# WHY KANBAN?

Troy Tuttle

Project Lead Consultant, AdventureTech

Troy.Tuttle@adventuretechgroup.com

TroyLTuttle@gmail.com

blog.troytuttle.com

twitter.com/troytuttle

linkedin.com/in/troytuttle

# Motivation

**What is my motivation?**

- A practicing “Pragmatic Agilist”

  *Context is King*

- Helping software teams find project success by indentifying and avoiding project <u>dysfunction</u>
- Family-friendly definitions of successful projects.

# What is Kanban?

Is Kanban the next martial arts fad?

# What is Kanban?

Is Kanban the latest miracle cure?

How Tacos Can Save Your Life
A startling new discovery in San Antonians' favorite Mexican foods that can prevent major disease -- even
STOP CANCER
TONIGHT 10
KENS5

# What is Kanban?

Is Kanban a …

… silver bullet?

# What is Kanban?

## Some Observations..

- Kanban is the result of practitioners applying lean principles to software engineering.
- Kanban is influenced heavily by existing first-generation Agile methodologies.
- Kanban is more like a tool than a methodology.

# What is Kanban?

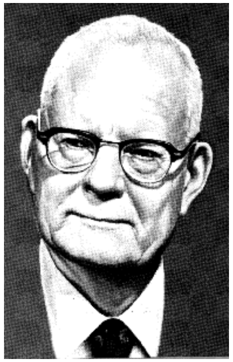
## It starts with Lean Principles:

- Pull
- Continuous Flow
- Customer Value
- Waste Elimination
- Continuous Improvement

Founder of Toyota Production Systems →

Taiichi Ohno

Quality Guru: System of Profound Knowledge →

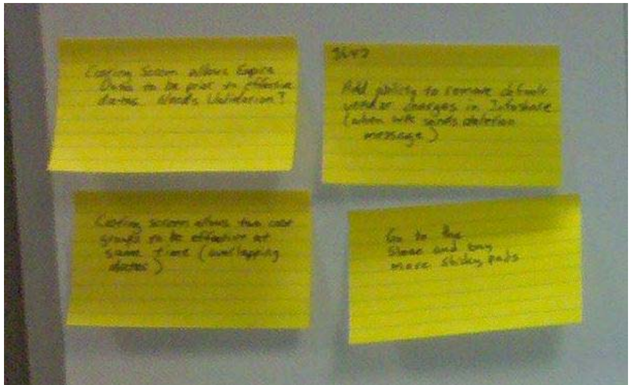


W. Edwards Deming

# Kanban Is

*Kanban is translated as:*
*"Visual Card" or "Signal Card"*

For our purposes (software), we can think of it as a virtual signal.


Costing Screen allows Expire Dates to be prior to Effective dates. Needs Validation?
3647
Add ability to remove default vendor charges in Infoshare (when wk sends deletion message)
Costing Screen allows two cost groups to be effective at same time (overlapping dates)
Go to the Store and buy more sticky pads

# What is Kanban?

## Kanban for Software's Lean Heritage:

- Don't **build** features that nobody needs **right now**
- Don't **write** more **specs** than you can **code**
- Don't **write** more code than you can **test**
- Don't **test** more code than you can **deploy**

*- Corey Ladas (leansoftwareengineering.com)*

[4]

# Kanban Is…

"Kanban is a transparent, work-limited, pull system."

*-- Eric Willeke*

*(KanbanDev Yahoo! Group)*

# Kanban for Software Principles

- Pull value through the Value Stream.
- Limit work in progress.
- Make it visible!

[3]

# Value Stream

- Lean software engineering starts with mapping the value stream.
- “Concept to Cash”
- “From the lips of the customer to a production system”

