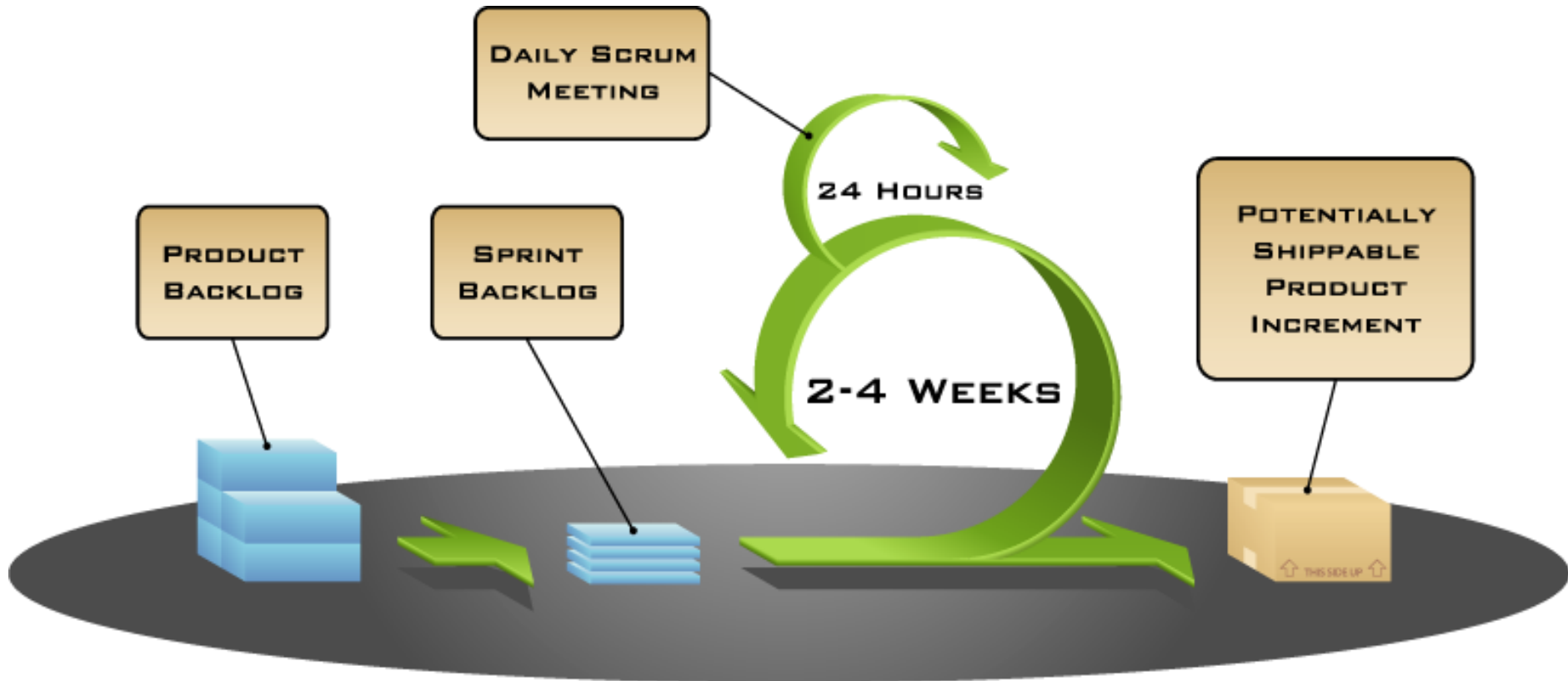


# THE SCRUM FRAMEWORK



COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE

# ROLES (1)

## Product Owner

- Represents the interests of all the stakeholders
- ROI objectives
- Prioritizes the product backlog



## Team

- Cross-functional
- Self-managing
- Self-organizing



# ROLES (2)

## Scrum Master

### The Scrum Master

Responsible for the scrum process

- Removes impediments
- Facilitates scrum events
- Facilitates communication



- **Serves the Team:**

- Helps the Team
  - to reach consensus for what can be achieved in a sprint/ during the daily scrum
  - to follow the agreed-upon rules for daily scrums
- Organizes sprint events
- Removes obstacles that are impeding the team's progress.
- Protects the team from outside impediments, disturbance
- Coaches the Team

- **Serves the PO:**

- Ensures that goals, scope, are understood by everyone in Team
- Helps the Team to understand the need for clear and concise Product Backlog items
- Ensures the Product Owner to know how to arrange the Product Backlog to maximize value

# ROLES (3)

## (Team) Coach

- Coaches the team in the Agile and Lean process
- Challenges the team for continuous improvement
- Ensures the following of Agile & Lean rules and practices
- Typically not a member of the Team, organization-wide responsibility
- Some similar (coaching) activities as SM, but at higher level



# USER STORIES AND ESTIMATION (1)

Describe requirements in product backlog

Syntax: As <role> I want to <requirement>  
because <business reason>

Example:

- As a customer I want to reserve movie tickets with my mobile
  - Because I want to be sure that I have a seat when I arrive to the theater

# USER STORIES AND ESTIMATION (2)

## Planning poker method

- Product owner (or a stakeholder with the best knowledge) explains the story
- The user story is divided to tasks
- Team members estimate the tasks/story independently and select a card
- They show the cards simultaneously
- Explain if estimates differ
- End or go back to step 2



# SPRINT PLANNING

## Time-box (eg. 2 hours)

- 1st - 1 hours max. for team to select Product Backlog and sets goal with Product Owner
- 2nd - 1 hours max. for team to define Sprint Backlog to build functionality

## Attendees

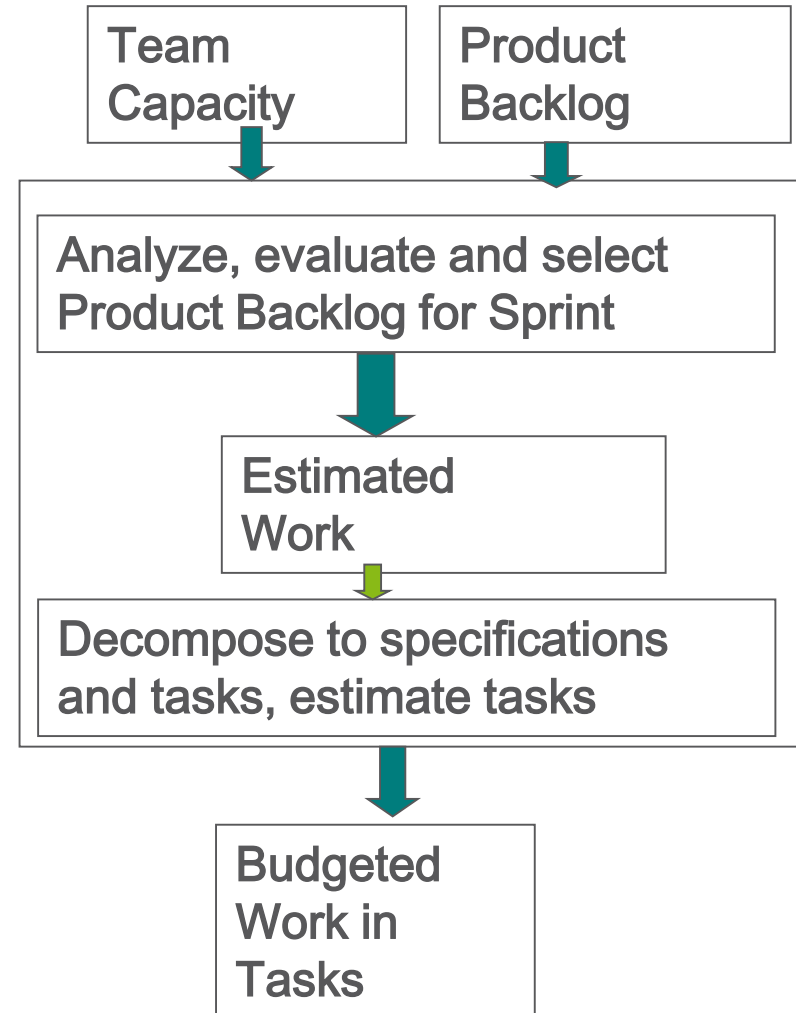
- Product owner, Team and Scrum Master

## Product Owner must prepare the Product Backlog prior to the meeting

- Product owner decides what the product backlog constitutes

## Output: Sprint backlog

- Tasks, task estimates, task assignments
- Cannot change during the sprint



# DEFINITION OF DONE (DoD)

## 10 POINT CHECKLIST

Code produced (all 'to do' items in code completed)

Code commented, checked in and run against current version in source control

Peer reviewed (or produced with pair programming) and meeting development standards

Builds without errors

Unit tests written and passing

Deployed to system test environment and passed system tests

Passed UAT (User Acceptance Testing) and signed off as meeting requirements

Any build/deployment/configuration changes implemented/documentated/communicated

Relevant documentation/diagrams produced and/or updated

Remaining hours for task set to zero and task closed



# TRANSPARENCY – TASK BOARD

NOT CHECKED OUT	CHECKED OUT	DONE! :D)	SPRINT GOAL: BETA-READY RELEASE!
<p><b>MIGRATION TOOL</b></p> <p>Tool migration 2d</p>	<p>GUI spec 2d</p> <p>Topology write 1d 2d</p>	<p><b>DEPOSIT</b></p> <p>Write failing test 2d</p> <p>Code cleanup 1d</p> <p>Integr. test 2d 3d</p> <p>DB design 2d 3d</p> <p>DAD</p>	<p><b>BURNDOWN</b></p>
<p><b>BACKOFFICE LOGIN</b></p> <p>Integr. with phone 2d</p> <p>Draft GUI 1d</p>	<p>Write failing test 2d</p>	<p>Write failing test 2d 3d</p>	<p><b>UNPLANNED ITEMS</b></p> <p>Fix memory leak (JIRA 123) 2d</p> <p>Sales support? 2d</p> <p>Write whitepaper 4d</p>
<p><b>BACKOFFICE USER ADMIN</b></p> <p>Write failing test 2d</p> <p>GUI design (CSS) 2d</p> <p>Clarify requirements 2d</p> <p>Draft GUI 4d</p>			<p><b>NEXT</b></p> <p><b>WITHDRAW</b></p>

# DAILY / STAND-UP MEETING

- › Daily Scrum standing at task board
- › A Daily Scrum is a:
  - Daily max. 15 minute work meeting;
  - Same place and time every day;
  - Everyone answers three questions;
    - › What have you done since last meeting?
    - › What will you do before next meeting?
    - › What is the obstacles in your way (if any)?

