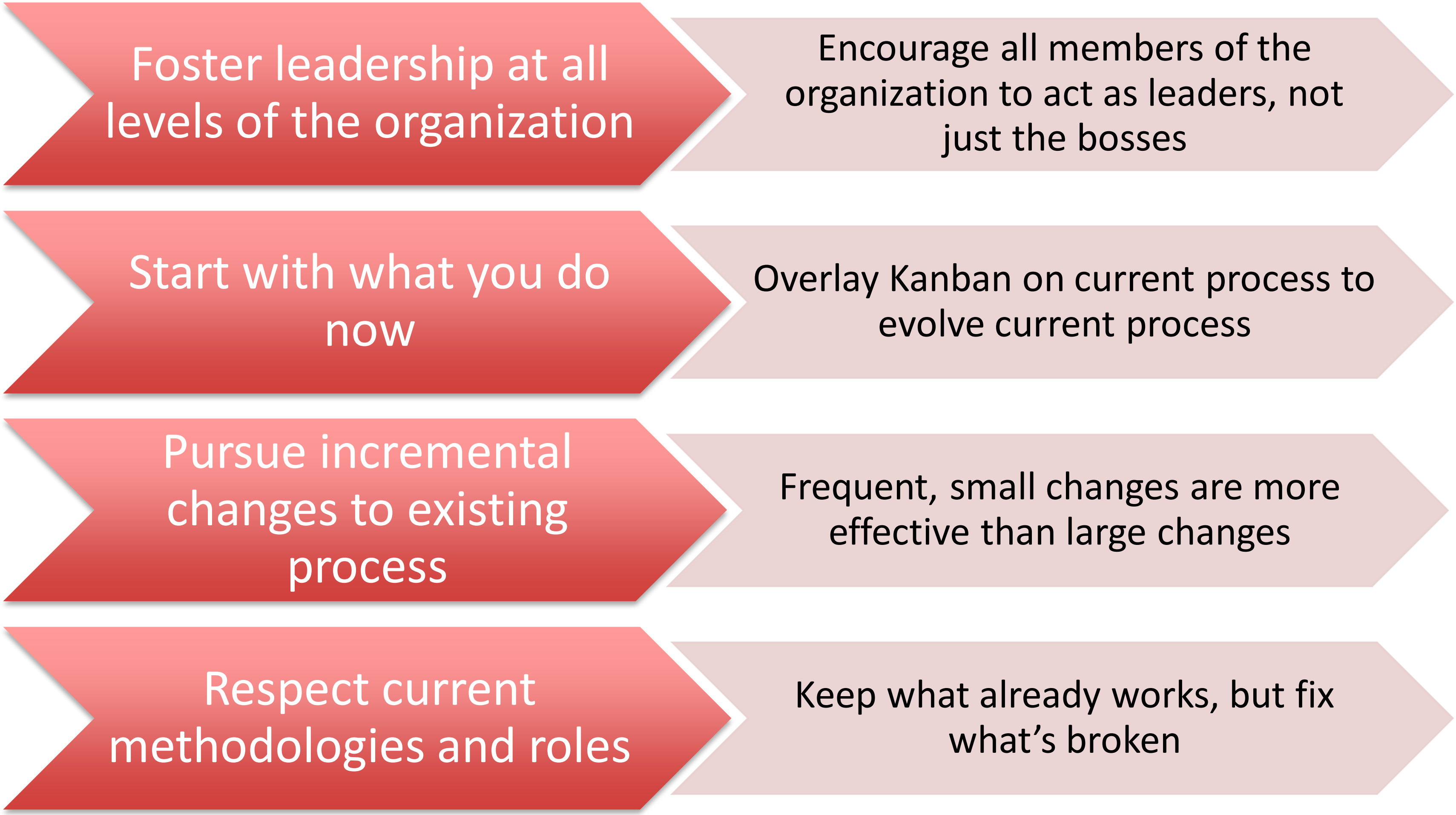
Eliminate alternatives as you gain experience and feedback



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

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 principles

Constant focus on continuous improvement

Embrace scientific thinking

Create flow and pull (not push)

Assure quality at source

Lead with humility

Respect every individual

Principles

Kanban Method

Anything that slows the software delivery pipeline is wasteful

Kanban helps to identify waste and optimize processes

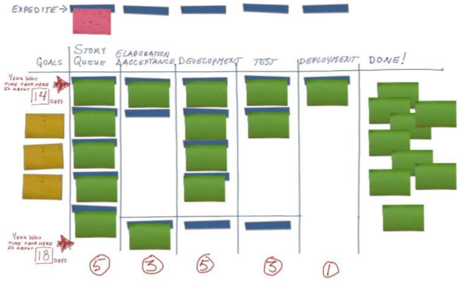
"Kanban is the science of not trying to do too much at once"