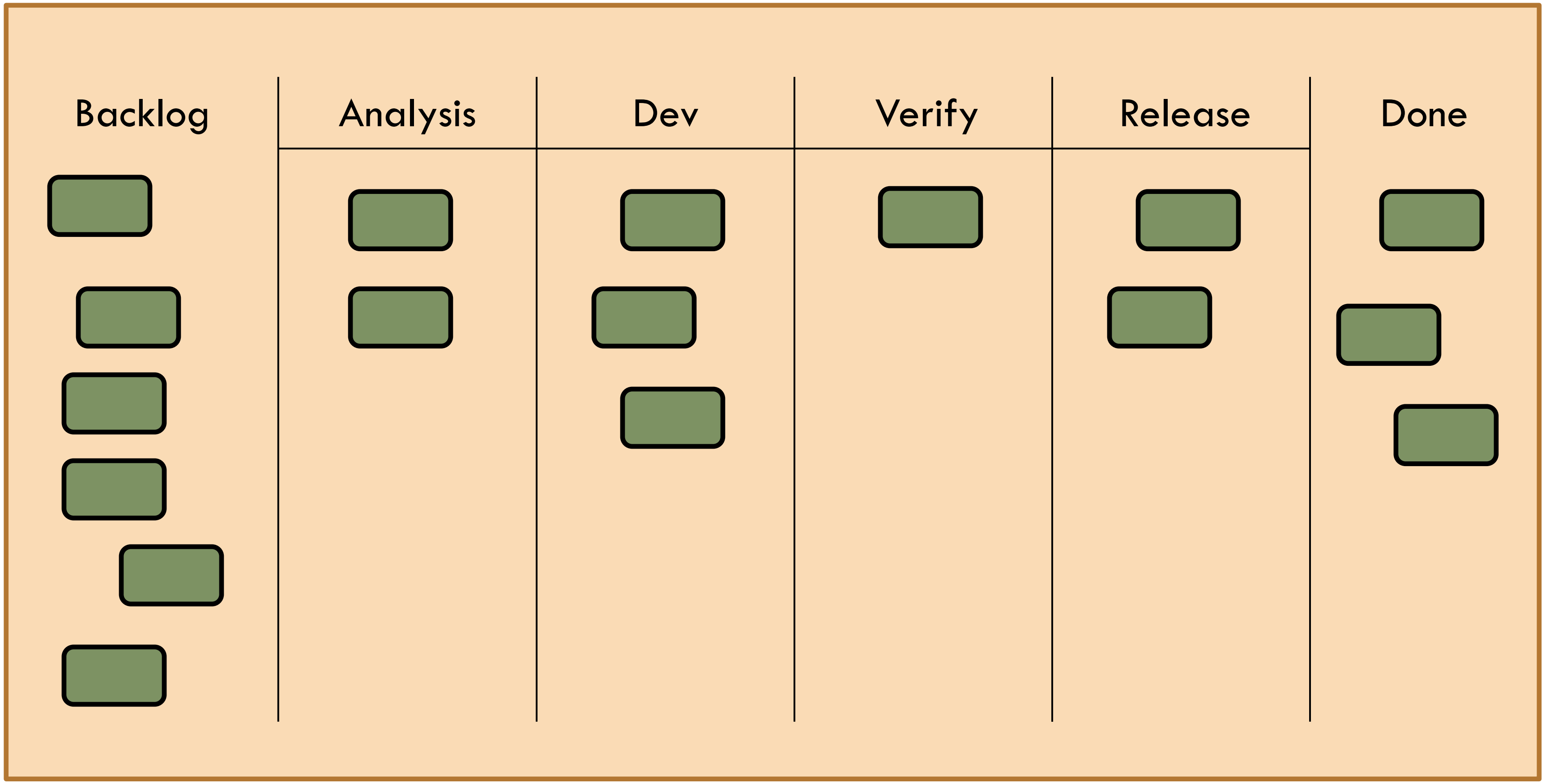
# Value Stream

Your value stream could be something like:

