Managing with Kanban
Come rivoluzionare l’ufficio
e vivere contenti
What?

Overview of DFP platform (and my TEAM)
Condensed scrum
Condensed Kanban
Question and answer
Me

Dfp Core team Technical Manager
Lean manager in becoming certified Scrum Master Kanban (upper case K) enthusiast
Future crossfitter Hermitie husband (2013)
My TEAM (Dfp Core)
Our Mission:

Maintain and evolve the:

Data
Flow
Platform
Enter the DFP

Too Easy!

Jboss
Java
Groovy
REAL LIFE!

4 FLOWS EACH COUNTRY:
• biblio
• abstract
• full text
• citation
• public
• secret

Production support
• bug fixing
• monitoring

NPL FLOWS:
• images
• pdf

CPC Classification:
• reorganization

4 different projects
• dfp-admin
• dfp-console
• dfp-core
• processes

4 different external teams

Production support
• bug fixing
• monitoring

Special assignments
Priorities changing almost daily

6 different product owners

Different level of experience inside the team
Scrum approach

product owner

Team

Scrum master
Sprint

Produce

Sprint backlog

Mantains

Monitors

Planning poker

Backlog
How it works
Monitors 1/2

Stand up

Board
Monitors 2/2

<table>
<thead>
<tr>
<th>Previous Velocity</th>
<th>Velocity</th>
<th>Work capacity</th>
<th>Total Commitment</th>
<th>Focus Factor</th>
<th>Adopted Work</th>
<th>Found Work</th>
<th>Target Value Contributions</th>
<th>Accuracy of estimation</th>
<th>Accuracy of Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>25</td>
<td>45</td>
<td>32</td>
<td>55.56%</td>
<td>40.63%</td>
<td>53.13%</td>
<td>100.00%</td>
<td>28.13%</td>
<td>43.84%</td>
</tr>
</tbody>
</table>
Real life strikes again!
Acceptance did not (always) depend on us
Demo was not taken in account while planning
External request always accepted
Board does not reflect reality
Managers tends to keep people busy, not to have work done
TEAM does not scale
Different types of activities:
• development
• bug fixings
• production monitoring
• support for other teams

Different type of constraint:
• production never waits
• some deadlines does not wait
• some deadlines can wait
Consequences

if (end_of_sprint && !task.done?)
  task.drop
end

the TEAM was demotivated and ALWAYS late
Customers do not understand points
Customers always angry
We need to do something!
Kanban and coffee

Upper case

kanban

Lower case

Done
More in depth

• kanban
• workflow (visualized)
• limited work in progress
• process are explicit
• measurement and management of the flow
• (?) using of models to recognize improvement opportunities (TOC, muda,...)
What about software?
<table>
<thead>
<tr>
<th>Jira ID</th>
<th>Class of Service</th>
<th>Size</th>
<th>Other info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1905</td>
<td></td>
<td></td>
<td>Run PT in INTG Talk with Fabrice to check the files</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5/20/2012</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Small</td>
</tr>
</tbody>
</table>
WIP

80
Manage and measure flow

Tickets per week
Cumulative flow chart
Cycle/Lead time
Explicit process and Rules

Expedite Class
Fixed date delivery class
Standard class
Intangible class
External class
How to assign a COF

- Normal: Cost increases linearly with time.
- Expedite: Constant cost regardless of time.
- Fixed Date: Cost jumps at a specific time.
- Intangible: Cost increases exponentially over time.
Class of service AND cycle time

StandardClass  69%
IntangibleClass  12%
ExternalClass  2%
Expedite  7%
FixedDateDelivery  10%

2009: 67
2010: 59
2011: 41
2012: 23
How do I implement it?
Visualize the flow
Set the WIP limit
Find the bottleneck
Exploit the bottleneck

Increase the limit
:goto 3
WARNING

NEVER EVER BREAK THE WIP LIMIT!
Why?

Create a pull system
Create slack time
## Pull system

<table>
<thead>
<tr>
<th>Ready 12</th>
<th>Analysis 4</th>
<th>Dev 4</th>
<th>RFA 8</th>
<th>Acceptance 4</th>
<th>Build 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In Progr</td>
<td>Done</td>
<td></td>
<td>In Progr</td>
<td>Done</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In Progr</td>
<td>Done</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In Progr</td>
<td>Done</td>
</tr>
</tbody>
</table>

---

1. Ready 12: Status shows the number of tasks in progress.
2. Analysis 4: Tasks in analysis stage.
3. Dev 4: Tasks in development stage.
4. RFA 8: Tasks in request for approval stage.
5. Acceptance 4: Tasks in acceptance stage.
6. Build 8: Tasks in build stage.

---

Note: The table above represents a pull system status with tasks in various stages of completion.
### The M/M/1/k Queue

<table>
<thead>
<tr>
<th>WIPCAP</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>Infinite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Cycle Time</td>
<td>1.0</td>
<td>1.5</td>
<td>2.8</td>
<td>4.6</td>
<td>7.2</td>
<td>10.0</td>
</tr>
<tr>
<td>Time in Queue</td>
<td>0</td>
<td>0.5</td>
<td>1.8</td>
<td>3.6</td>
<td>6.2</td>
<td>9.0</td>
</tr>
<tr>
<td>Utilization Percent</td>
<td>47%</td>
<td>63%</td>
<td>79%</td>
<td>85%</td>
<td>89%</td>
<td>90%</td>
</tr>
<tr>
<td>Empty Percent</td>
<td>53%</td>
<td>37%</td>
<td>21%</td>
<td>15%</td>
<td>11%</td>
<td>10%</td>
</tr>
<tr>
<td>Blocking Percent</td>
<td>47%</td>
<td>30%</td>
<td>13%</td>
<td>5%</td>
<td>1%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Note: Assumes 90 percent utilization.

WIP constraints tradeoff reductions in cycle time against blocking and underutilization costs.
The Economics of Queues

To Maximize Profits, Minimize Total Cost

Dollars

Total Cost

Cost of Excess Capacity

Cost of Delay

Excess Product Development Resource

How many product developers can draw the red curve?
Slack time (is not evil)

Improve the process
Improve yourself
Automate
Manage emergency
Do important but not urgent things (Intangible class)
Spend time on FB...
What we have done during slack time...

Cleaned up the database
Migrated application to new version of grails and TB
Revamped rollback for NPL
Closed some memory leaks
Throw balls at each other and drink coffee together
Consequences

tasks.each do |task|
  task.work_until_done
end
TEAM is not late anymore...
customers are happy (not always)
We are pulling from the customers
Why you should use it

Easy to Implement (you need only post-it)
Works out of the box (docdb)
Scales from small to big team (from 1 to 50)
Organization of work comes for free (even with distributed team and resources)
Continuos improvement (state of mind)
Swarming in problem
Work is finished when is finished!
Deliver value to customer
Clients do understand days
dank je wel
Links and Resources

http://scrumalliance.org
http://rapidscrum.com/MetricsVideos.php
http://www.agilemanagement.net

The Definitive Guide! KANBAN, Successful Evolutionary Change For Your Technology Business

www.lovadina.org
sayat.me/nicolalovadina
twitter.com/nicolalovadina
Personal Kanban

For a team of 1
Who said that Kanban board must be boring?