

# Lean

**Lean and Kanban**

# Agile

**Michael Cox**

**Vice President and Senior Consultant**

**NetObjectives, Inc.**

[michael.cox@netobjectives.com](mailto:michael.cox@netobjectives.com)



“What is all this Agile  
stuff about, anyway?”

# *Lean and Kanban*

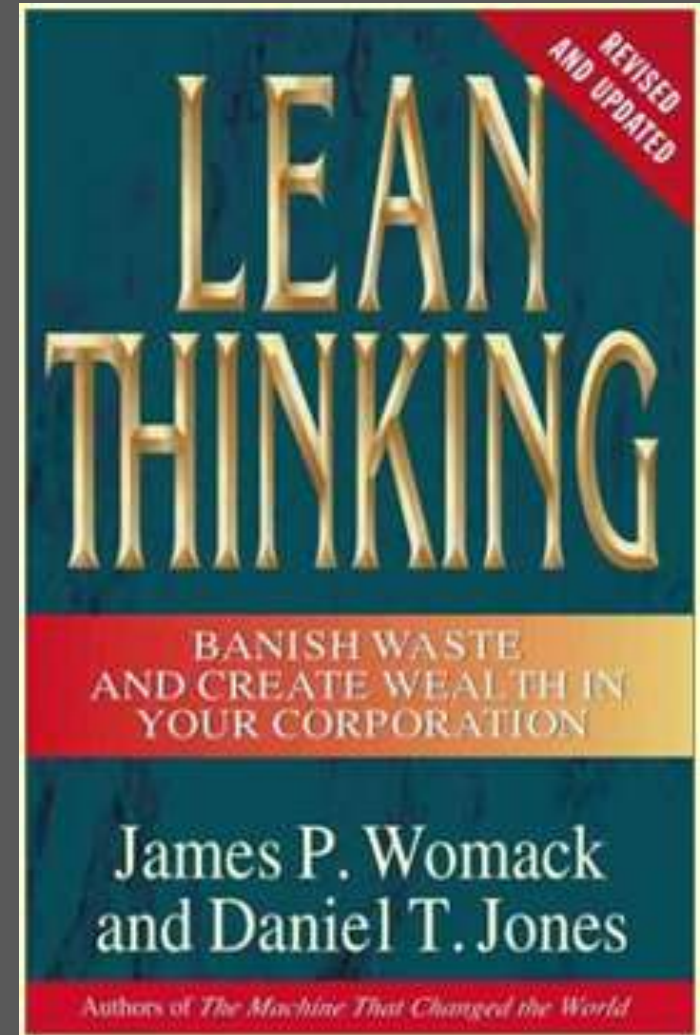
**How do they  
compliment each  
other?**

**How do you use them?**

**Why does it work?**

# Lean

- **Value from the customers perspective**
- **Identify and eliminate waste – non value added activities**
- **Flow of work at customer demand**
- **Continuous improvement**



# Kanban

A management discipline.

A constant exercise of matching demand with supply, to deliver the right thing at the right time.

*See also:* Visibility, Prioritization, WIP limits, Pull

# *Agile*

**Agile is a method that features rapid delivery of  
functional product iterations**

**Relies on immediate customer feedback**

**Allows for evolving understanding of system**

# Agile

Agile is about

Business Iterations  
*not* Development  
Cycles

# Agility

Predictability  
of Business Value  
Realization



“Where did this **stuff** come from?”

# How Did We Get Into This Spot?

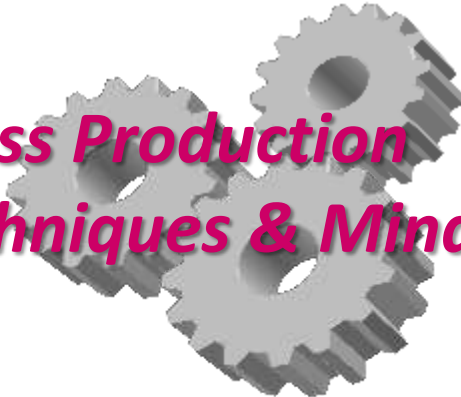
- Tremendous rise in the standard of living the past 100 years in all developed countries
- Rise was largely driven by productivity improvements
  - Agricultural up 3 to 5% a year since 1900
    - 50% of workforce in 1900, < 2% today, more production
  - Production up by 3% a year since Depression
    - 35% of workforce in 1940, < 15% today, 100x output rise

**Basis has been the  
Invention and  
Widespread  
Adoption of Mass  
Production Techniques**



# How Did We Get Into This Spot?

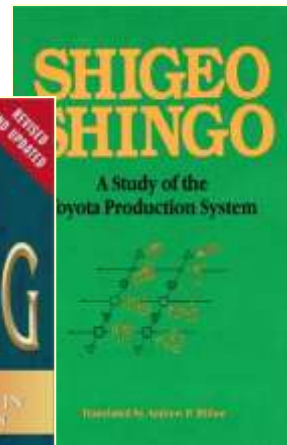
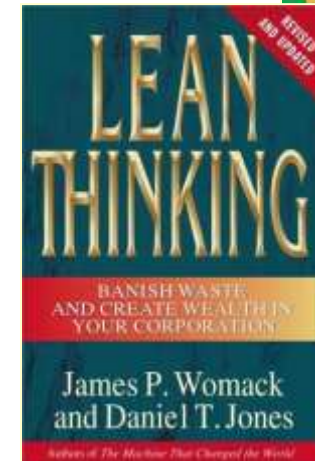
*Mass Production  
Techniques & Mindset*