Request → Analysis → Dev → Verify → Release

# Value Stream

## Our first cut…



Backlog

Analysis

Dev

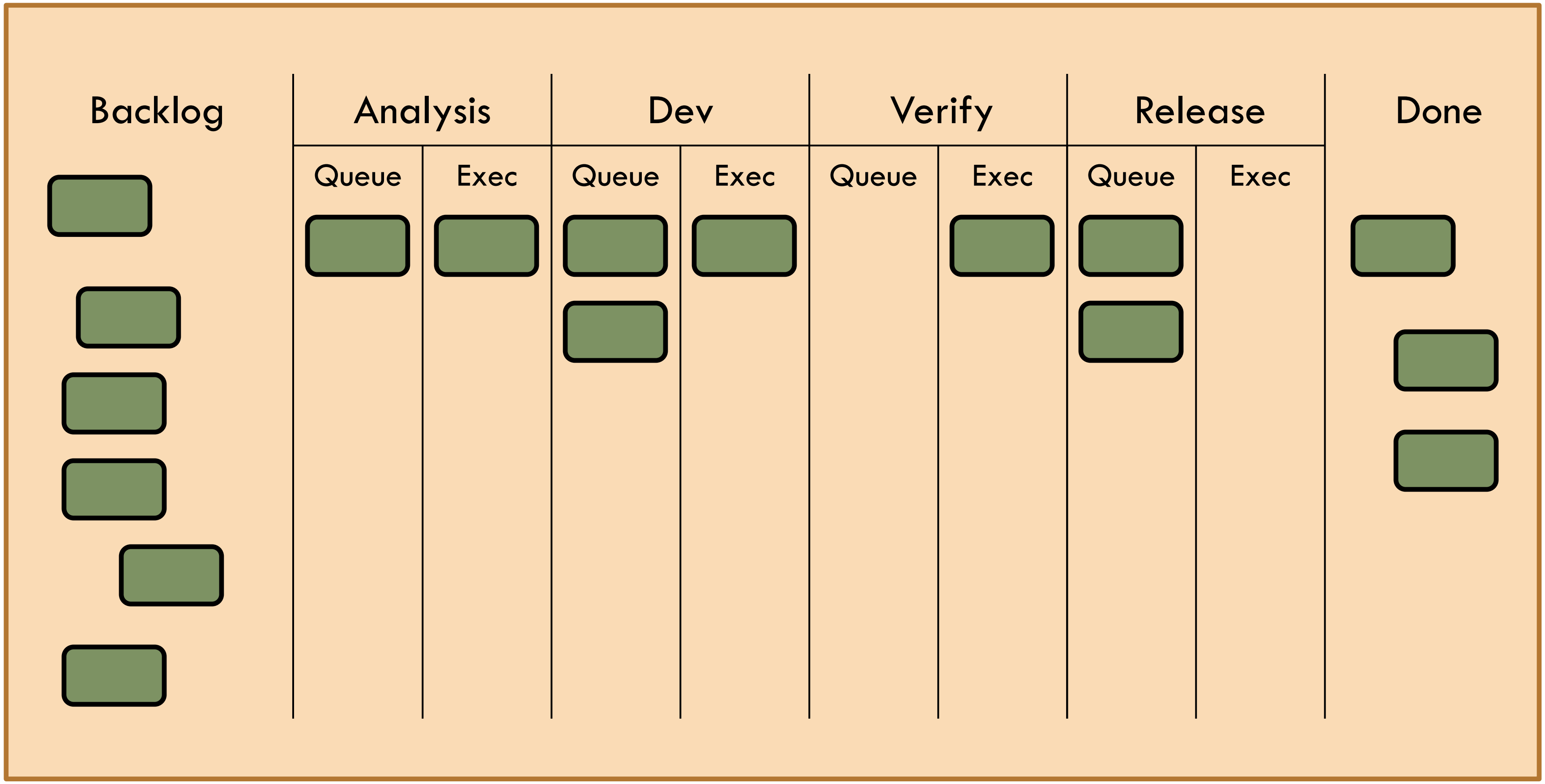
Verify

Release

Done

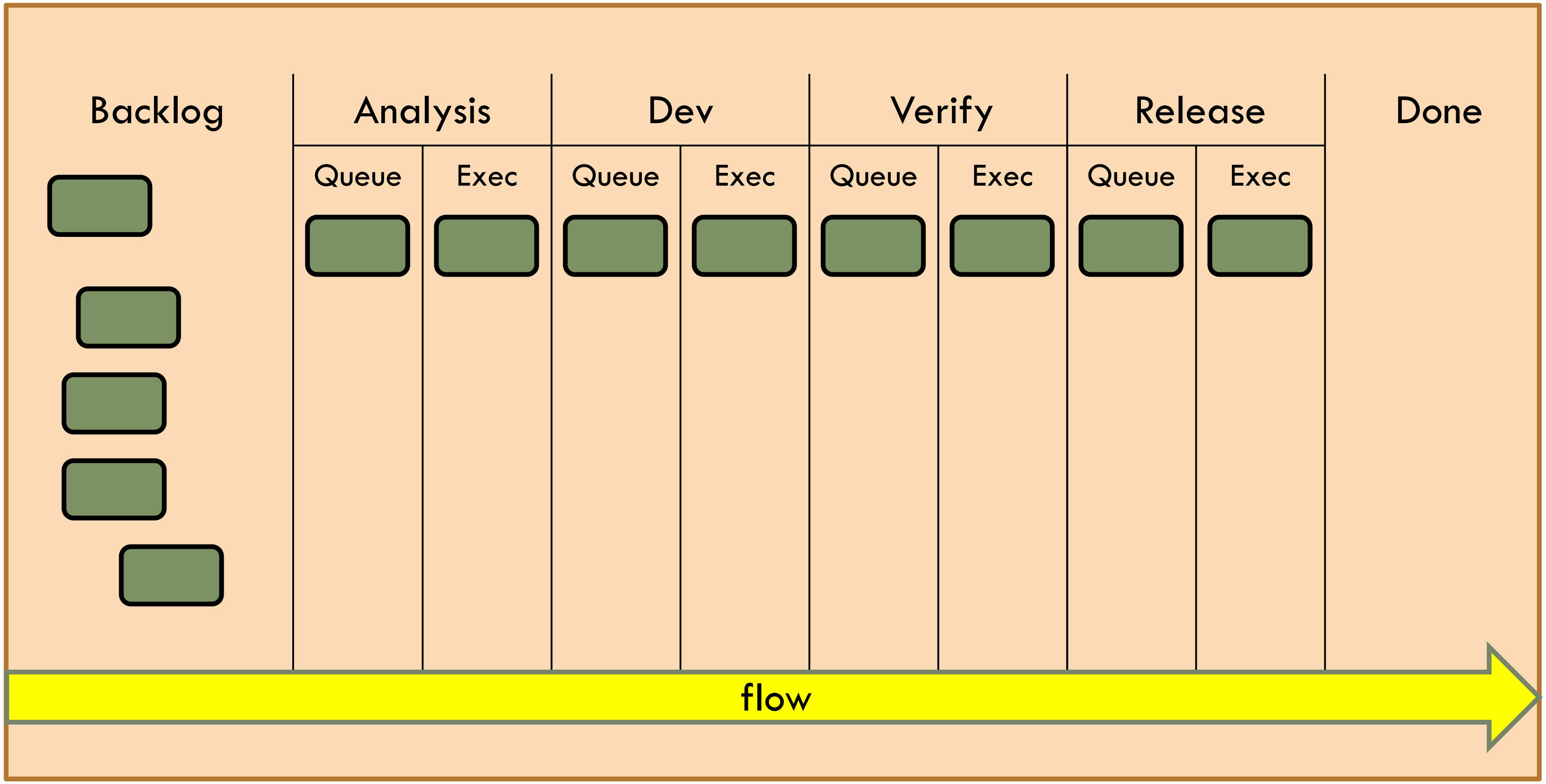
# Value Stream

Queues are important!



| Backlog | Analysis | | Dev | | Verify | | Release | | Done |
|---|---|---|---|---|---|---|---|---|---|
| | Queue | Exec | Queue | Exec | Queue | Exec | Queue | Exec | |
| 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | | 🟩 | 🟩 | | 🟩 |
| 🟩 | | | 🟩 | | | | 🟩 | | 🟩 |
| 🟩 | | | | | | | | | 🟩 |
| 🟩 | | | | | | | | | |
| 🟩 | | | | | | | | | |
| 🟩 | | | | | | | | | |

# Kanban Pull



Backlog

Analysis

Queue

Exec

Dev

Queue

Exec

Verify

Queue

Exec

Release

Queue

Exec

Done

flow

[1]

# Kanban Pull and Flow

- Phase-based development (waterfall) transfers the entire batch from state to state.
- Existing Agile development transfers small batches from state to state (iterations).
- Our goal with Kanban is to transfer one piece at a time (***one piece flow***), or at least work in that direction.