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

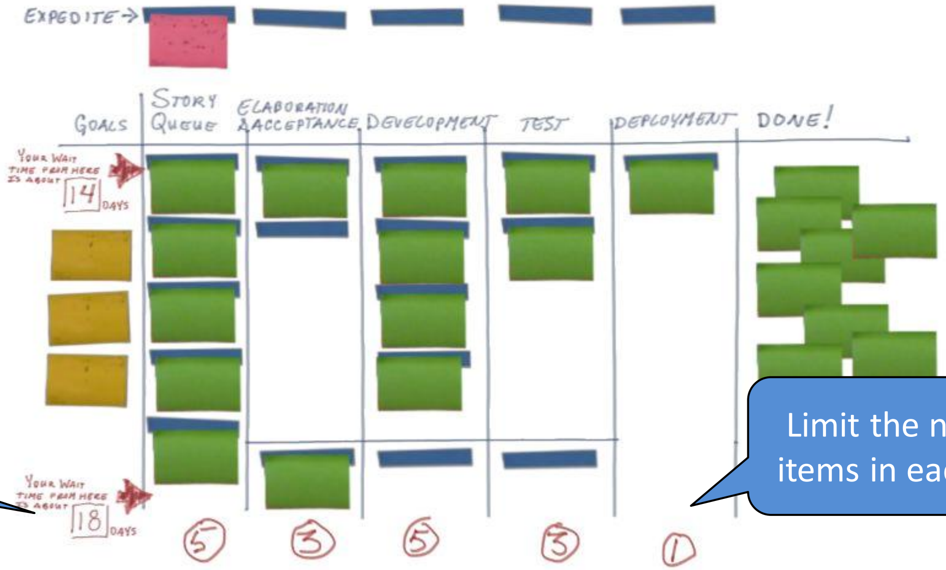
- ***Pull*** a task through the flow when capacity is available
- ***Don't push*** a task through the flow on demand

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



Measure average cycle time

Measure average wait time

Limit the number of items in each column

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

Kanban practices

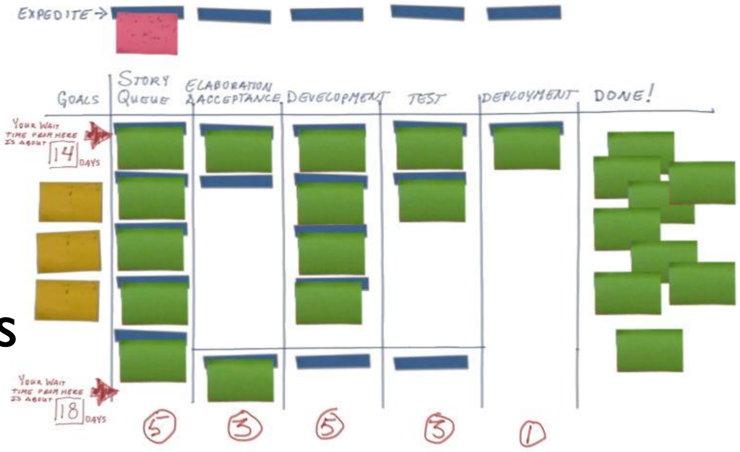
Practices

Visualize Workflow

Must understand the current workflow to identify an optimal workflow

Include a column for each stage in the workflow

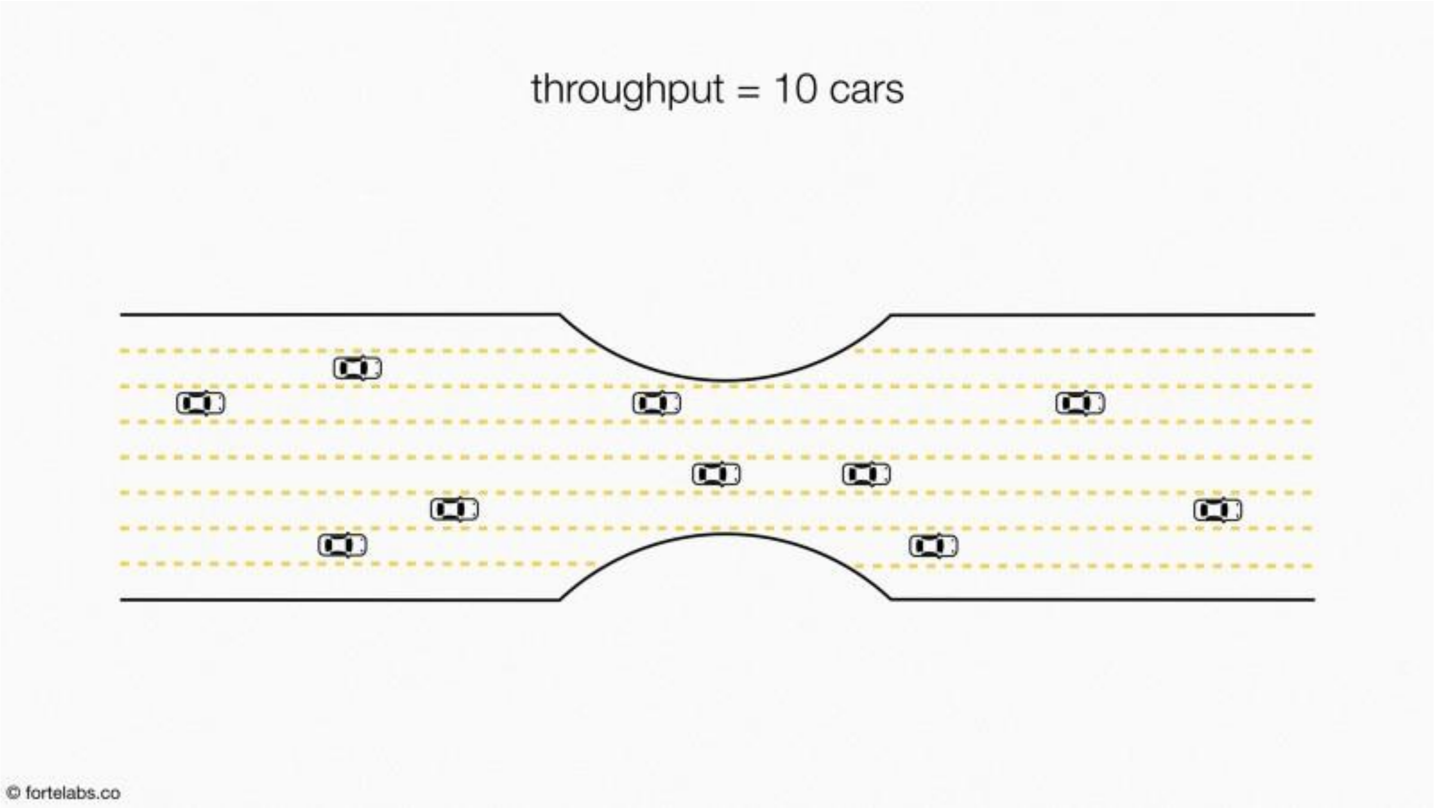
  Separate columns don't imply handoffs between people, just different tasks