*Thought Basis for  
Current Management  
Practices*

- Managing via hierarchy, command and control
- Scientific management – the one best way
- Economies of scale
- Batch production

**Lean Principles have generated  
Lean Practices**



# How Did We Get Into This Spot?

- Mass production management techniques in systems and software development have largely failed
  - Documentation = Understanding
  - The right tasks, correct pressure - force it to happen
  - “If they would freeze requirements, we would be fine”
  - “Heroes” called in when program is in real trouble
- A dissatisfied customer community has imposed more controls and rigidity
- Contractors countered with rigid contracts and change orders to batter the customer with cost and schedule
- Product owners were not involved until too late



we are always  
working with  
**uncertainty**

*Lean and  
Kanban help  
us deal with  
uncertainty*

**The result is agility**

*Lean suggests limit  
TIME between steps*

*Kanban suggests  
limit # of items  
being worked on in  
each step*

*time*

*size of queue*

# Understanding Lean

1. Value from the Customer's Perspective
2. Value stream
3. Flow
4. Pull
5. Perfection

- Define the value
- See the value stream
- Flow and where value comes from
- JIT
- Cycle time
- Reduce waste

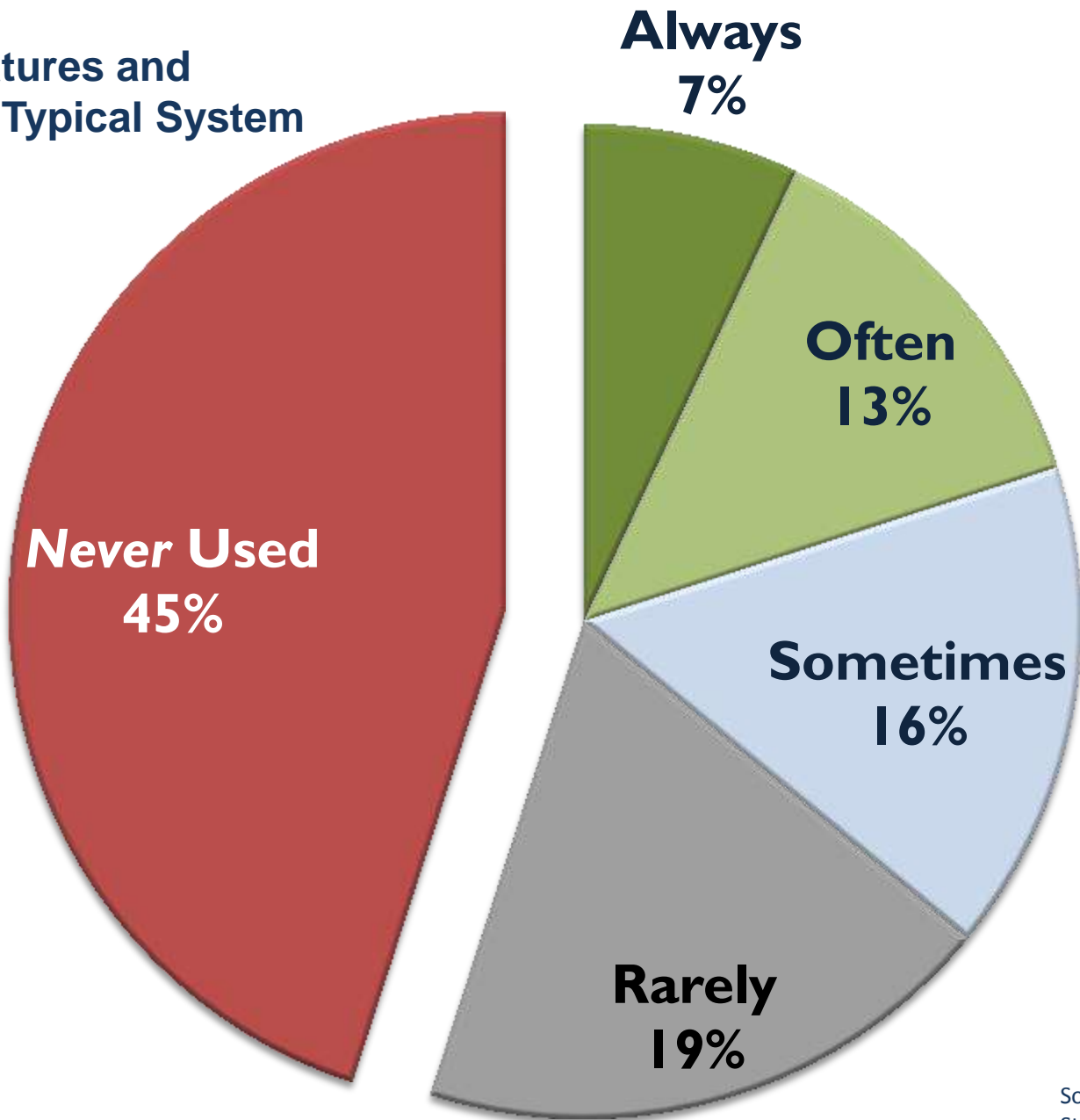
thinking  
points



“You cannot **build** the right thing  
if you have not **discovered** it first!”