*Consider watering your garden with a bucket versus a hose.* [1]

# Limiting Work In Progress

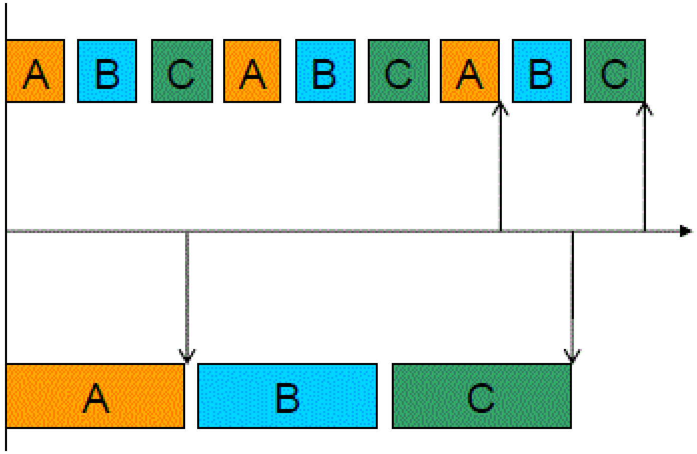
## Multitasking Exercise!

| | | |
|---|---|---|
| A | 1 | I |
| B | 2 | II |
| C | 3 | III |

[5]

# Limiting Work In Progress

Performing tasks sequentially yields results sooner.

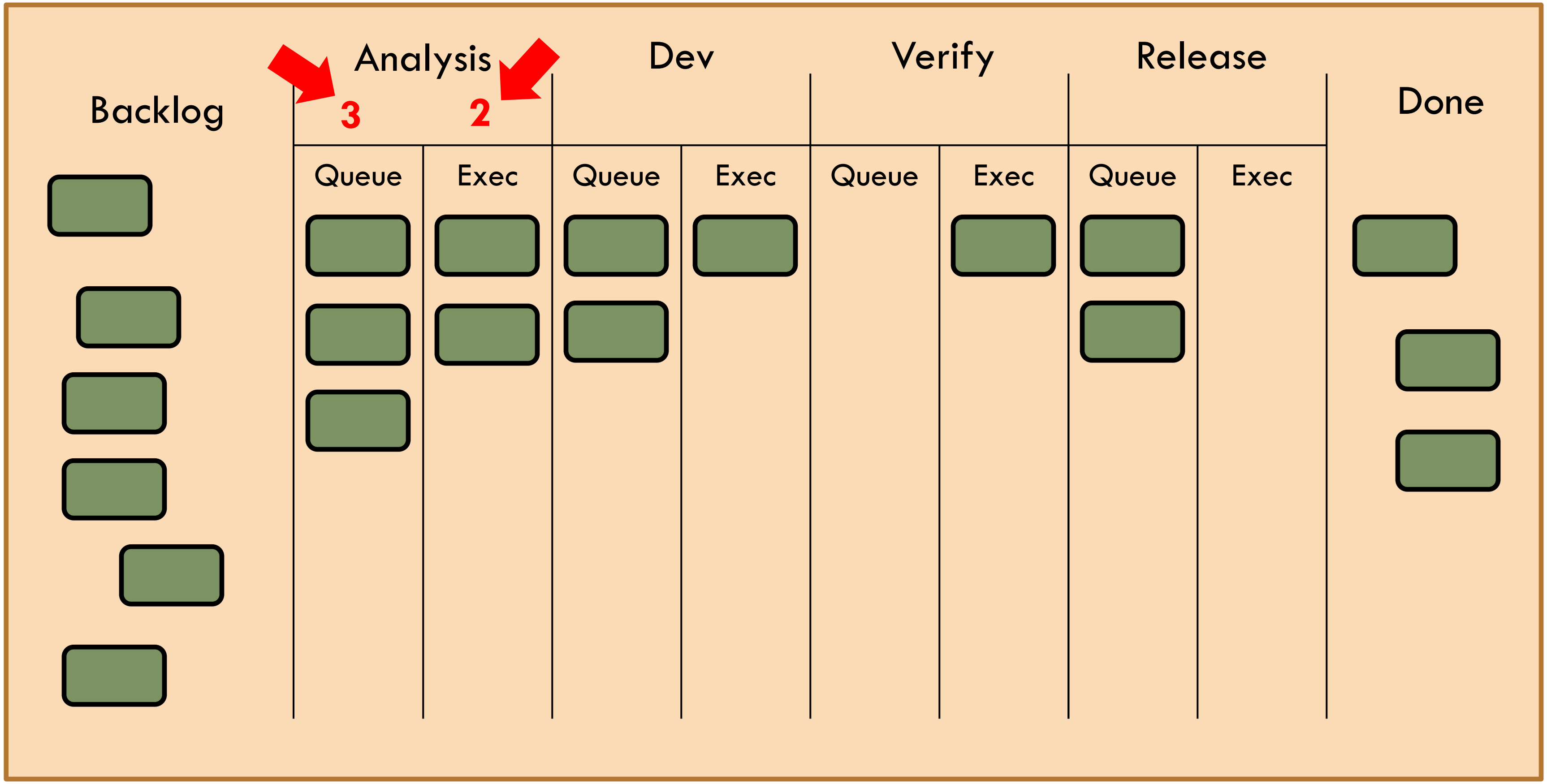


*(Observe service organizations to see this principle in action.)*

[1]

# Limiting Work In Progress

New work items can only be pulled into a state if there is capacity under the WIP limit.