# RETROSPECTIVES

## Set the stage

- Focus for this retrospective

## Gather data

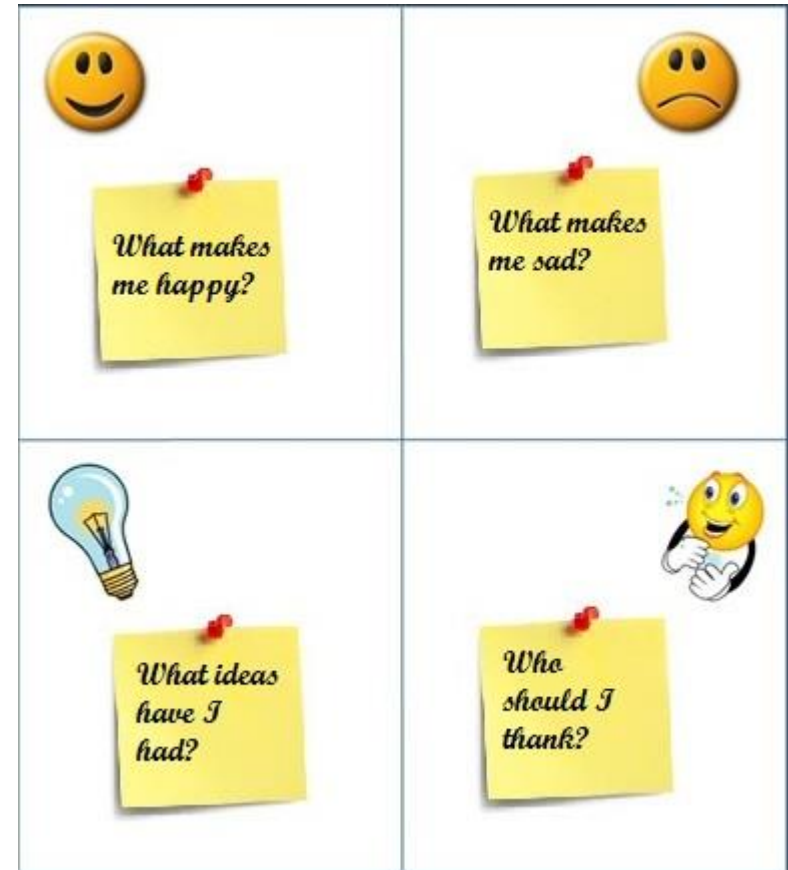
- Ground it in facts, not opinions

## Generate insights

- Observe patterns

## Decide what to do

- Move from discussion to action



# SCRUM, SUMMARY (BY ISTQB)

## Practices

- Sprint (Iteration)
- Product increment
- Product's backlog
- Definition of Done (DoD) – exit criteria
- Timeboxing – fix duration for iteration, fix daily meetings
- Transparency

No specific software development techniques

## Roles

- Scrum Master (SM) ensures practices and rules are implemented, followed – process focused scrum theory
- Product Owner (PO) represents the customer and owns product backlog – he/she can change product backlog any time
- Development Team (3-9, self-organized) develops and tests product

Scrum does not prescribe testing approach

## 看板 – KANBAN CARDS LIMIT EXCESS WORK IN PROGRESS

看板 – kanban literally means “visual card,” “signboard,” or “billboard.”

Toyota originally used Kanban cards to limit the amount of inventory tied up in “work in progress” on a manufacturing floor

kanban cards act as a form of “currency” representing how WIP (Work In Progress) is allowed in a system.

Kanban is an emerging process framework that is growing in popularity since it was first discussed at Agile 2007 in Washington D.C.

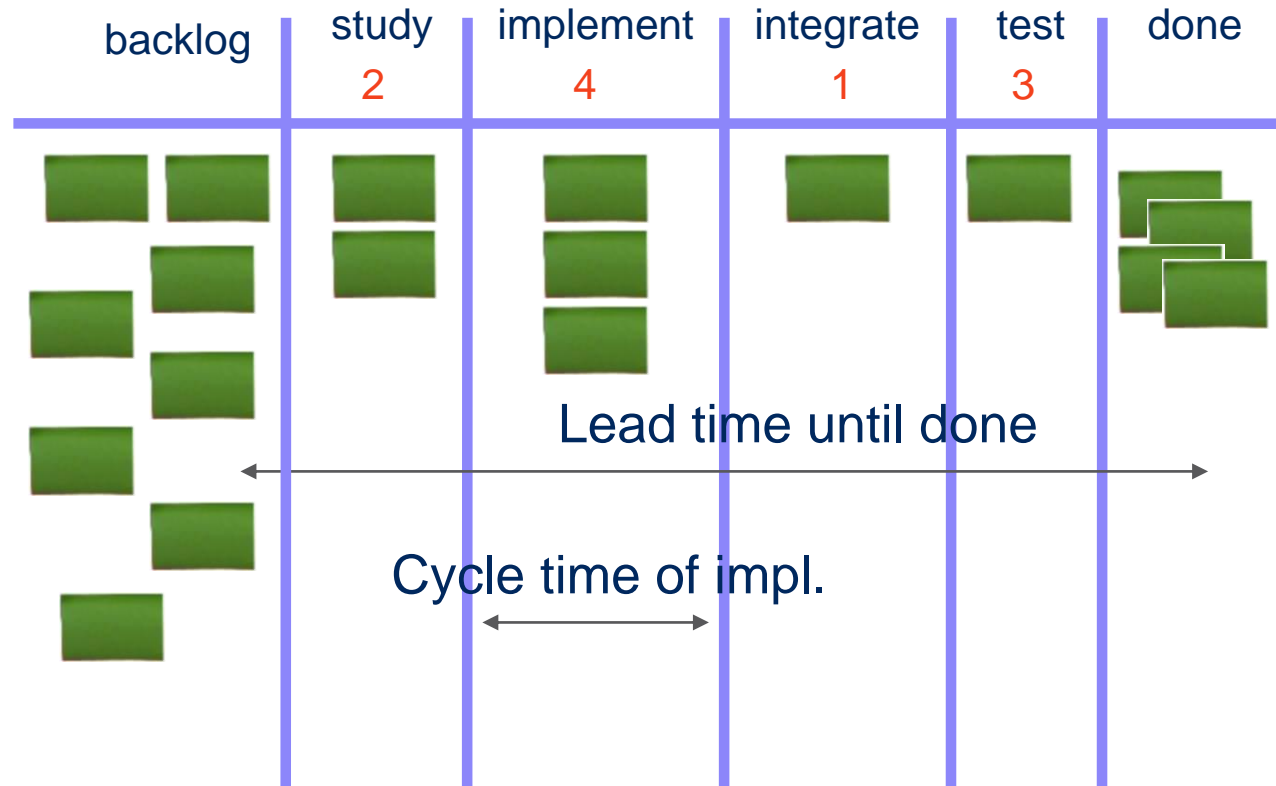


# KANBAN BASIC RULES

Visualize the workflow

Limit Work In Progress (WIP)