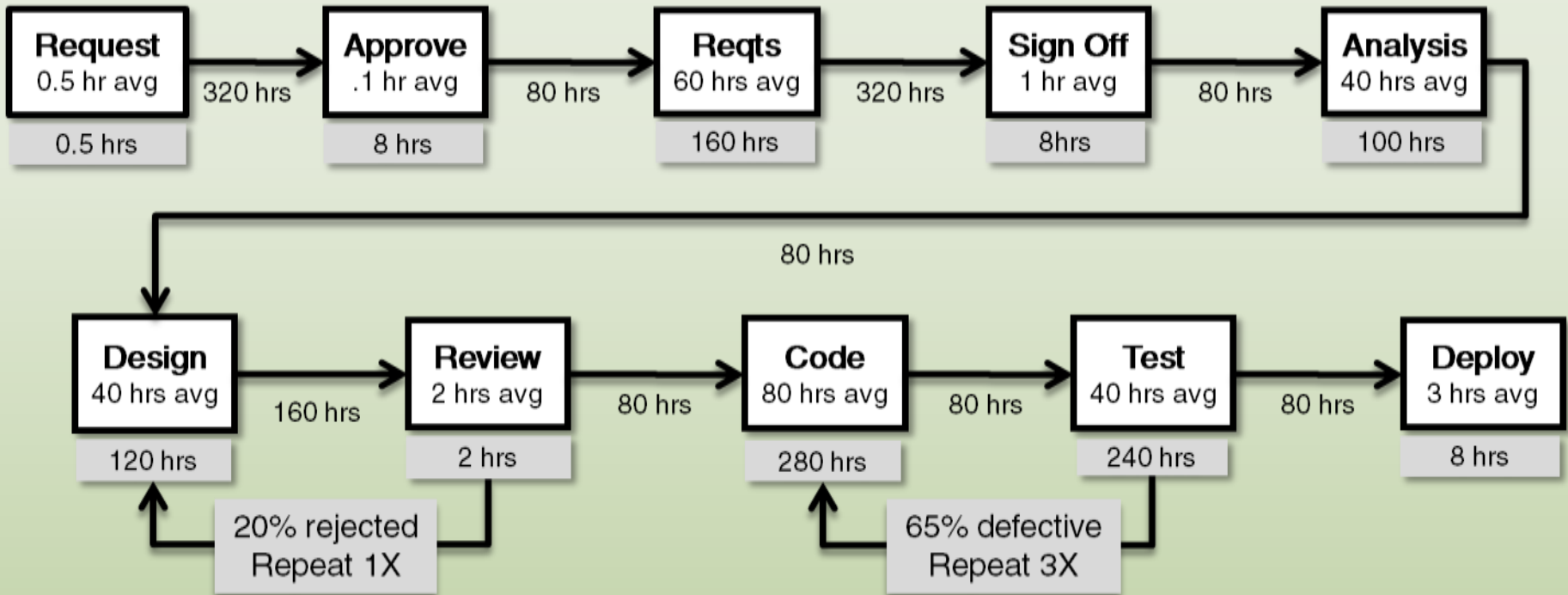
*The product owner must own the  
definition of value!*

# Usage of Features and Functions in Typical System



Source: Standish Group  
Study of 2000 projects at  
1000 companies





visualize the **entire value stream**

## *The value stream*

- Continuous flow of valuable work and features into deployment
- Includes *everybody* from the customer to operations and support engineers, and *not just development*

visualize the **entire value stream**



# FLOW

FOCUS ON  
TIME

Optimize  
the  
whole!