Backlog

Analysis

3

2

Dev

Verify

Release

Done

Queue

Exec

Queue

Exec

Queue

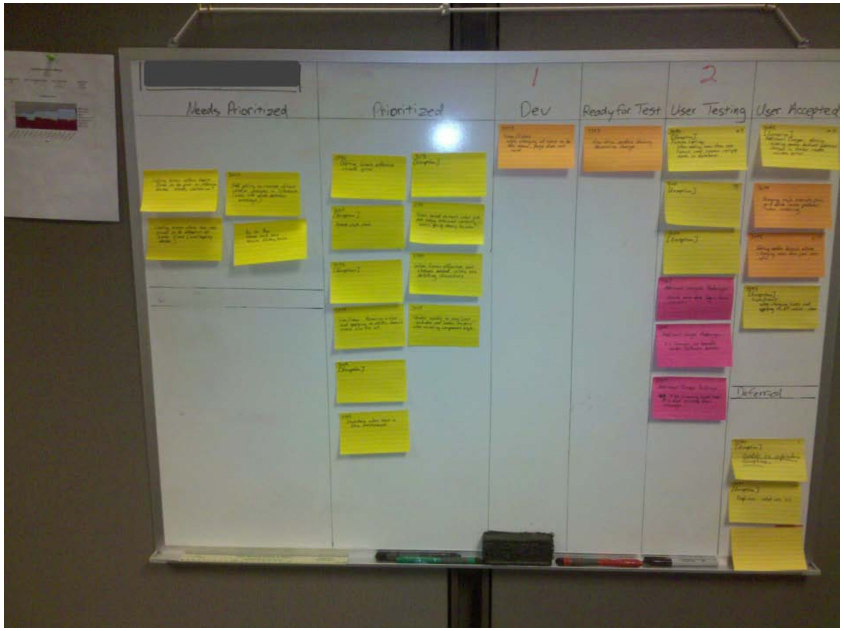
Exec

Queue

Exec

# Visualize!

Put work items on a white board and make it visible!



Then have the conversation with the customer, "this is what we have in process now, to take on new work, what shall we back out of the system?"

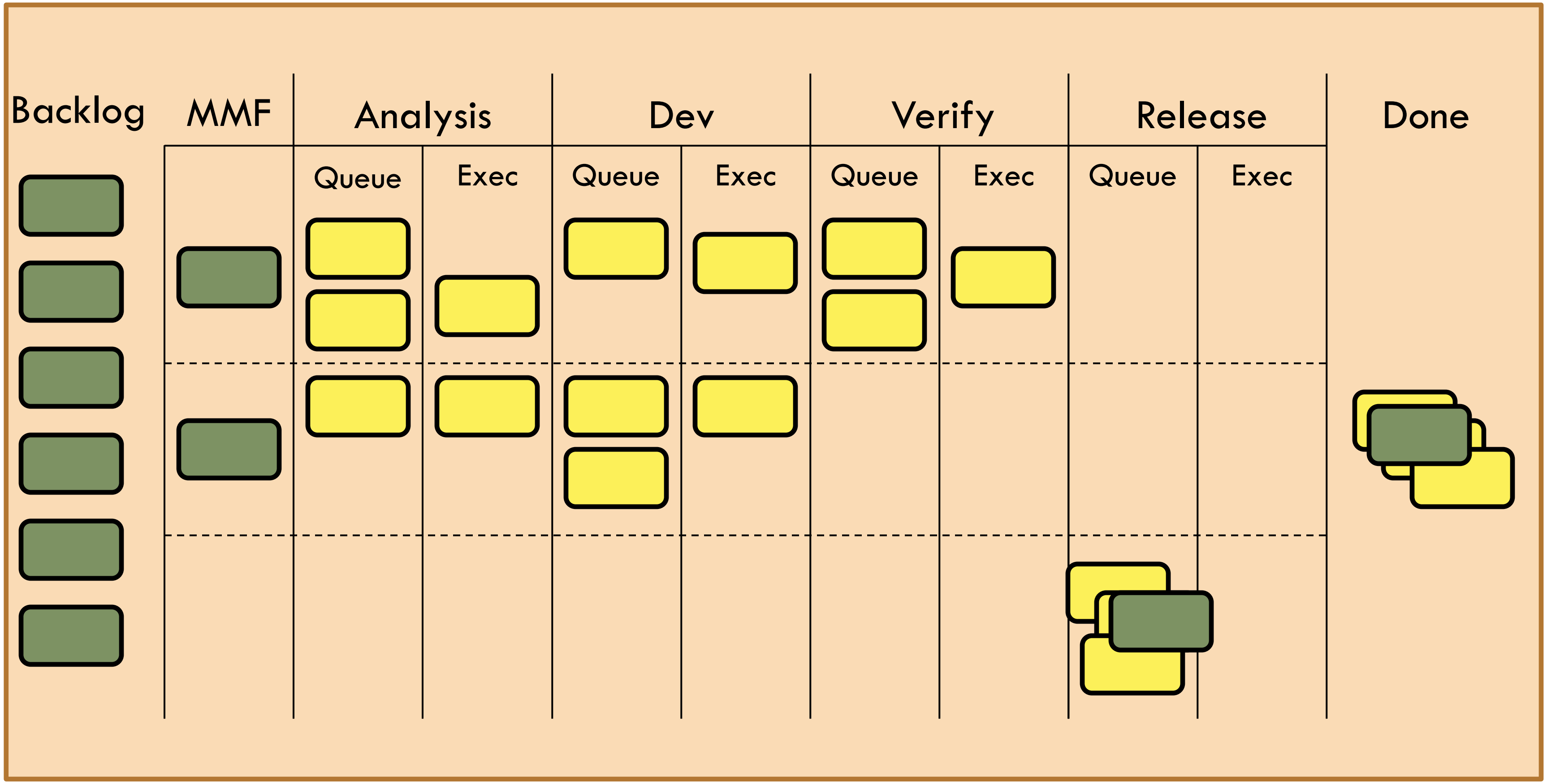
# Work Items: MMF

“A minimal marketable feature is a chunk of functionality that delivers a subset of the customer’s requirements, and that is capable of returning value to the customer when released as an independent entity” *-- M Denne & H Cleland-Huang, Software by Numbers*