Measure and optimize lead time

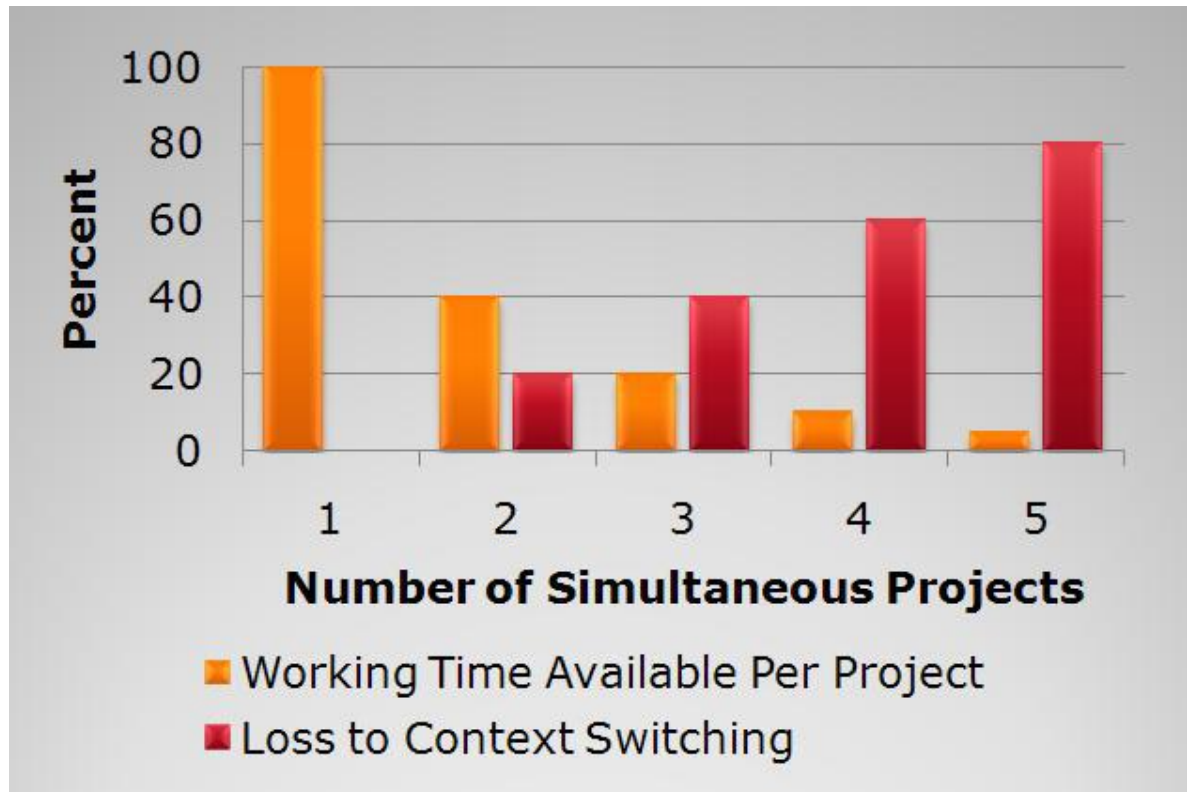


Columns: number, titles according to the need

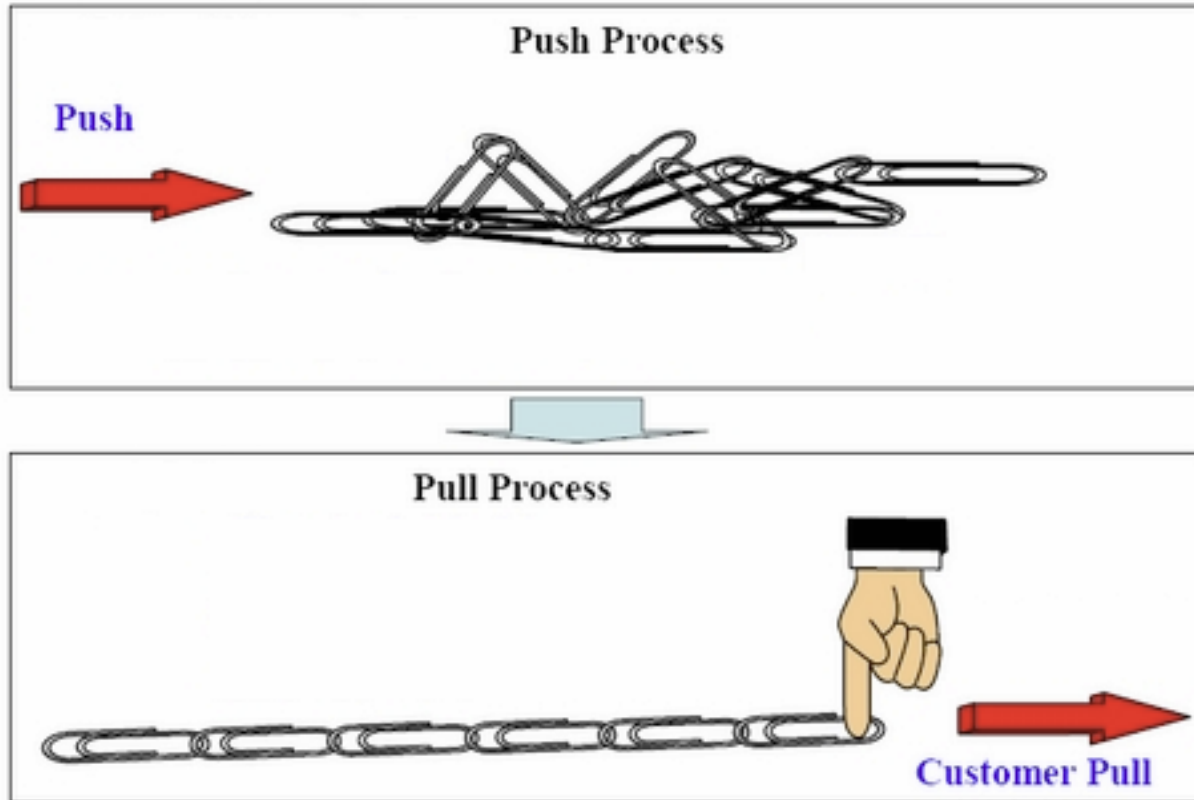
Defined when a task can be moved to next column

# LIMITING WORK IN PROGRESS

20% time is lost to context switching per task, so fewer tasks means less time lost (from *Gerald Weinberg, Quality Software Management: Systems Thinking*)