Use a card for each incoming work request

Move cards from column to column as tasks are completed

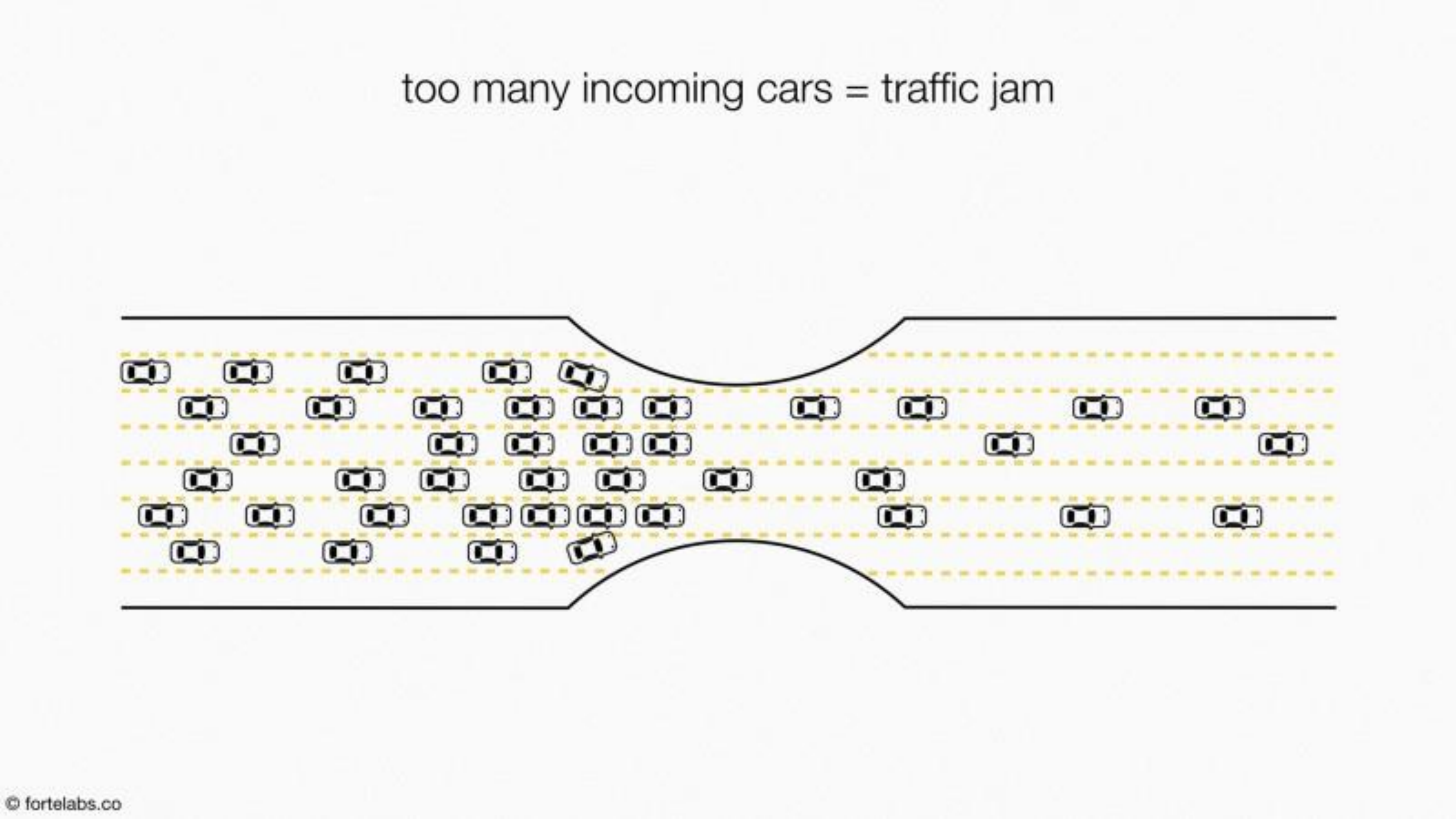
Limiting Work In Progress



© fortelabs.co

https://medium.com/praxis-blog/theory-of-constraints-104-balance-flow-not-capacity-60baa74ce9f4