Large batches create delay  
and waste

while

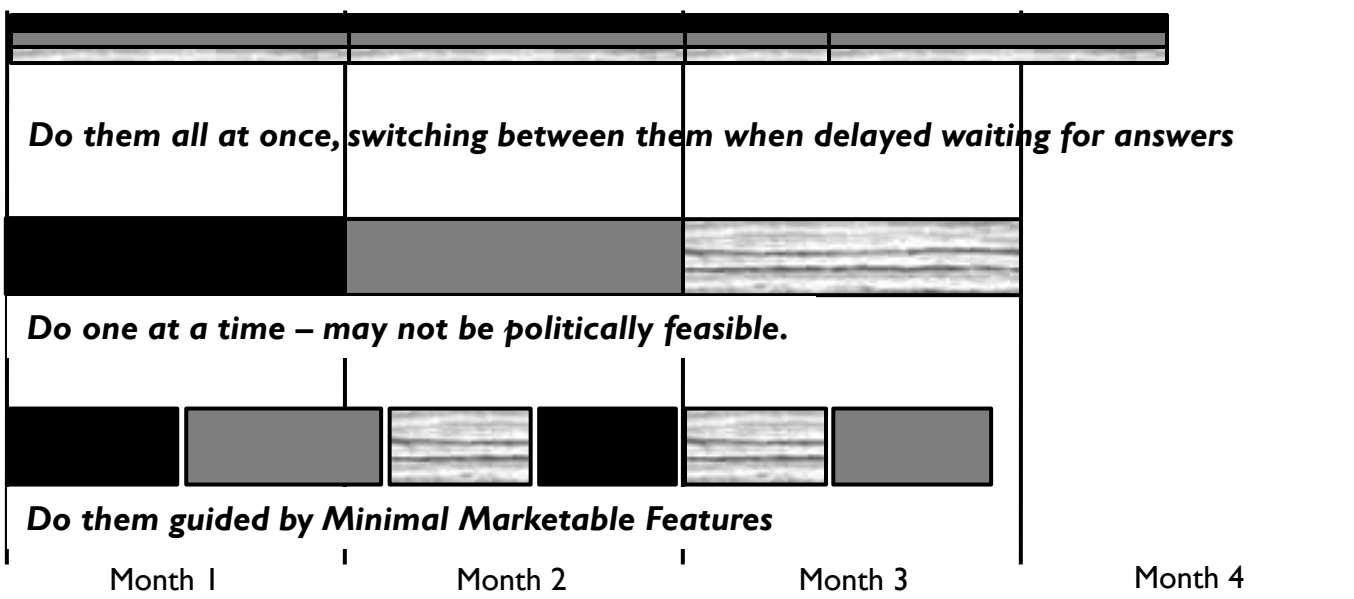
Small batches create  
incremental value






# Task-Switching and Schedules

LEAN THINKING

## Three ways to do three projects



- Project 1 
- Project 2 
- Project 3 



# DELAY IS

finding

redoing

reworking

waiting

hand-offs

bottlenecks

information delay

untested code

unread requirements

transaction related

coordination related







*Cycle Time  
is Key!*

# Requirements ...

**Decay** and

**Lose Value**

**over time**

# Requirements

are not fully understood even  
after a formal sign-off



# Requirements

change often

during long development cycles





# Requirements

piled on  
 poorly prioritized  
 long delivery cycles



# Pull

The work enters a queue.

When someone needs new work, they pull from the queue.

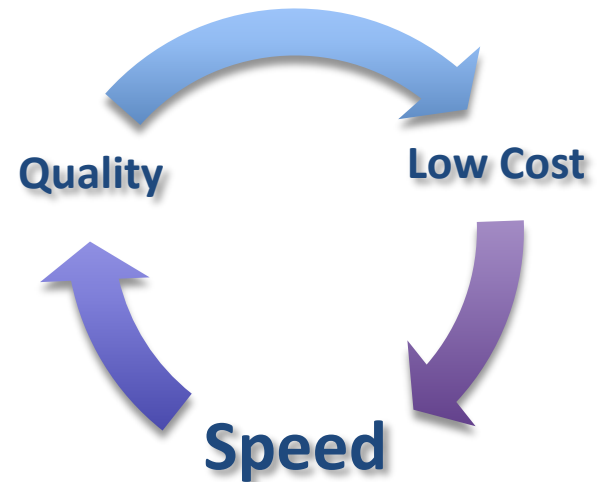
The work goes through a number of stages. When the work is done in a stage, it flows down to the next stage.

Until it is done.

## Short Cycle Time

# Principles of Lean Software Development

- Optimize the Whole
- Eliminate Waste
- Build Quality In
- Deliver Fast
  
- Defer Commitment
- Create Knowledge
- Empower People



*kanban*

*improves quality and  
lowers cost*

by *eliminating delays*

by **managing WIP**



# Kanban for Systems and Software

Workflows can be seen and managed

You can divide the work into small value adding increments

It is possible to develop value-adding increment in a continuous flow, from requirement to deployment

# Kanban for Systems and Software

Limit Work in Process (WIP)