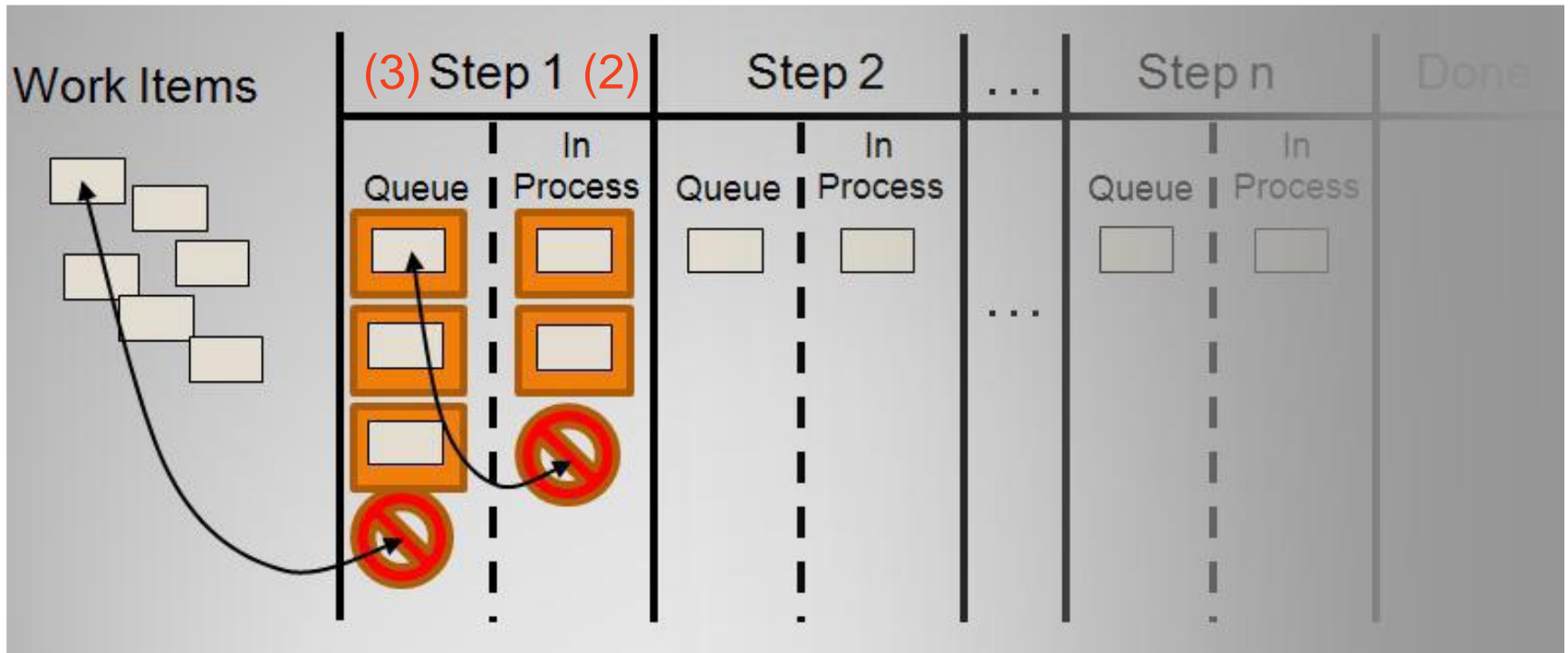
# LIMITING WORK IN PROGRESS



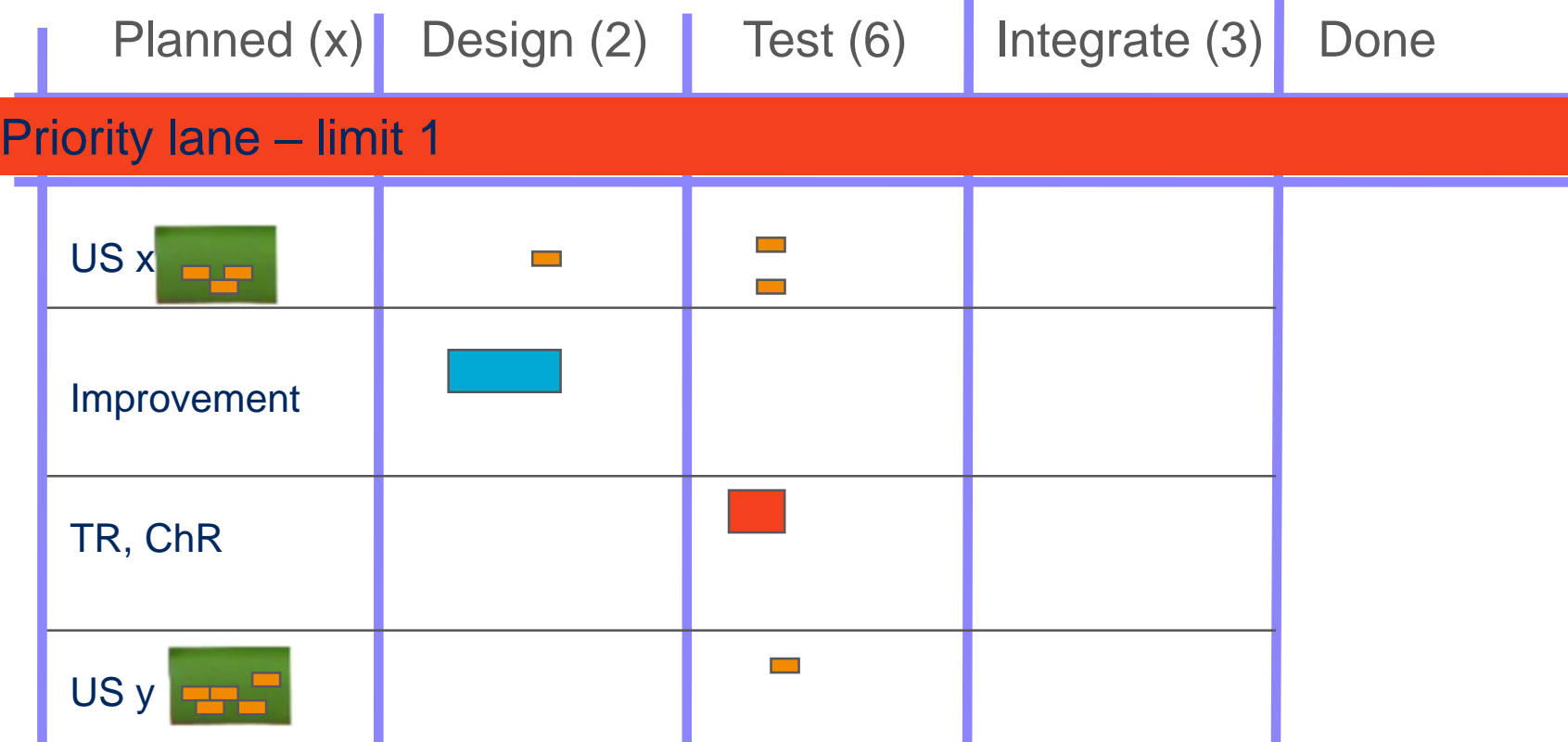


# LIMITING WORK IN PROGRESS

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



# LIMITING WIP – EMERGENCY LANE



# METRICS

Metrics are a tool for **everybody**

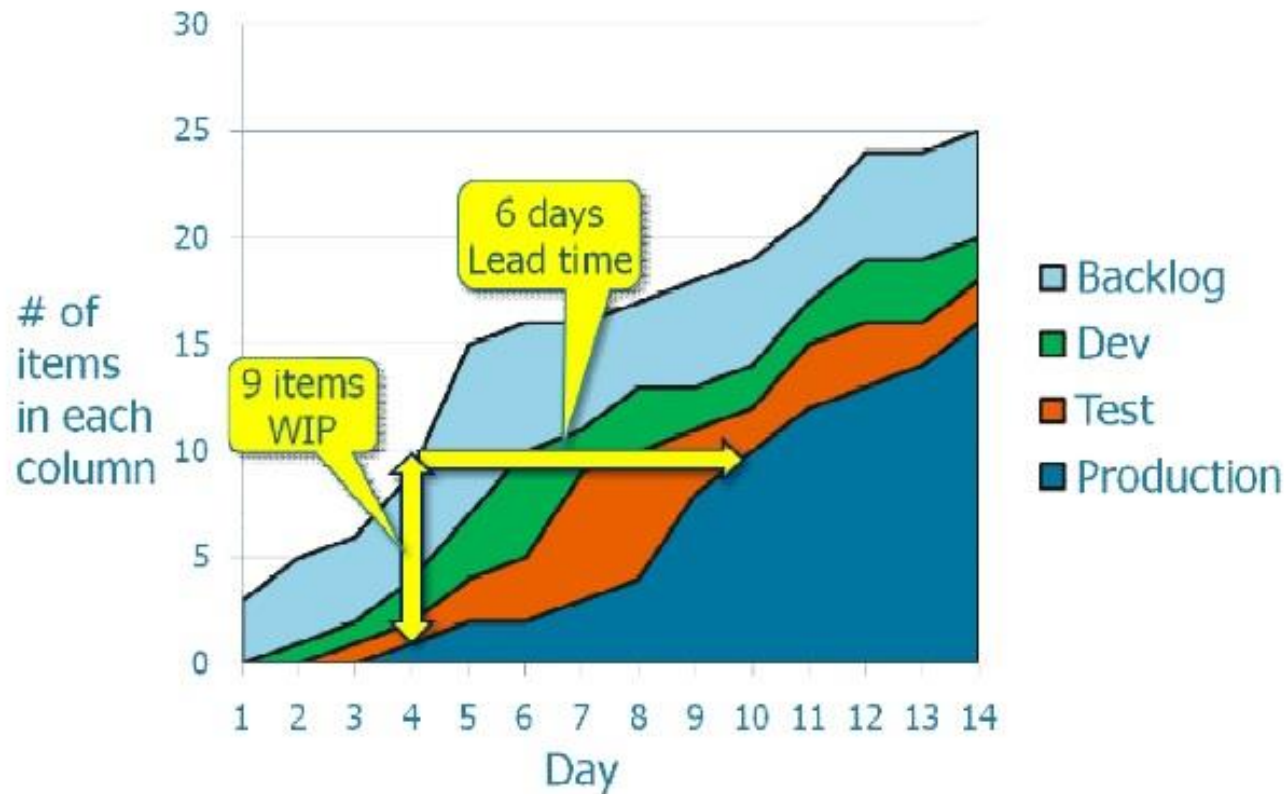
The **team** is **responsible** for its metrics

Metrics allow for **continuous improvement**

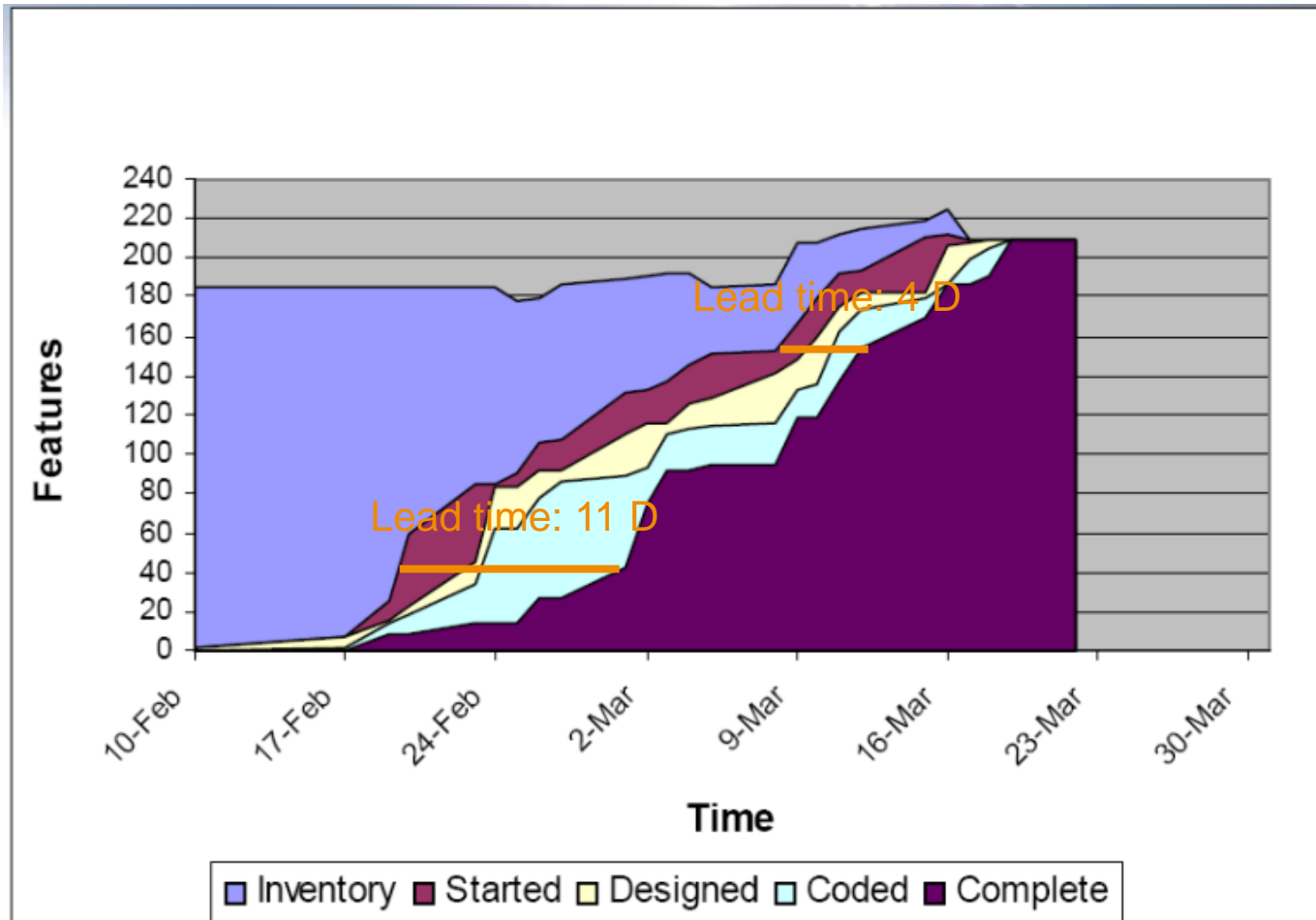
Manage **quantitatively** and **objectively** using only a few simple metrics

- Quality
- Work in Process
- Lead / Cycle time
- Waste / Efficiency
- Throughput