Limiting Work In Progress

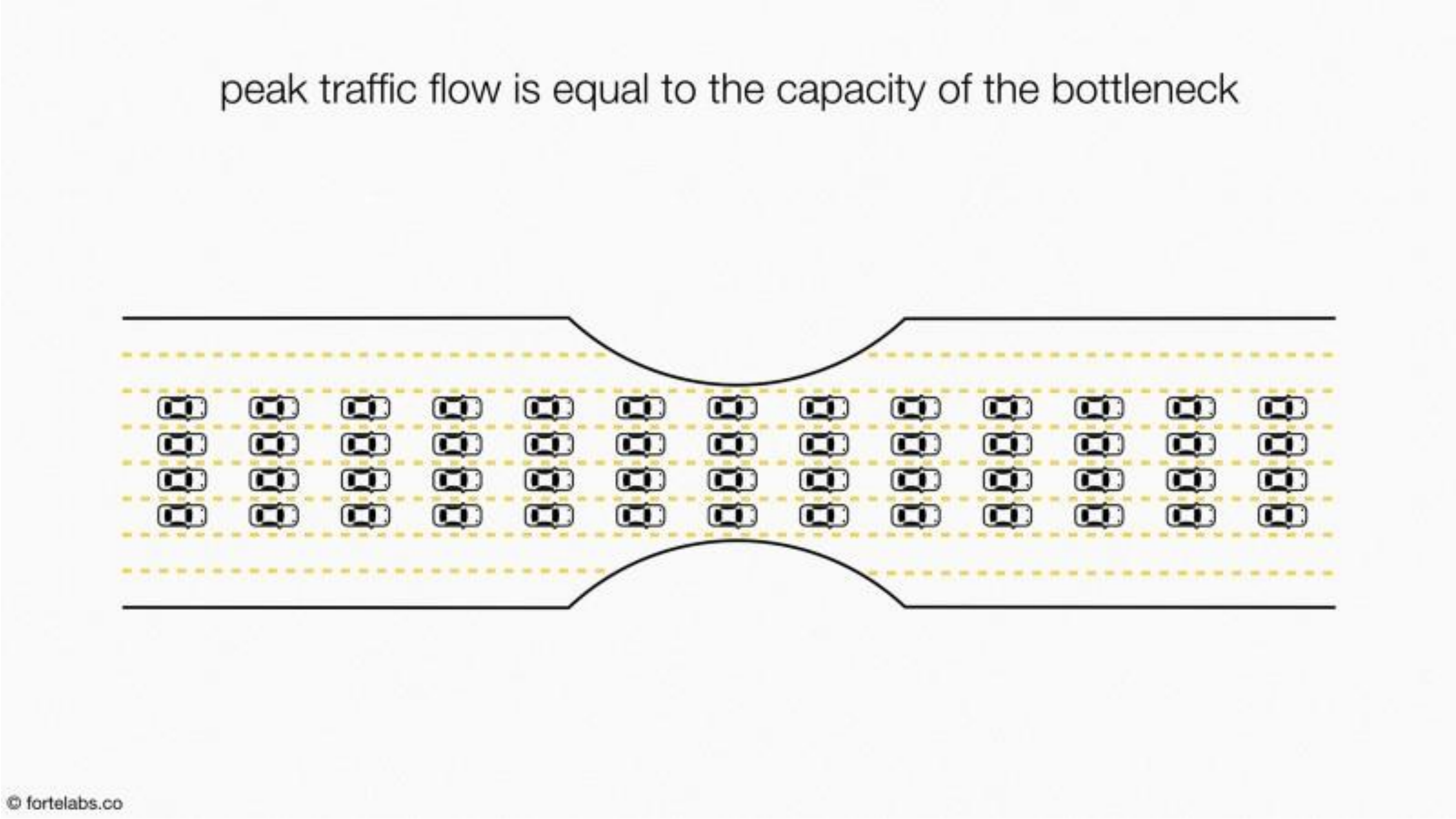


too many incoming cars = traffic jam

© fortelabs.co

https://medium.com/praxis-blog/theory-of-constraints-104-balance-flow-not-capacity-60baa74ce9f4

Limiting Work In Progress


peak traffic flow is equal to the capacity of the bottleneck

© fortelabs.co

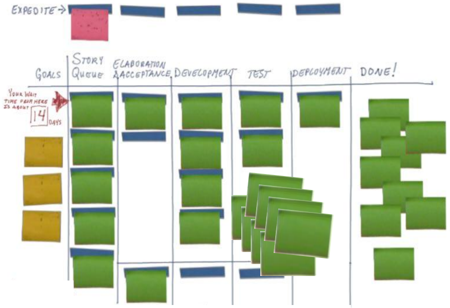
https://medium.com/praxis-blog/theory-of-constraints-104-balance-flow-not-capacity-60baa74ce9f4