**AKA: Epics, Feature Sets, and MMR** [6]

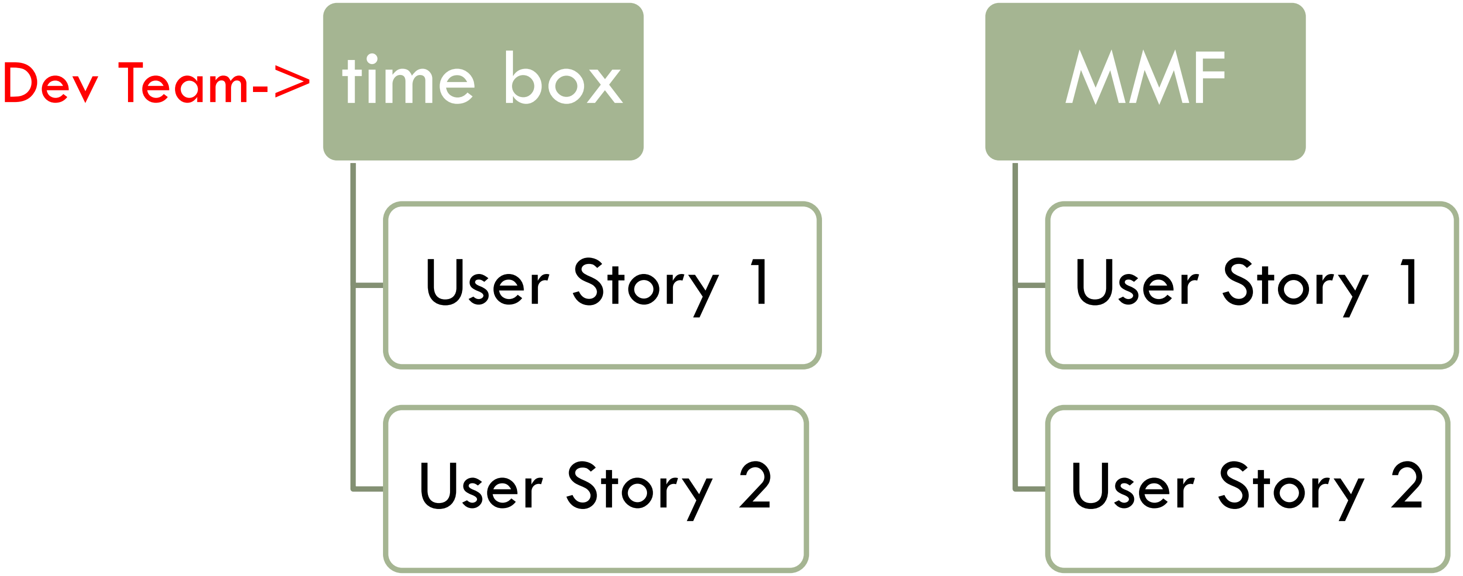
# Work Items: MMF

A two-tiered Kanban board with MMF's and user stories.



# Work Items: MMF

For Agile practitioners, MMF's in Kanban often replace the practice of time-boxed iterations.