# USE CUMULATIVE FLOW DIAGRAMS TO VISUALIZE WORK IN PROGRESS



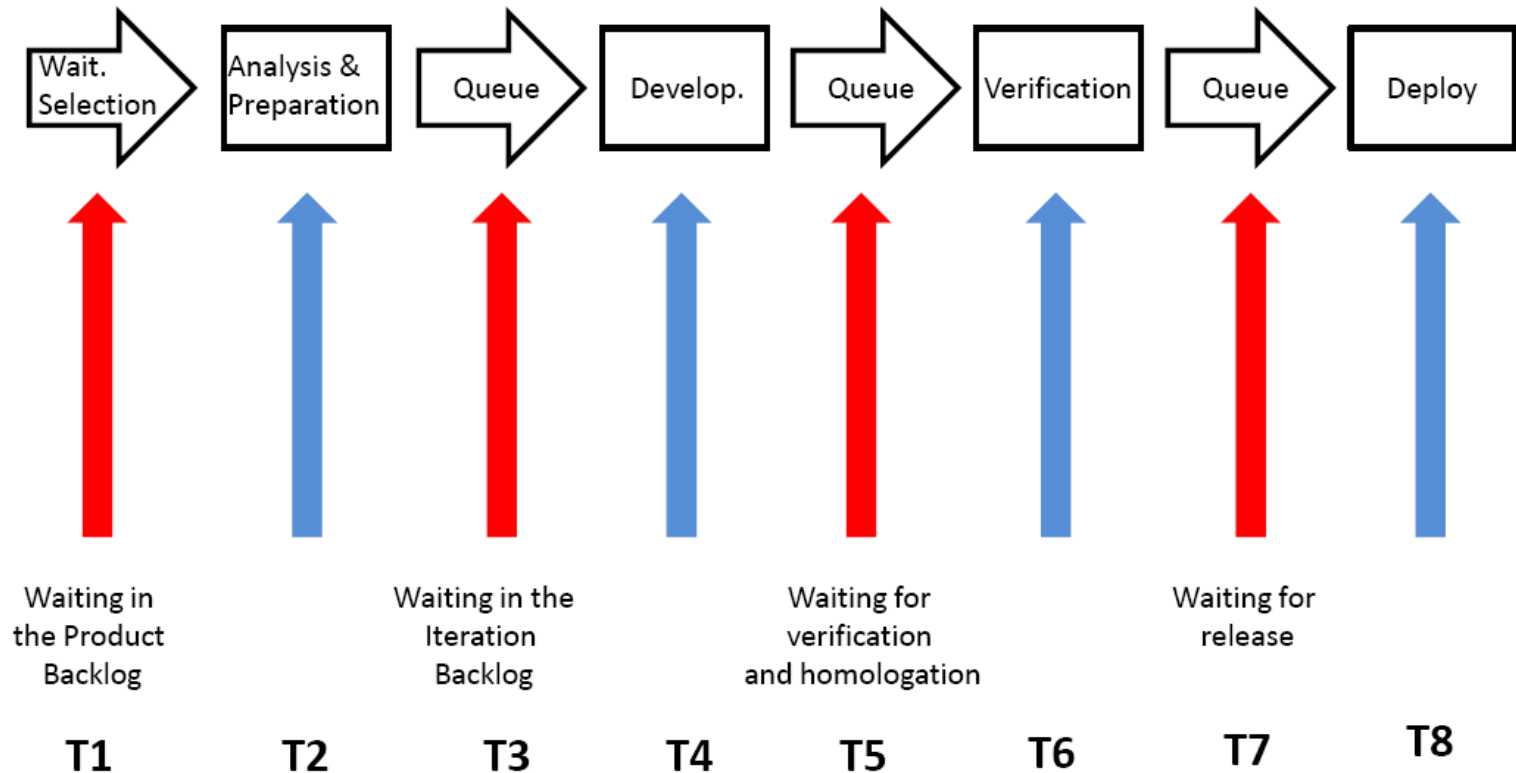
# USE CUMULATIVE FLOW DIAGRAMS TO VISUALIZE WORK IN PROGRESS



[www.agilemanagement.net/Articles/Papers/BorConManagingwithCumulat.html](http://www.agilemanagement.net/Articles/Papers/BorConManagingwithCumulat.html)

# VALUE STREAM MAPPING









Value Stream for Product XYZ



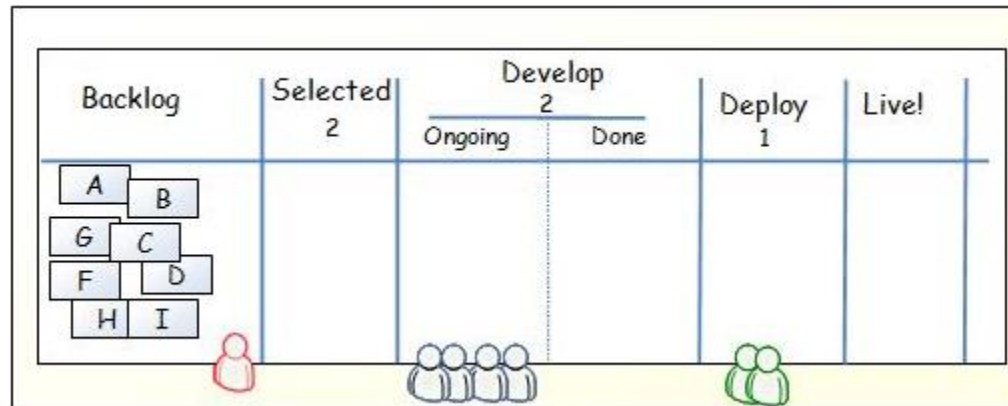
$$\text{Efficiency (\%)} = \frac{\text{TValue} * 100}{\text{TValue} + \text{TWaste}}$$

# PRIORITIZATION METHOD

- › Business value?
- › Cost of delay classification

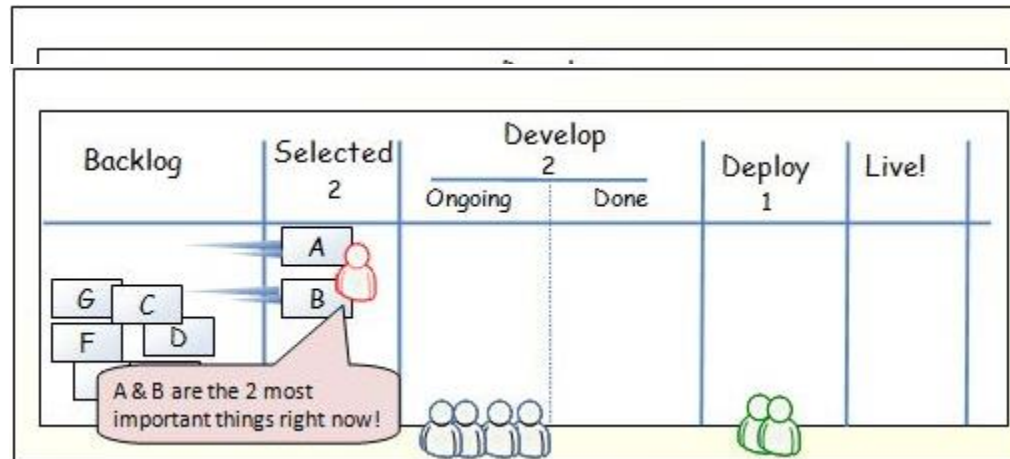
		<b>Expedite:</b> critical, and immediate cost of delay; can exceed other kanban limit (bumps other work) 1 <sup>st</sup> priority
		<b>Fixed date:</b> cost of delay goes up significantly after deadline; 2 <sup>nd</sup> priority
		<b>Accelerating:</b> cost of delay goes up increasingly over time; 3 <sup>rd</sup> priority
		<b>Normal:</b> Cost of delay linear over time; 4 <sup>th</sup> priority