Kanban practices

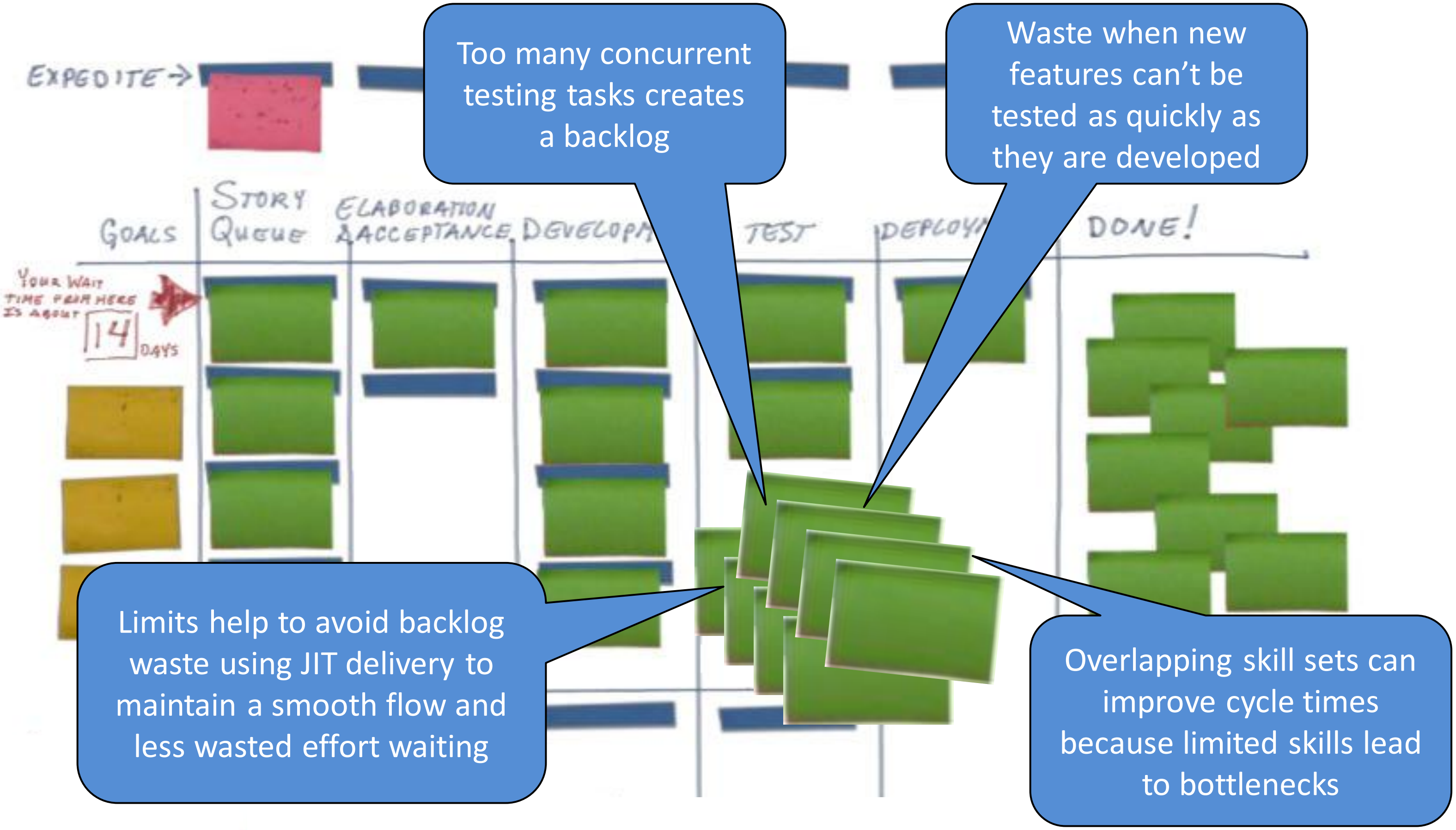
Practices

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



Why limit WIP?



Too many concurrent testing tasks creates a backlog

Waste when new features can't be tested as quickly as they are developed

Limits help to avoid backlog waste using JIT delivery to maintain a smooth flow and less wasted effort waiting

Overlapping skill sets can improve cycle times because limited skills lead to bottlenecks

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

Kanban practices

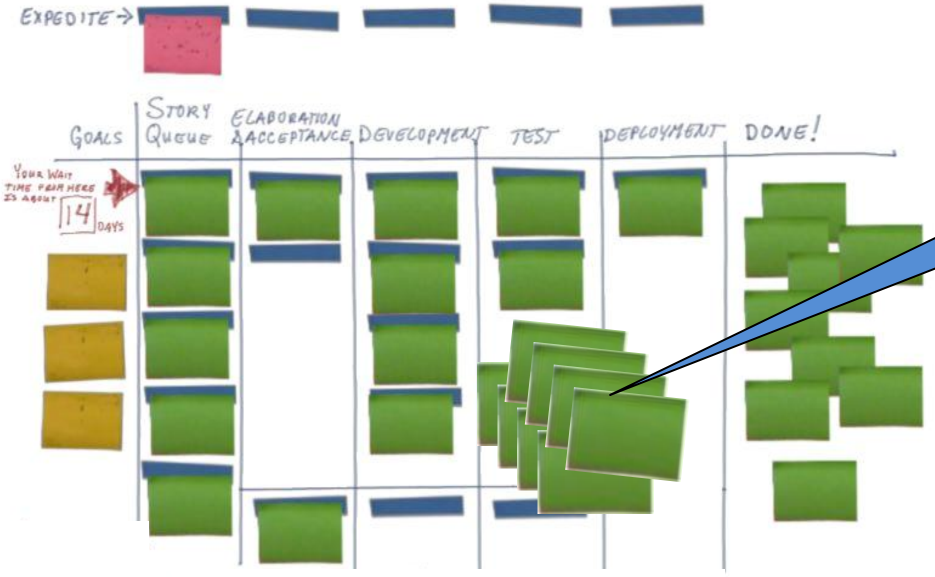
Practices

Manage Flow

Use flow to identify and eliminate problems

Where do tasks stall?