Pull value through

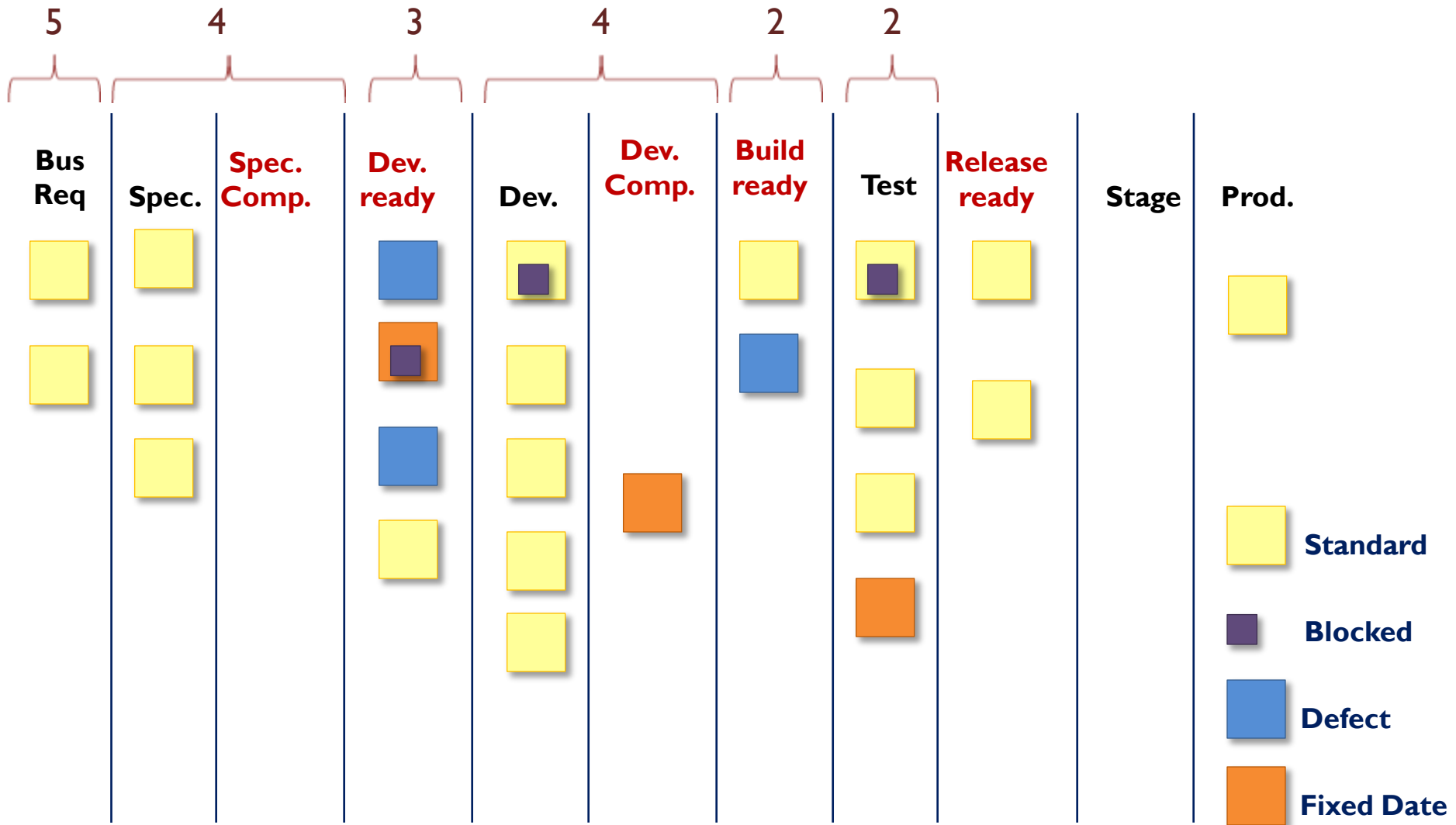
Make it visible

Increase throughput

Prioritized Backlog

Quality is built in

Team continuously monitor and improve



A person wearing a dark blue shirt and dark pants is suspended in a red bucket, working on the glass facade of a building. The person is seen from behind, looking towards the left. The glass reflects a bright blue sky with scattered white clouds. The text "make all work visible" is overlaid on the right side of the image in a dark blue, sans-serif font. The word "visible" is significantly larger than "make all work".

make all work  
**visible**

**Kanban boards reflect**  
**Priority**  
**WIP**  
**Process**

# Business Value Kanban

*Business Discovery*

*Business Delivery*

	?	?	?	?	?	?	?	?	?	
	Input	Prioritize	Sequence / incremental	Technical Analysis	Staging	Readiness	Specify	Execute	Deploy & Ready to Use	Implement
		◆				◆				
						10				
Exit Entry										

**Don't build features that nobody needs**

(right now or in some cases, ever)

**Don't write more specs than you can code**

**Don't develop more code than you can test**

**Don't test more code than you can deploy**

# Kanban Success

Focus on Quality

Reduce WIP

Balance demand against throughput

Prioritize

lean  
thinking



throughput



utilization





# Stop Starting and Start Finishing

# Pull

The work enters a queue.

When someone needs new work, they pull from the queue.

The work goes through a number of stages. When the work is done in a stage, it flows down to the next stage.

Until it is done.

# WIP Limit...

Governs the *maximum* number of work items that can be in that state at any instant.

## **Below its limit:**

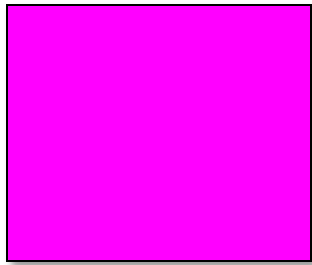
Receive a work item from upstream

## **At its limit:**

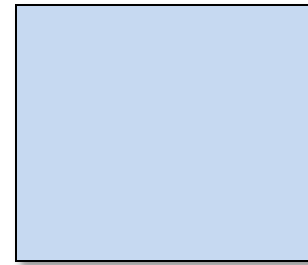
Wait for one of its own items to be completed and flowed downstream