# ONE DAY IN KANBAN LAND

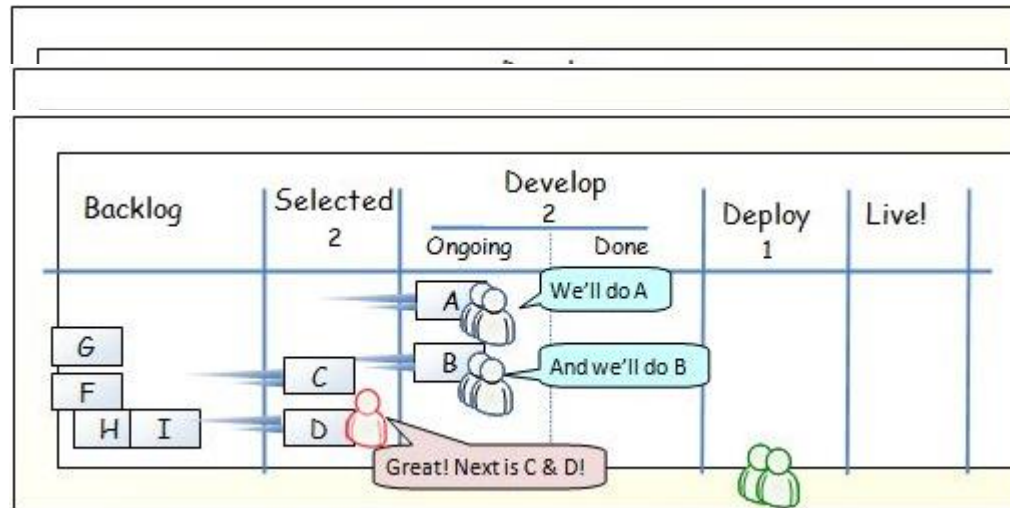




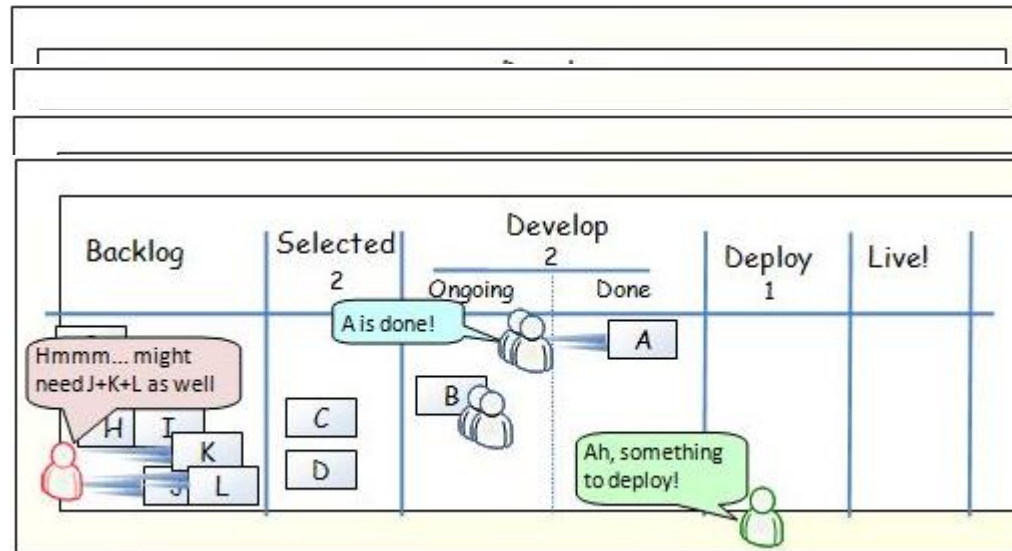
# ONE DAY IN KANBAN LAND



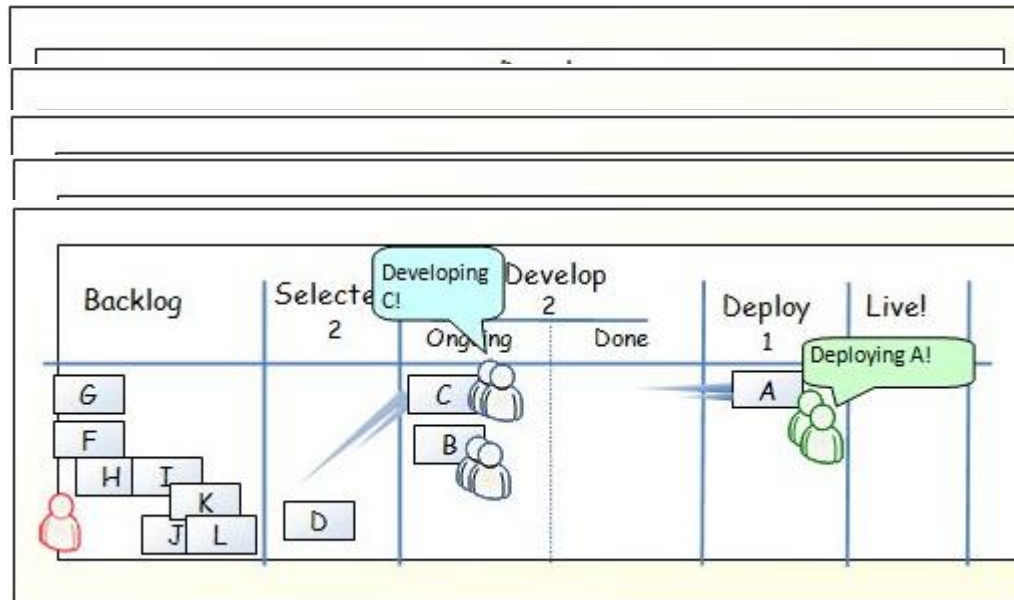
# ONE DAY IN KANBAN LAND



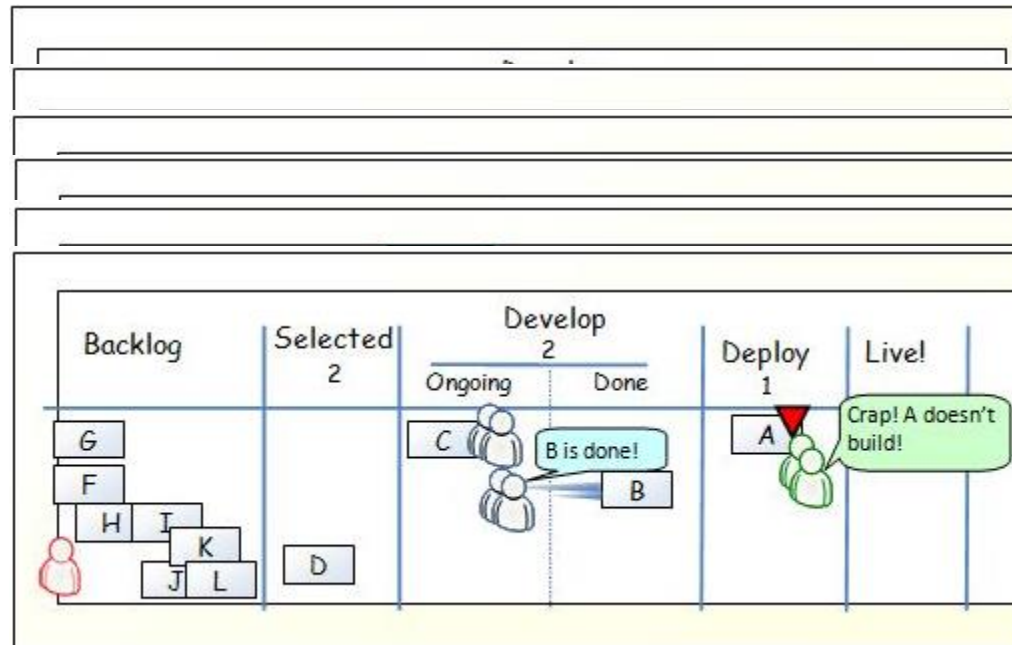
# ONE DAY IN KANBAN LAND



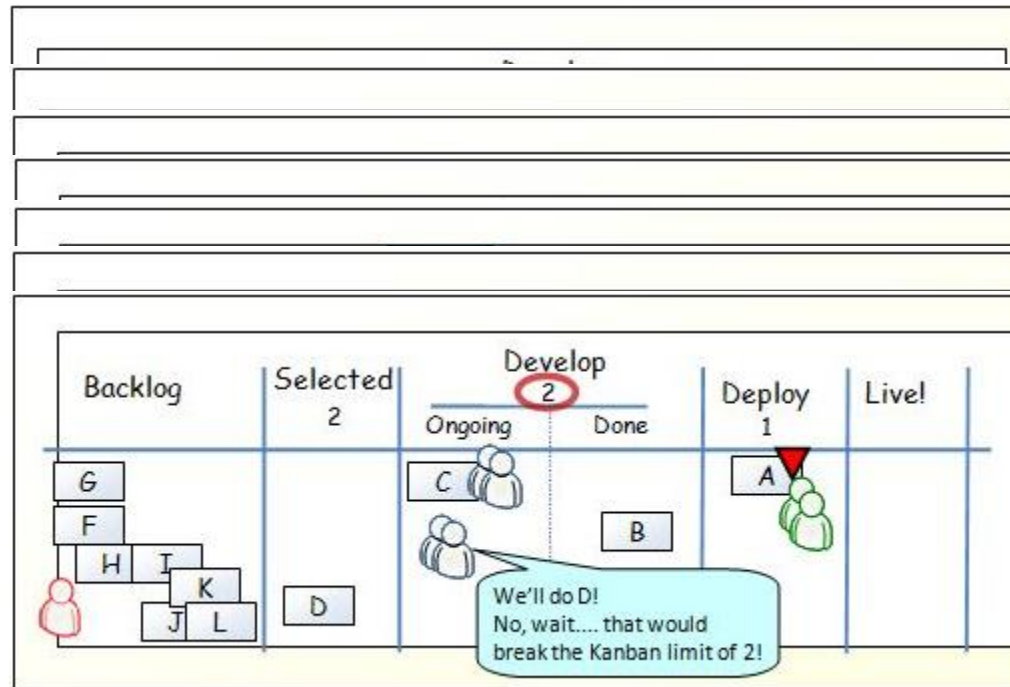
# ONE DAY IN KANBAN LAND



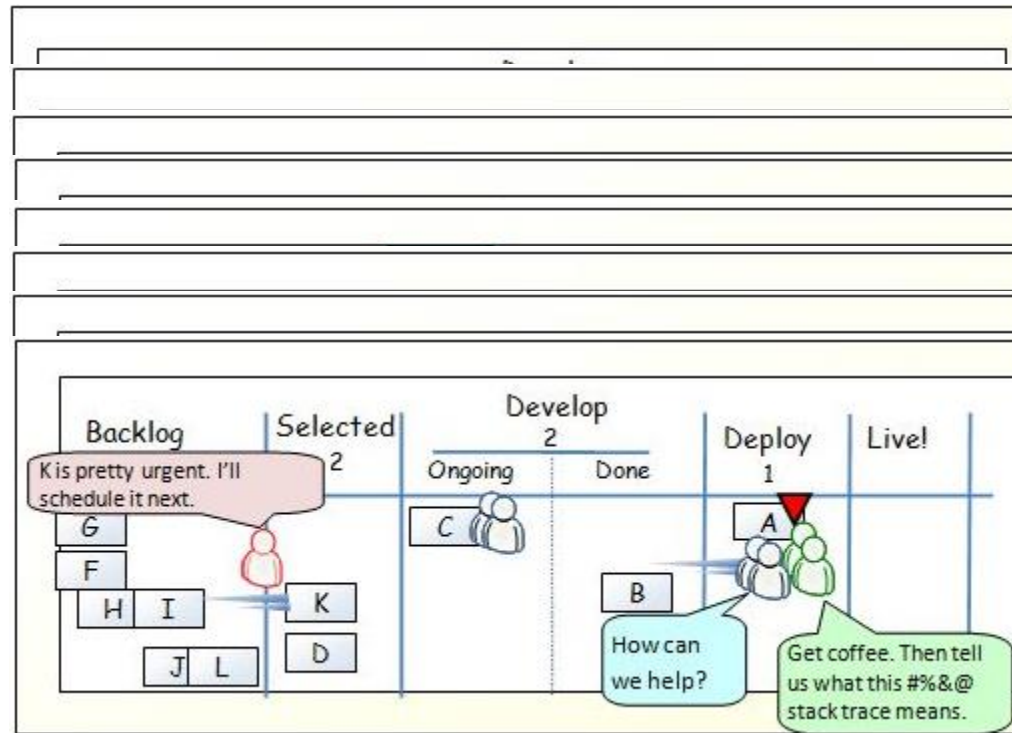
# ONE DAY IN KANBAN LAND



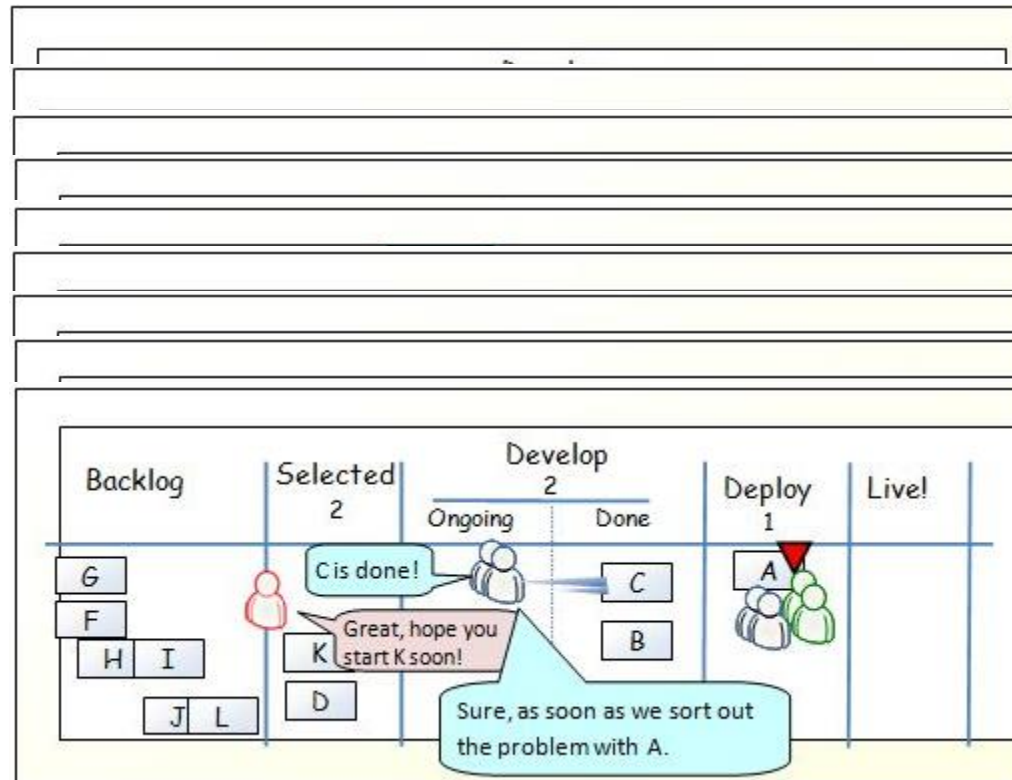
# ONE DAY IN KANBAN LAND



# ONE DAY IN KANBAN LAND

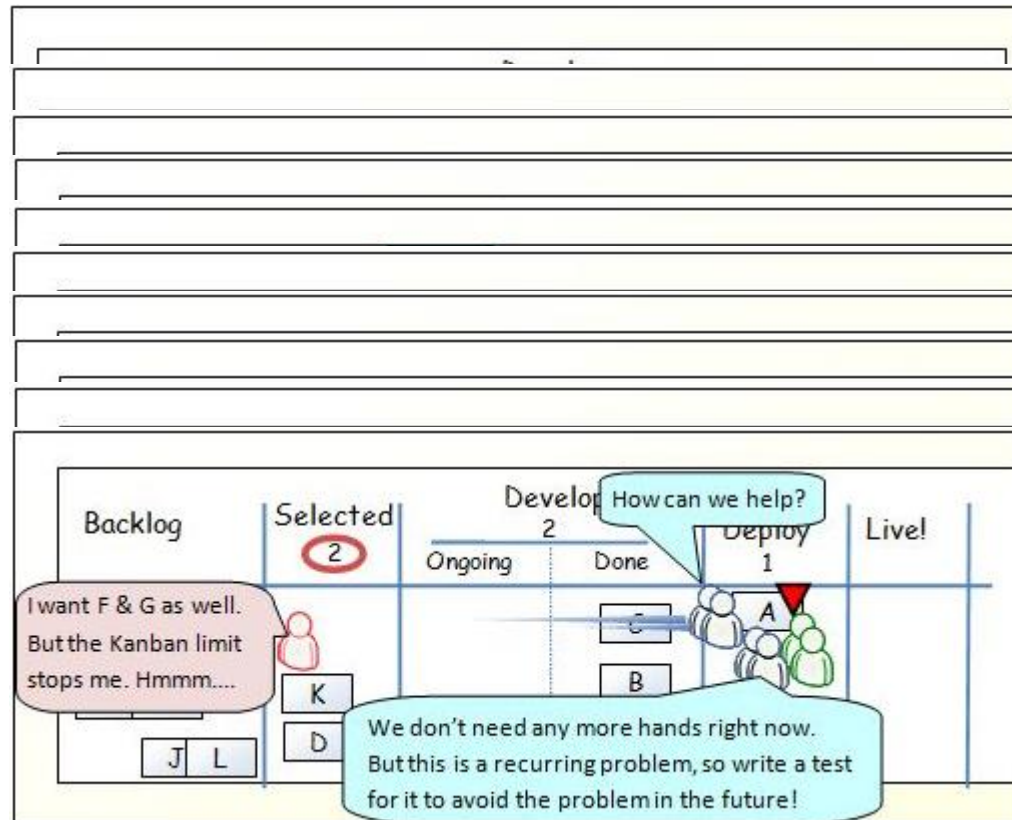


# ONE DAY IN KANBAN LAND

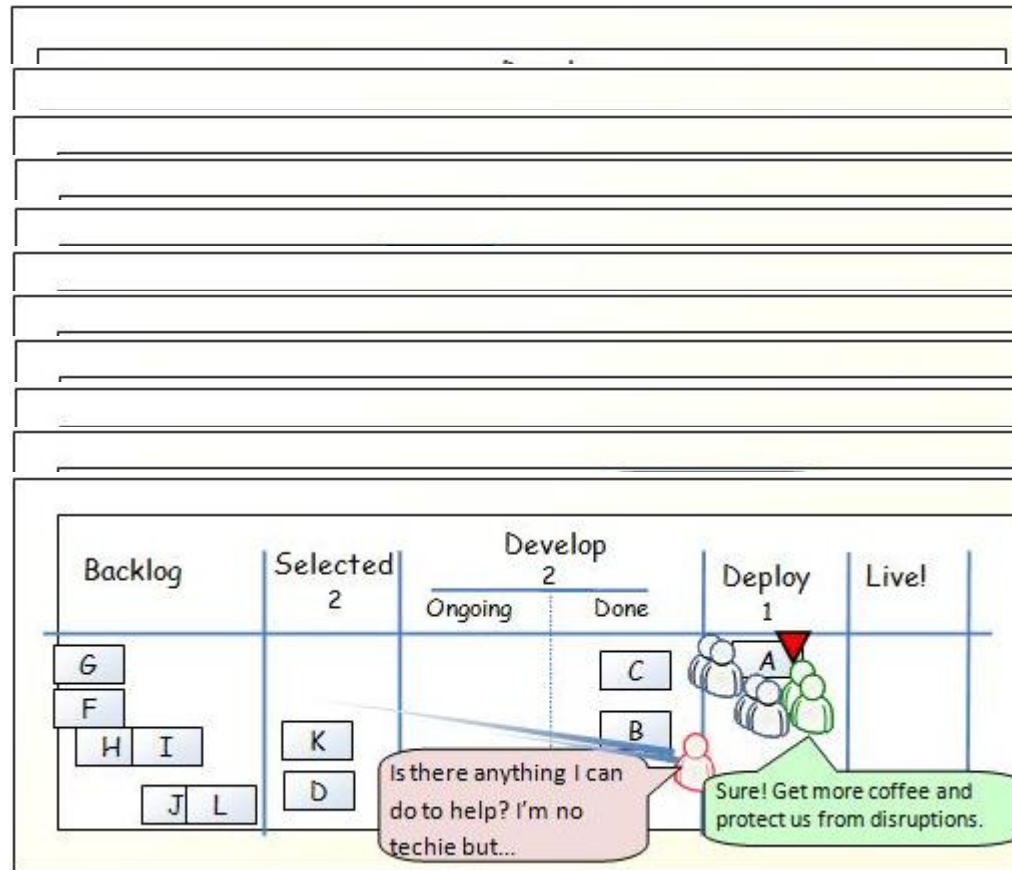




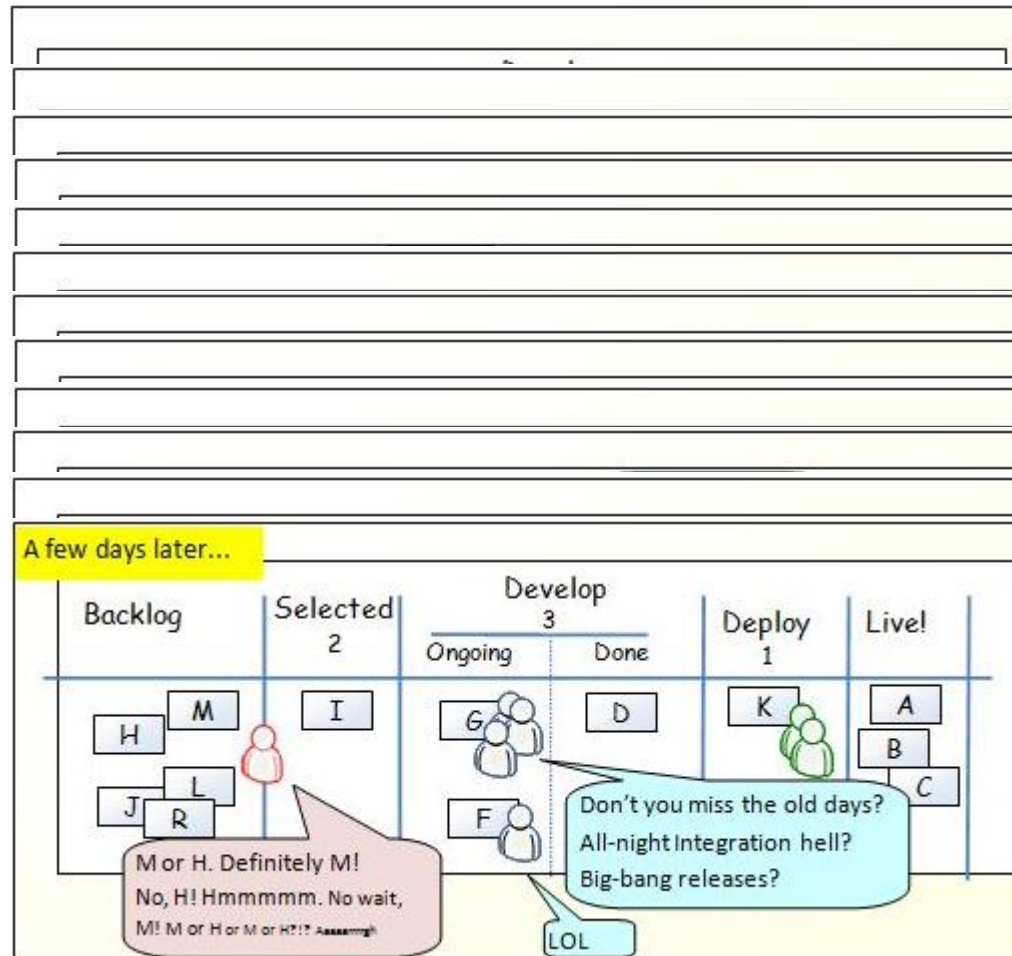
# ONE DAY IN KANBAN LAND



# ONE DAY IN KANBAN LAND



# ONE DAY IN KANBAN LAND



# AFTER A KANBAN IMPLEMENTATION...

“Nothing else in their world should have changed. Job descriptions are the same. Activities are the same. Handoffs are the same. Artifacts are the same. Their process hasn't changed other than you are asking them to accept a WIP limit and to pull work rather than receive it in a push fashion”

**David Anderson.**

# SOURCES

- › <http://www.limitedwipsociety.org/>
- › <http://www.crisp.se/henrik.kniberg/kanban-vs-scrum.pdf>

# KANBAN SUMMARY (BY ISTQB)

Optimize flow of work in value-added chain

Instruments:

- Kanban board
- Work-in-progress (WIP) limit
- Lead time

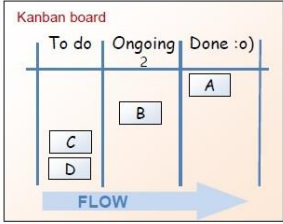
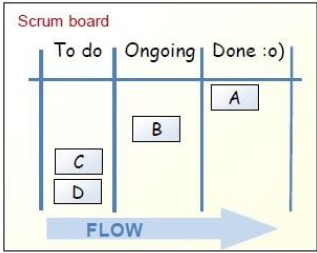
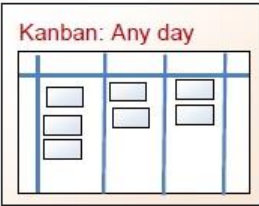
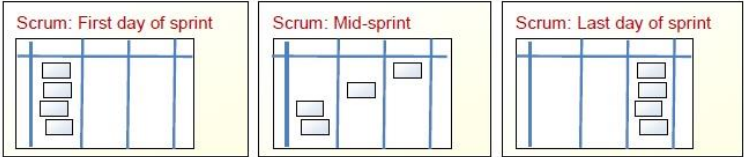
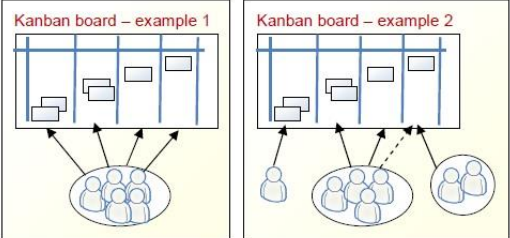
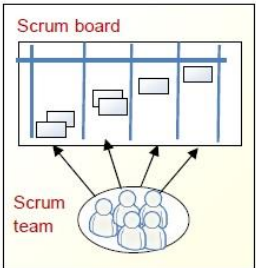
Both Kanban and Scrum provide status transparency and backlogs, but:

- Iteration is optional in Kanban
- Items can be delivered one at a time or in a release
- Timeboxing is optional

# SCRUM VS KANBAN (1)

Kanban		Scrum
No prescribed roles	Roles	Pre-defined roles of Scrum Master, Product Owner and team member