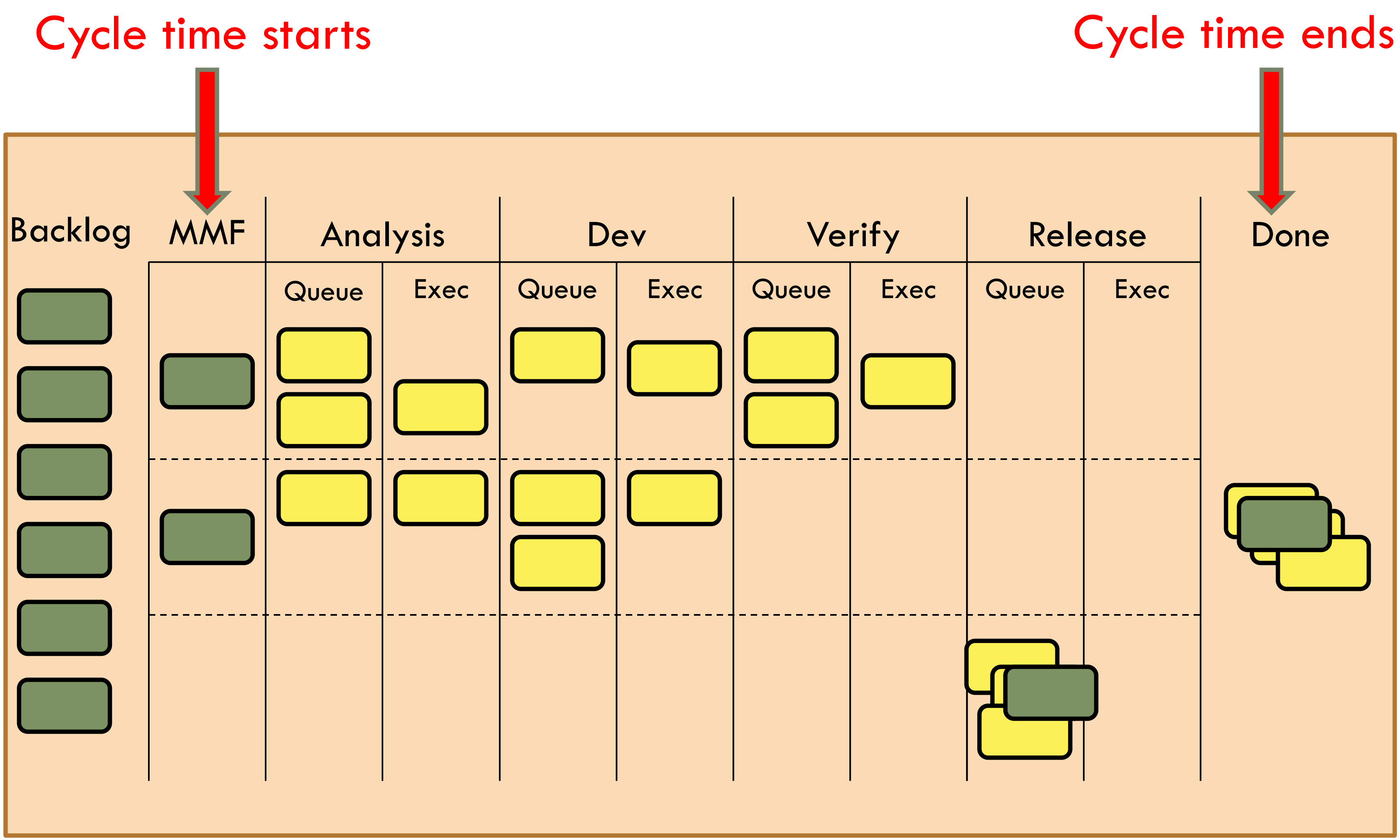
# Work Items: MMF

Example of MMF > User Story > Scenarios:

- The system should allow company users to resolve duplicate customer records.
  - So that we can eliminate duplicate processes, as a data processor, I need to merge two or more duplicate records into one.
    - Before merging records, confirm user has “merge” role permissions in system.
    - When merging customer records, verify at least one record contains valid demographic information.
    - After a successful merge, log merge relationship information to merge log table.

# The Estimation Game

# The Estimation Game: Cycle Time



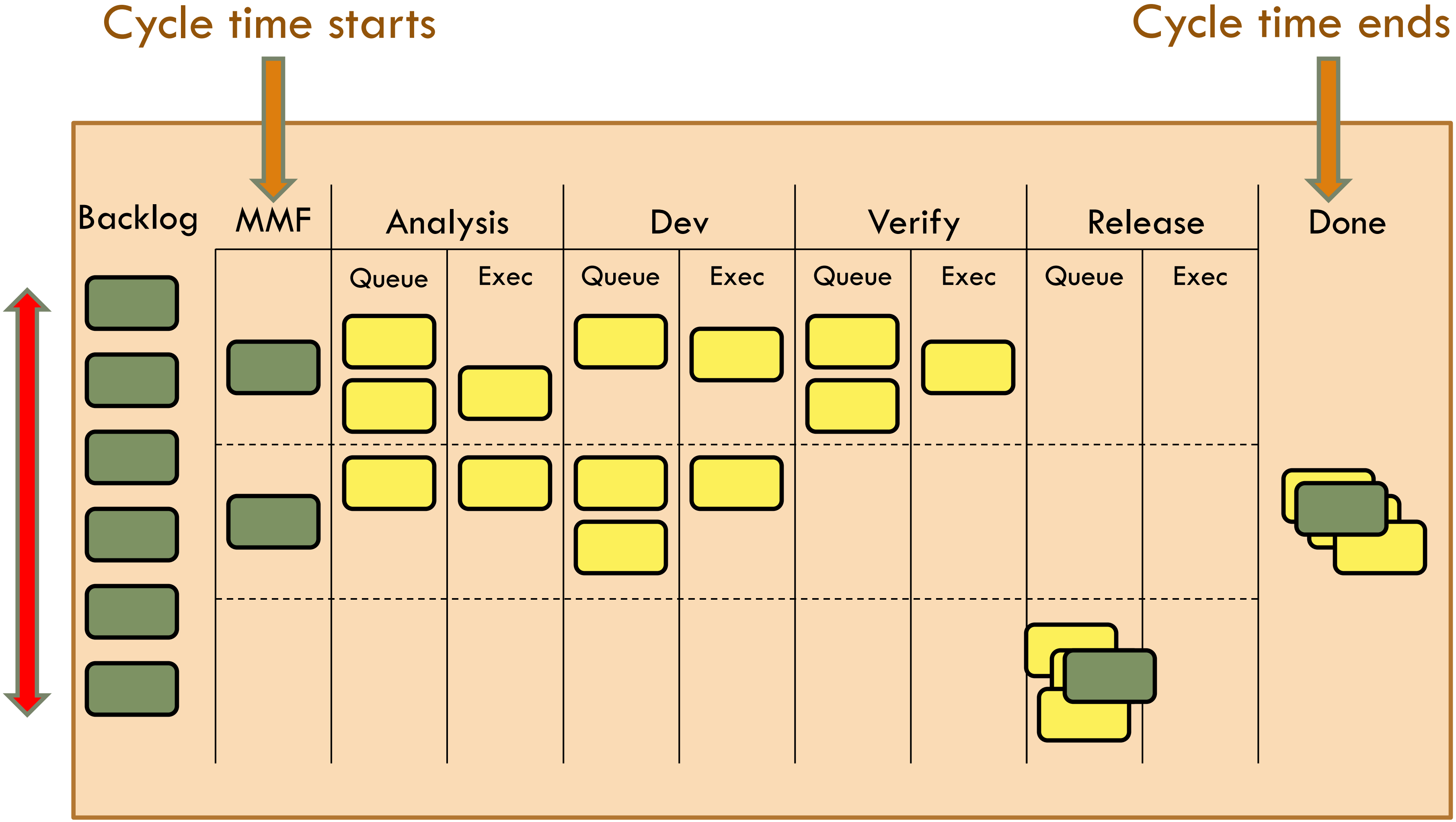
# The Estimation Game

- Kanban + cycle time metrics allow teams to stop playing the estimation game where value is placed on ***predicting*** when software will be completed.
- Instead, teams can ***project*** software completion rates from ***actual*** data.
- The effort that formerly went into improving ***prediction*** practices can now go to improving actual software delivery.