*In Knowledge Work, complexity grows exponentially with WIP*

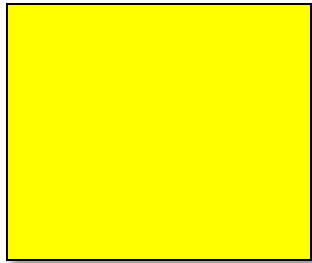
# Classes of Service



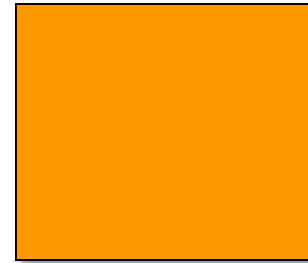
Expedite



Development  
Story



Specific  
Delivery Date



Outside  
Impact



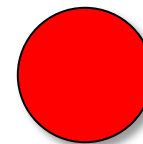
Standard  
Maintenance or  
Break-Fix Work



Impediment

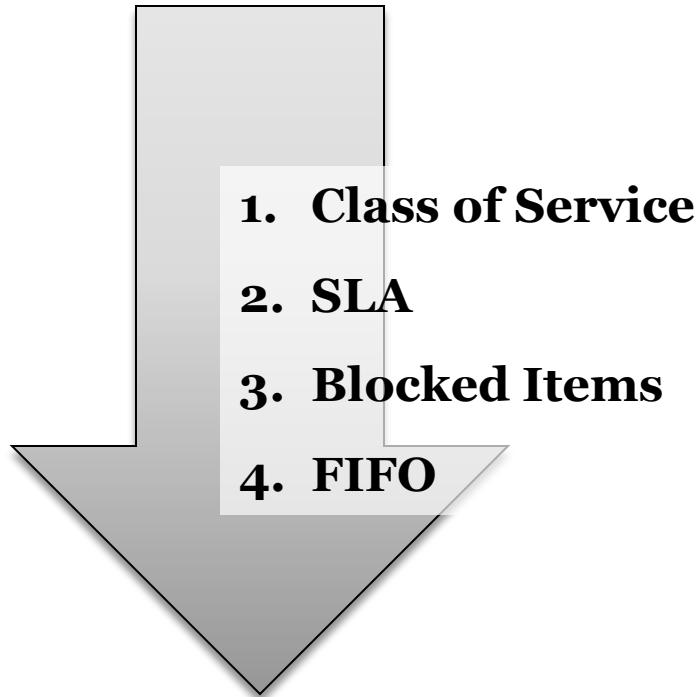


Standard  
New or  
Value-Added Work

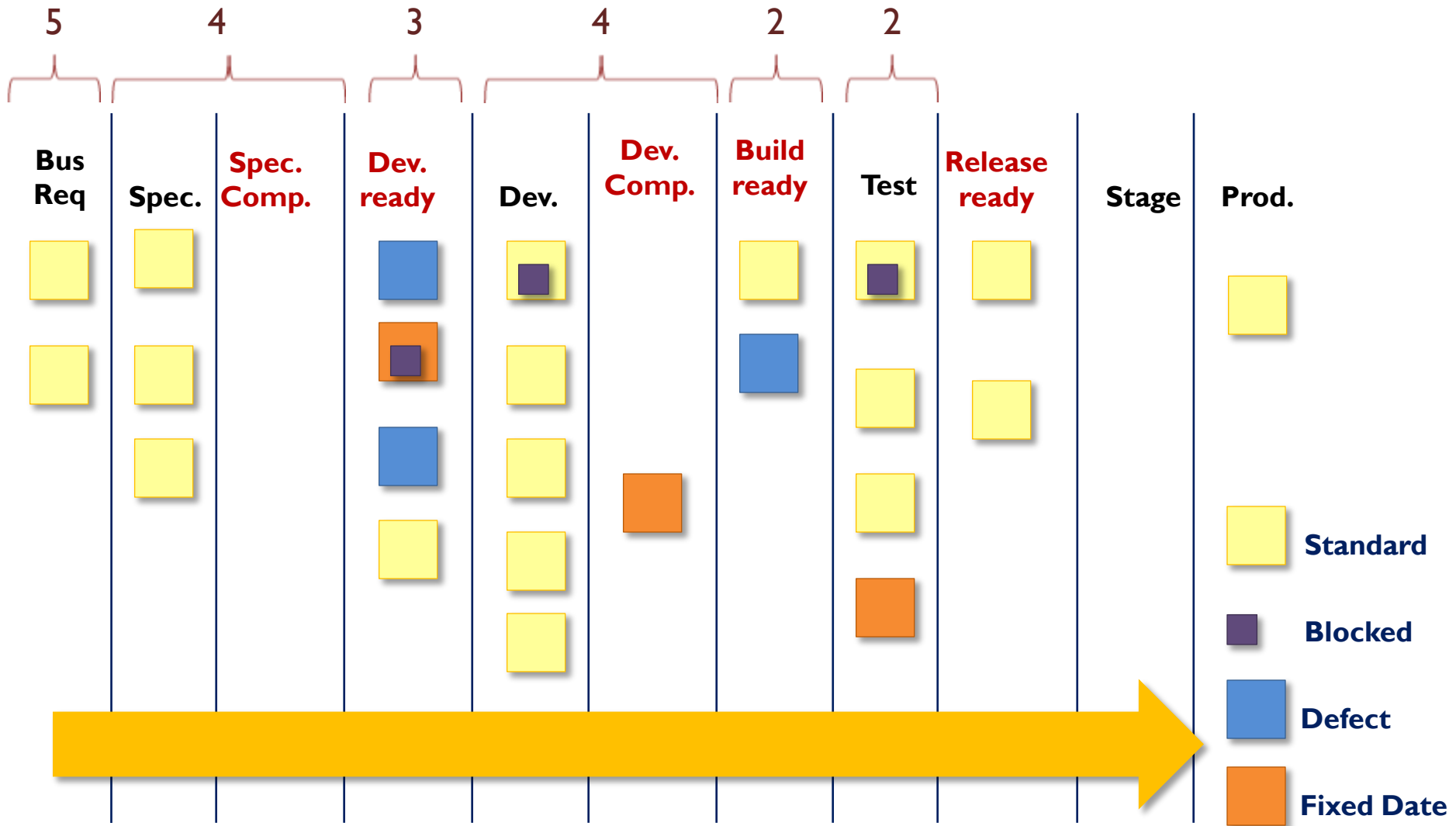


Red Flag  
Issue

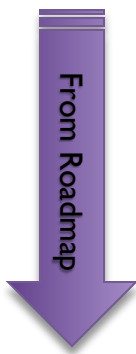
# Policies & SLAs



Direct the team  
in the **priority** of  
processing work items



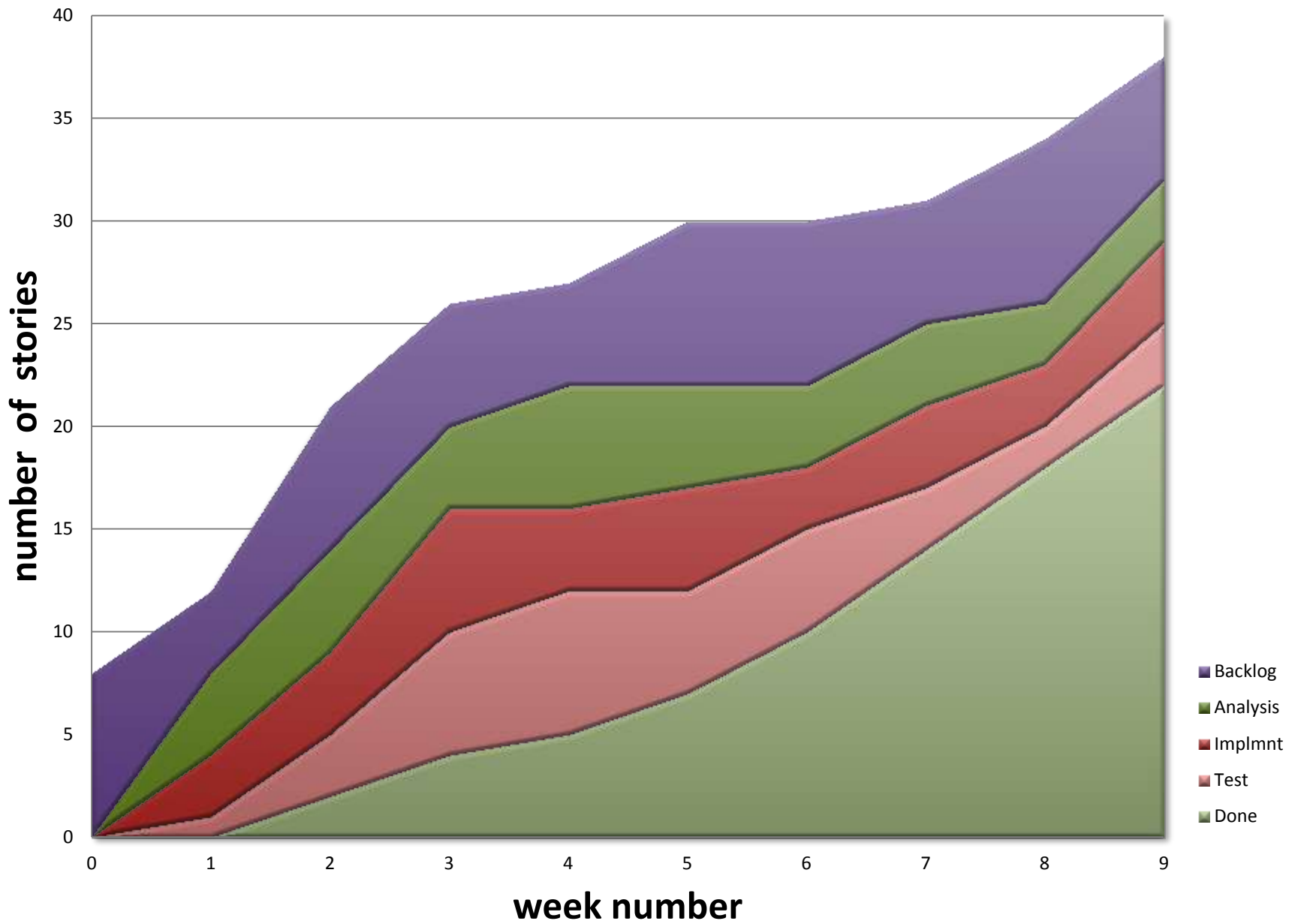