Continuous Delivery	Delivery	Timeboxed sprints, delivery at the end (if PO approves)
Work is pulled through the system (single piece flow)	Flow	Work is pulled through the system in batches (the sprint backlog)
Changes can be made at any time	Changes	No changes allowed mid-sprint
Cycle Time Lead Time	Key Metrics	Velocity
More appropriate in operational environments with a high degree of variability in priority	Where fits?	More appropriate in situations where work can be prioritized in batches that can be left alone

# SCRUM VS KANBAN (2)

Kanban		Scrum
<p>Limits WIP per workflow state (directly)</p> 	<p>WIP</p>	<p>Limits WIP per iteration (indirectly)</p> 
<p>Continuous</p> 	<p>Board</p>	<p>Reset between each iteration</p> 
<p>Optional</p>	<p>Esti- mation</p>	<p>Prescribed</p>
<p>Not requested</p> 	<p>Teams</p>	<p>Requested</p> 



# DOCUMENTATION SYSTEMS

Helps in generating on-line documentation or offline reference manual from documented source files.

Combine source code with documentation and other reference materials

Make it easier to keep the documentation and code in sync

We will see:

- Doxygen
- Javadoc
- T3Doc

# DOXYGEN

Source code documentation generator tool, Doxygen is a documentation system for C++, C, Java, Objective-C, Python, IDL (Corba and Microsoft flavors), Fortran, VHDL, PHP, C#, and to some extent D.

## Most useful tags:

- `\file`
- `\author`
- `\brief`
- `\param`
- `\returns`
- `\todo` (not used in assignments)

# JAVADOC

Attach special comments, called *documentation comment* (or *doc comment*) to classes, fields, and methods. `/** ... */`

Use a tool, called *javadoc*, to automatically generate HTML pages from source code.

Javadoc Tags: Special keyword recognized by javadoc tool. Common Tags:

- @author Author of the feature
- @version Current version number
- @since Since when
- @param Meaning of parameter