Review the process and eliminate waste



Waste when new features can’t be tested as quickly as they are developed

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

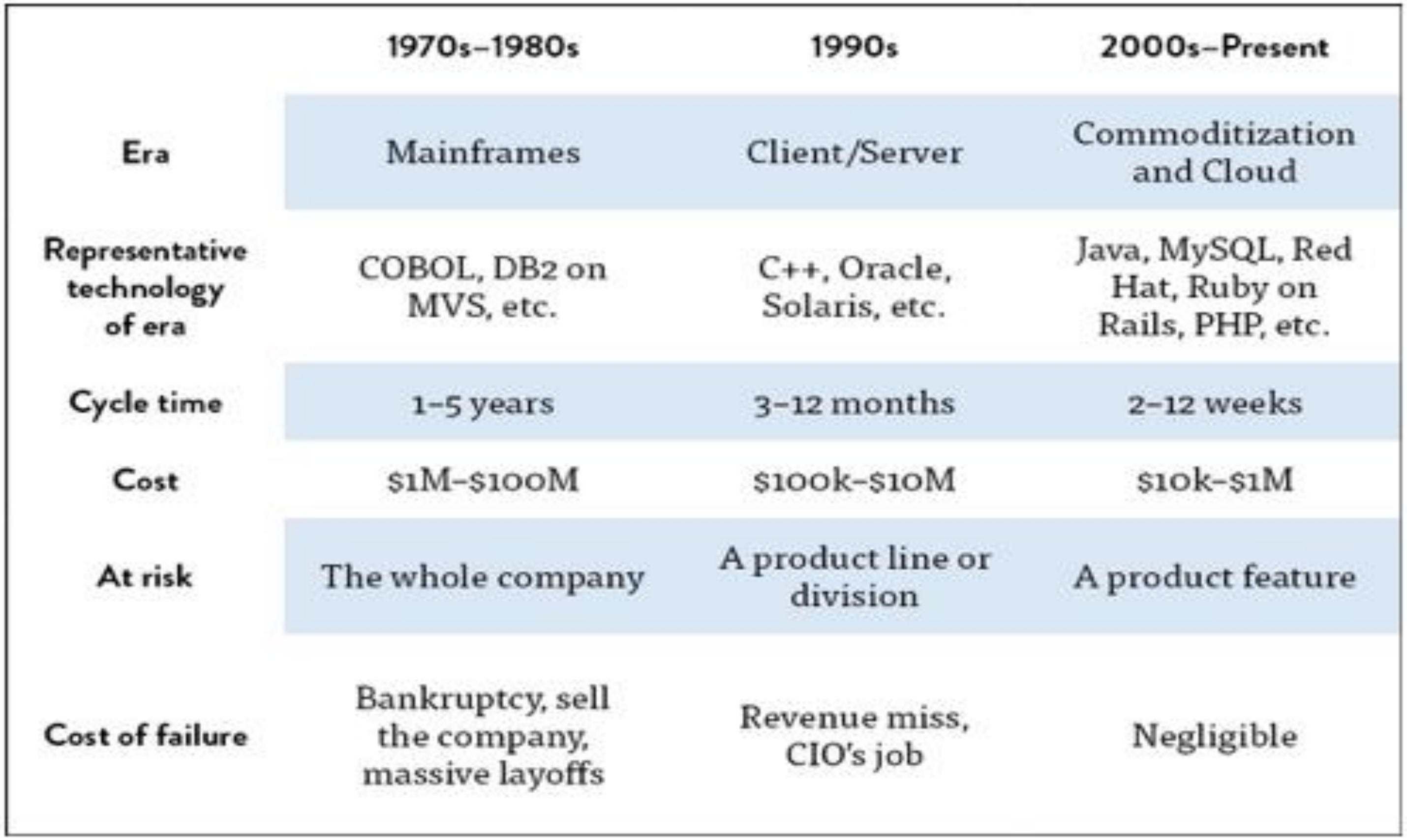
Advantages

Lean methods and manufacturing

Lean methods dramatically improved manufacturing

Before Lean	After Lean
On average, 6 weeks lead time	On average, < 3 weeks lead time
< 70% of orders shipped on time	> 95% of orders shipped on time

Software product improvements



	1970s–1980s	1990s	2000s–Present
Era	Mainframes	Client/Server	Commoditization and Cloud
Representative technology of era	COBOL, DB2 on MVS, etc.	C++, Oracle, Solaris, etc.	Java, MySQL, Red Hat, Ruby on Rails, PHP, etc.
Cycle time	1–5 years	3–12 months	2–12 weeks
Cost	$1M–$100M	$100k–$10M	$10k–$1M
At risk	The whole company	A product line or division	A product feature
Cost of failure	Bankruptcy, sell the company, massive layoffs	Revenue miss, CIO's job	Negligible

(Source: Adrian Cockcroft, "Velocity and Volume (or Speed Wins)," presentation at FlowCon, San Francisco, CA, November 2013.)

Lean success story…

GM NUMMI assembly line was so bad that GM closed it

The NUMMI workers were trained by Toyota to use Lean methods

After adopting Lean methods, NUMMI was building the highest quality cars of any GM assembly plant in the US

- Before: problems on the assembly line were diverted
- After: workers stop the assembly line and managers work with workers to understand and resolve the problem

Reflect on the problem to find a better solution

Managers job is to help workers

Lean doesn’t always work

The NUMMI approach didn’t work well at other GM assembly plants

Managers measured on the number of cars off the line

- **Not** the number of cars without problems

Just adopting a framework won’t help:

- Must also change management and leadership practices

Solution:

- Automatically detect errors
- Stop the process (andon cord)
- Solve the problem
- Continually improve the process