# Monitoring flow: Kanban for portfolio

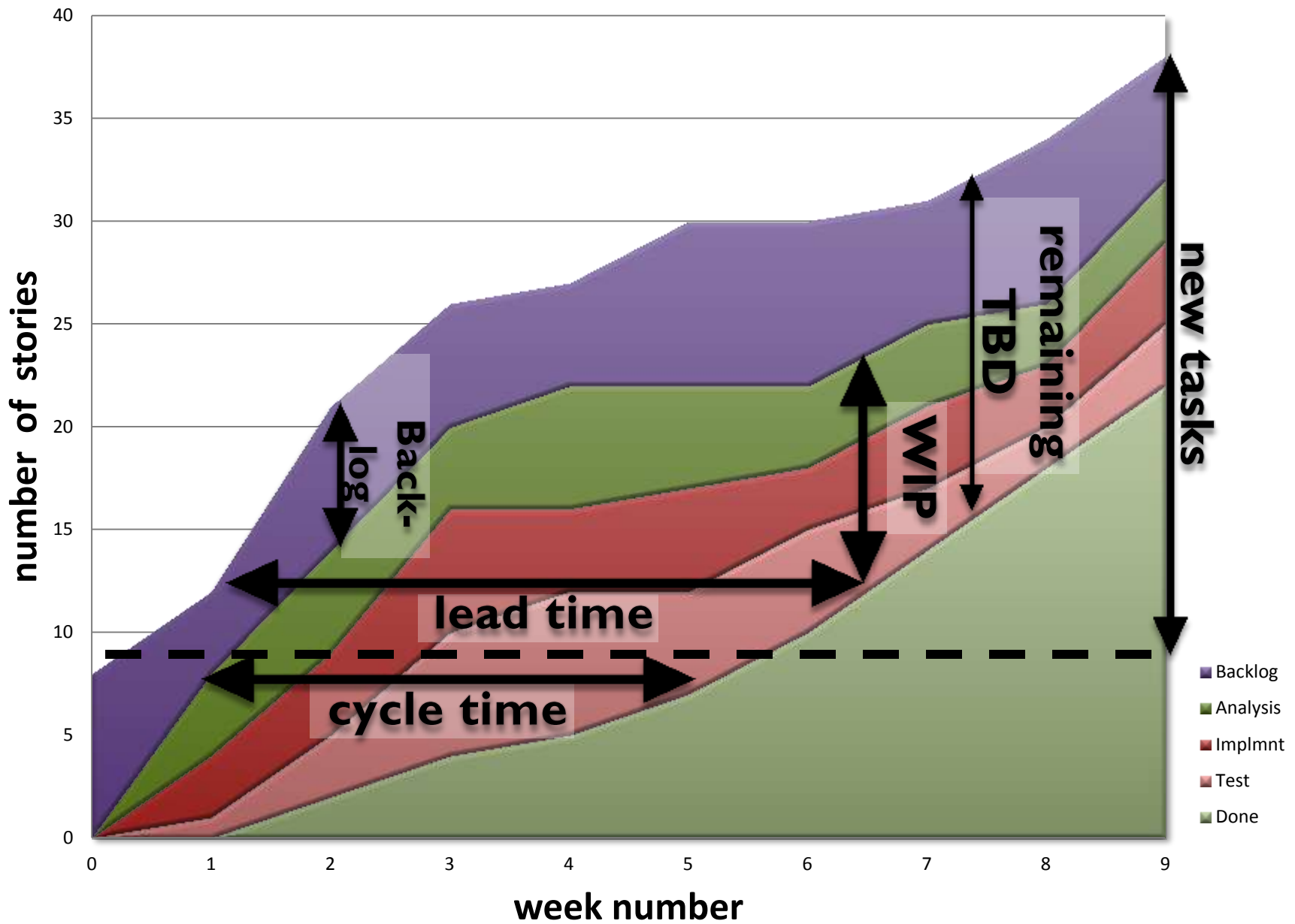


Status	Backlog	Specify (right size)	Execute	Validate	Done/ Released
Support					
Project X					
Project Y					
Project Z					
WIP Limit	14	4	3	3	









always focus on

# WHAT SHOULD WE FINISH NEXT?

## Kanban Stand-up

### Who Talks?

- Only Team members moving stickies across the board!

### Do This

- Start from the **right**
- Work by the **highest priority**
- Pay attention to:
  - Oldest
  - Blocked
  - Class of Service
  - SLA in jeopardy
- Ask
  - Do we have a bottleneck (congestion or gaps in the queues)?
  - Do we have a “blocker” not dealt with?
  - Are we keeping to our WIP limits?
  - Are priorities clear?

### When done

- Update the board
- Remove done items from the board

GUIDELINES

- **Agree to goals**
- **Map the value stream**
- **Define a set of work item types**
- **Meet with external stakeholders**
- **Create board for tracking**
- **Agree to standup**
- **Agree to operational review**
- **Educate the team**
- **Start doing it**

# Kanban

## *What you will see:*

- Queues start backing up immediately following any blockage
- Predictable consequences
- The entire board will slow down as a result of flow issues
- Teams see issues right away and act together to fix them



# *Lean and Kanban*

**Lean is the theory**

**Kanban is the approach**

**Agile is the result**

Questions

?