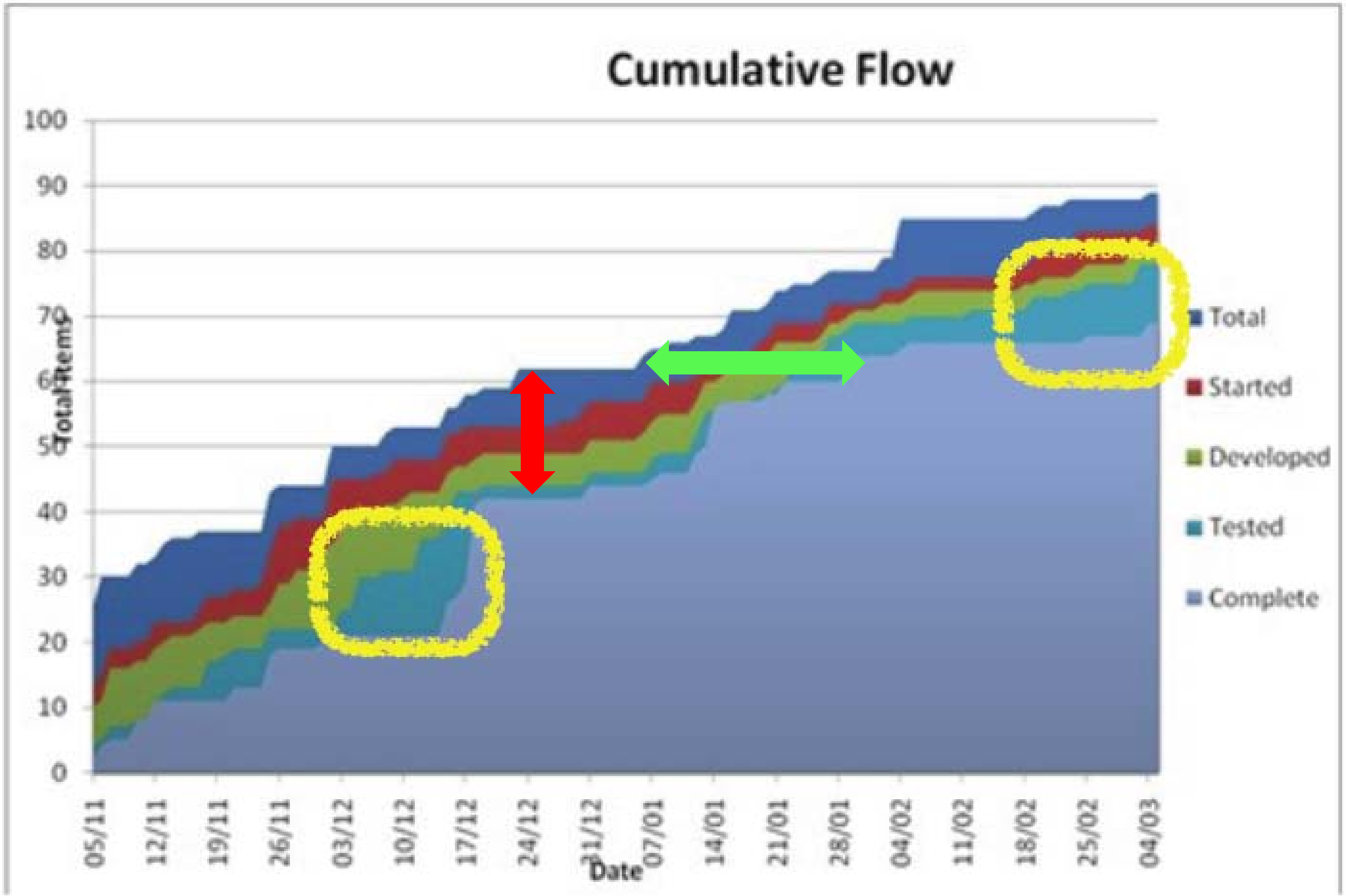
# Metrics

- Kanban teams need just a few metrics and tools to be effective:
  - Cycle time
  - Lead time
  - Cumulative Flow Diagrams

# Lead Time


Cycle time starts
Cycle time ends
Backlog
MMF
Analysis
Dev
Verify
Release
Done
Queue
Exec
Queue
Exec
Queue
Exec
Queue
Exec

# Cumulative Flow Diagram



[3]

# Additional Considerations

- Focus on system, not individual performance.
- Kanban as a change management tool – evolution not revolution.
- Achieve predictability by not asking developers to predict!
- Know your true WIP!

# Personal Software Manifesto?

| | |
|---|---|
| metric-based planning | estimation-based planning |
| story elaboration just-in-time | story elaboration through up front planning |
| value delivered via continuous flow | value delivered in small batches |

*While there is value in the items on the right, I value the items on the left more.*

# Summary

- Want to try Kanban?
  - Map your **value stream** and establish a **pull** system
  - **Limit work in process**
  - Make it **visible** for your team and customer

# Thank you!

Troy Tuttle

Project Lead Consultant, AdventureTech

Troy.Tuttle@adventuretechgroup.com

TroyLTuttle@gmail.com

AdventureTech

www.adventuretechgroup.com

blog.troytuttle.com

twitter.com/troytuttle

linkedin.com/in/troytuttle

# References

(1) Karl Scotland – http://availagility.wordpress.com and KanbanDev Yahoo! Group.

(2) Eric Willeke - http://manicprogrammer.com/cs/blogs/willeke/default.aspx and KanbanDev Yahoo! group

(3) David Peter Joyce - **http://leanandkanban.wordpress.com** and "Pulling Value Lean and Kanban"

(4) Corey Ladas - Scrumban book, www.leansoftwareengineering.com and KanbanDev Yahoo! group.

(5) Jon Cook – Critical Chain yahoo list through Clarke Ching - www.clarkeching.com

(6) David Anderson – www.agilemanagement.net

(7) Kenji Hiranabe - http://www.infoq.com/articles/agile-kanban-boards