- @return Meaning of return value
- @throws Meaning of exception
- @see Link to other feature

# T3DOC

## TTCN-3 source code tagging

Standard: ETSI ES 201 873-10

## Example

```
• /*****  
  ** @desc XYZ                               **  
  ** Initialize to pre-trial defaults.       **  
  **                                         **  
  123  
  *****/
```

# TTCN-3 DOCUMENTATION TAGS

	Simple Data Types	Structured Data Types	Component Types	Port Types	Modulepars	Constants	Templates	Signatures	Functions (TTCN-3 and external)	Altsteps	Test Cases	Modules	Groups	Control Parts	Component local definitions	Used in implicit form (see clause 7)	Embedded in other tags
<a href="#">@author</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
<a href="#">@config</a>											X						
<a href="#">@desc</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
<a href="#">@exception</a>								X								X	
<a href="#">@member</a>		X <sup>1</sup>	X	X	X <sup>1</sup>	X <sup>1</sup>	X <sup>1</sup>									X	
<a href="#">@param</a>							X	X	X	X	X					X	
<a href="#">@priority</a>											X						
<a href="#">@purpose</a>											X	X					
<a href="#">@remark</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
<a href="#">@reference</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
<a href="#">@requirement</a>									X	X	X	X					
<a href="#">@return</a>								X	X							X	
<a href="#">@see</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
<a href="#">@since</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
<a href="#">@status</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
<a href="#">@url</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
<a href="#">@verdict</a>									X	X	X	X					
<a href="#">@version</a>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		

NOTE: <sup>1</sup> Preceding language elements of record, set, union or enumerated types only.