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

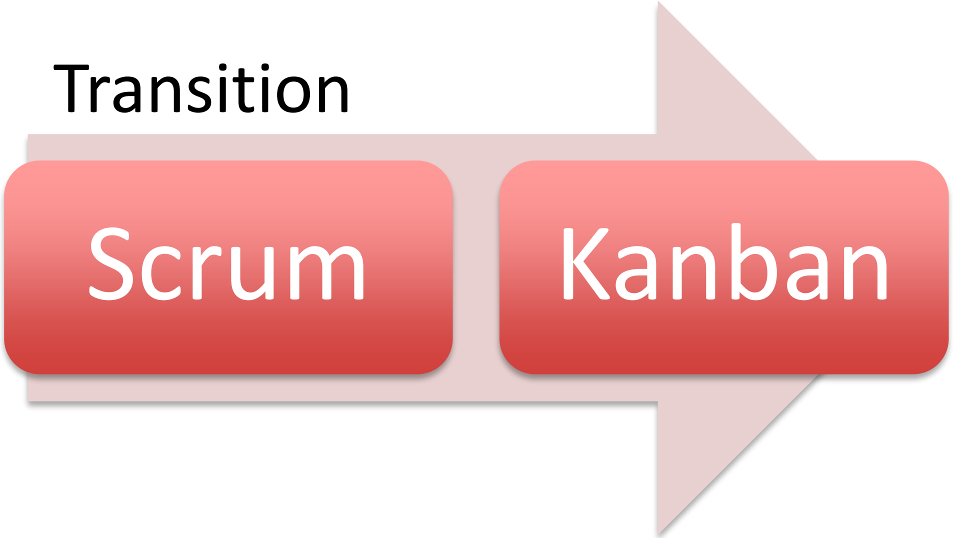
- Scrum features:
    - Scrum roles
        - Product Owner, Scrum Master, Developers
    - Scrum meetings
        - sprint planning, stand up, sprint review, sprint retrospective
    - Time-boxed deliverables
- Kanban features:
    - Just in time planning
    - Work In Progress limits

Scrumban alternatives

Include Kanban techniques in your Scrum process

$$Scrumban = Scrum + Kanban$$

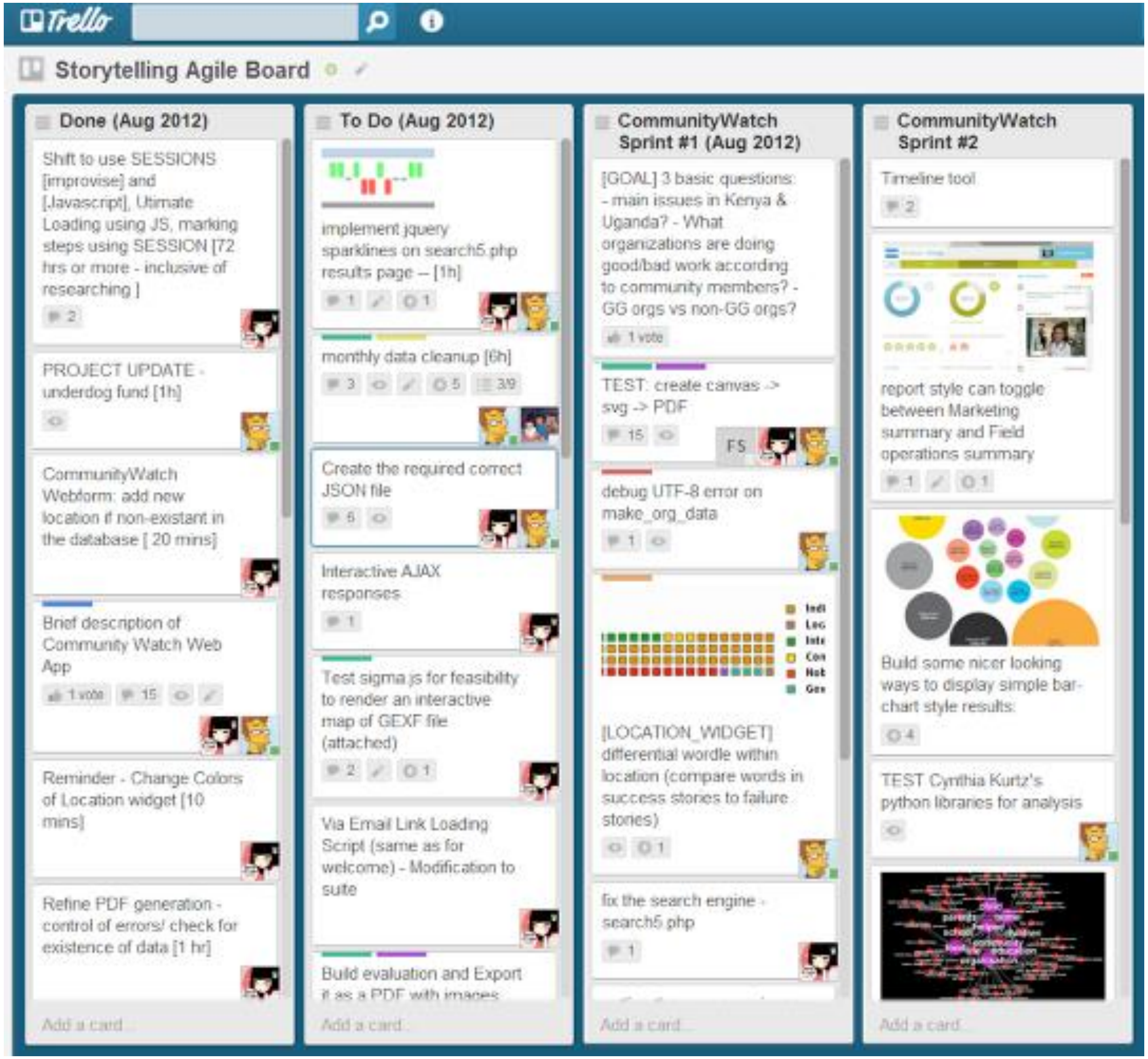
Transition from Scrum to Kanban



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 talk about building code
- May need more frequent releases than Scrum supports

Trello.com Kanban Boards



Storytelling Agile Board

Done (Aug 2012)

- Shift to use SESSIONS [improvise] and [Javascript], Ulimate Loading using JS, marking steps using SESSION [72 hrs or more - inclusive of researching] — 2
- PROJECT UPDATE - underdog fund [1h]
- CommunityWatch Webform: add new location if non-existant in the database [20 mins]
- Brief description of Community Watch Web App — 1 vote, 15
- Reminder - Change Colors of Location widget [10 mins]
- Refine PDF generation - control of errors/ check for existence of data [1 hr]

Add a card...

To Do (Aug 2012)

- implement jquery sparklines on search5.php results page – [1h] — 1, 1
- monthly data cleanup [6h] — 3, 5, 3/9
- Create the required correct JSON file — 6
- Interactive AJAX responses — 1
- Test sigma.js for feasibility to render an interactive map of GEXF file (attached) — 2, 1
- Via Email Link Loading Script (same as for welcome) - Modification to suite
- Build evaluation and Export it as a PDF with images

Add a card...

CommunityWatch Sprint #1 (Aug 2012)

- [GOAL] 3 basic questions: - main issues in Kenya & Uganda? - What organizations are doing good/bad work according to community members? - GG orgs vs non-GG orgs? — 1 vote
- TEST: create canvas -> svg -> PDF — 15 — FS
- debug UTF-8 error on make_org_data — 1
- [LOCATION_WIDGET] differential wordle within location (compare words in success stories to failure stories) — 1
- fix the search engine - search5.php — 1

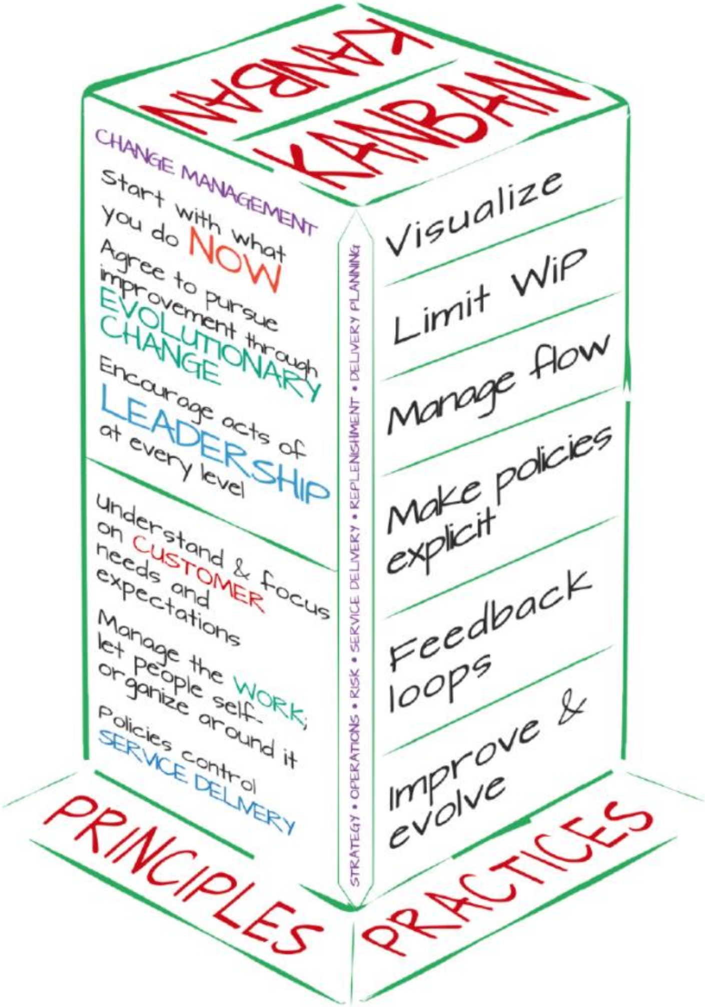
Add a card...

CommunityWatch Sprint #2

- Timeline tool — 2
- report style can toggle between Marketing summary and Field operations summary — 1, 1
- Build some nicer looking ways to display simple bar-chart style results. — 4
- TEST Cynthia Kurtz's python libraries for analysis

Add a card...

Kanban Summary


KANBAN
KANBAN
CHANGE MANAGEMENT
Start with what you do NOW
Agree to pursue improvement through EVOLUTIONARY CHANGE
Encourage acts of LEADERSHIP at every level
Understand & focus on CUSTOMER needs and expectations
Manage the WORK; let people self-organize around it
Policies control SERVICE DELIVERY
STRATEGY • OPERATIONS • RISK • SERVICE DELIVERY • REPLENISHMENT • DELIVERY PLANNING
Visualize
Limit WiP
Manage flow
Make policies explicit
Feedback loops
Improve & evolve
PRINCIPLES
PRACTICES

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

Questions?