A Kanban System for Sustaining Engineering on Software Systems

David J Anderson
Senior Director Software Engineering

Rick Garber
Manager Process Engineering
Corbis is a Creative Services Company whose main business is licensing digital images

- World’s 2nd largest stock photography business
- Privately owned by Bill Gates
- Based in Seattle, USA
- Represents ~3500 professional photographers
- Sells image rights to publishers, advertising agencies and corporations for use in print and online media
Major IT system releases were too infrequent to provide sufficient business agility

- Interval between major releases was 3 months and growing
- New major projects were even larger; some planned to take 18 months
- Sustaining process was funded by Governance committee providing 10% more headcount in relevant functions
- Goal was to deliver a minor release (or upgrade) every 2 weeks
A dedicated maintenance team was not viable given the wide range of systems and the specialist nature of business and technical resources required.

- Sustaining effort had to pull from a floating pool of resources working on major projects.
- Sustaining work had to be scheduled around major project work.
- Middle-management needed to show that the 10% funded resources were being utilized on sustaining work.
Managing each minor release as a mini-project didn’t work

- Transaction costs of negotiating scope and developing a schedule for each release was onerous.
- Line management, individual contributors and middle managers spent up to 2 weeks negotiating a plan for a release.
  - Implication was that 50% capacity was being burned on transaction costs.
  - Impact was extending beyond 10% or resources and reducing productivity on major projects.
  - Sustaining releases were not happening regularly.
  - By early September 2006 there hadn’t been a sustaining release for 2 months.
SLA = 21 calendar days from date added to "Engineering Ready" queue to date in Production.

"Engineering Ready" queue will have a limit; item(s) can be added to it only when (1) a slot is open and (2) a release is scheduled within 21 days; item(s) added to the queue can be change requests or bugs.

Releases will be scheduled every two weeks.

Software Engineering - Managed
Kanban board and daily standup meeting were introduced in early February to add a sense of urgency and team collaboration.

- More personal responsibility and accountability
- Resulted in better visual control
- Enabled more self-organization

- Less management supervision
- Better productivity
- Spontaneous quality circles and frequent Kaizen events
Look how the board has changed by March! Empirically adjusted Kanban limits and much neater presentation – team pride showing through
And again in April, more changes to Kanban limits and forward extension of the process to business analysis.
Waste bin spontaneously introduced by team to visually communicate rejected CRs that wasted energy and sucked productivity
A report was created to detail rejected or cancelled work items ("muda")

Rejected and Cancelled Work Items

<table>
<thead>
<tr>
<th>ID</th>
<th>Work Item Type</th>
<th>Title</th>
<th>Business Dept.</th>
<th>GTM-Related</th>
<th>Business Priority</th>
<th>Submitted Date</th>
<th>Approved Date</th>
<th>Closed Date</th>
<th>Reason</th>
</tr>
</thead>
</table>
And the process is spreading…
And the technique is being introduced to major projects with much longer time horizons. This example has a monthly “integration event” rather than a release.
More and more reports were demanded to facilitate management decisions. In this case, new reports to facilitate weekly prioritization.

### Proposed CRs - Pending Triage

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Business Dept.</th>
<th>Business Priority</th>
<th>GTM-Related</th>
<th>Deploy OnBy</th>
<th>Deploy Date</th>
<th>Date Submitted</th>
</tr>
</thead>
</table>

### Proposed CRs - Blocked or More Info Needed

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Business Dept.</th>
<th>Business Priority</th>
<th>GTM-Related</th>
<th>Deploy OnBy</th>
<th>Deploy Date</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1750</td>
<td>[Description]</td>
<td>Images</td>
<td>2 - High</td>
<td>Not related to GTM</td>
<td></td>
<td>More Info</td>
<td>1/19/2007</td>
</tr>
</tbody>
</table>

### Proposed CRs - Eligible for Requirements Ready

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Business Dept.</th>
<th>Business Priority</th>
<th>GTM-Related</th>
<th>Deploy OnBy</th>
<th>Deploy Date</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1567</td>
<td>[Description]</td>
<td>Media Services</td>
<td>2 - High</td>
<td>Not related to GTM</td>
<td></td>
<td>Trigated</td>
<td>1/10/2007</td>
</tr>
<tr>
<td>1442</td>
<td>[Description]</td>
<td>Customer Experience</td>
<td>3 - Medium</td>
<td>GTM Related</td>
<td></td>
<td>Trigated</td>
<td>11/12/2006</td>
</tr>
<tr>
<td>1443</td>
<td>[Description]</td>
<td>Customer Experience</td>
<td>3 - Medium</td>
<td>GTM Related</td>
<td></td>
<td>Trigated</td>
<td>11/12/2006</td>
</tr>
<tr>
<td>1444</td>
<td>[Description]</td>
<td>Customer Experience</td>
<td>3 - Medium</td>
<td>GTM Related</td>
<td></td>
<td>Trigated</td>
<td>11/12/2006</td>
</tr>
<tr>
<td>1445</td>
<td>[Description]</td>
<td>Sales</td>
<td>2 - High</td>
<td>Not related to GTM</td>
<td></td>
<td>Trigated</td>
<td>11/12/2006</td>
</tr>
</tbody>
</table>
Spontaneous Quality Circles started forming

- Kanban board gives visibility into process issues – ragged flow, transaction costs of releases or transfers through stages in process, bottlenecks
- Daily standup provides forum for spontaneous association to attack process issues affecting productivity and lead time
- For example, 3 day freeze on test environment was a transaction cost on release that caused a bottleneck at “build” state. This was reduced to 24 hours after a 3 person quality circle formed to investigate the policies behind the freeze. Result was improved smooth flow resulting in higher throughput and shorter lead time
Empirically adjusted kanban limits several times
  E.g. test kanban too small, causing ragged flow

UAT state added
  Prompted by test who were experiencing slack time

Expanded kanban limit on Build Ready state, added Test Ready state
  Introduced to smooth flow post release due to environment outage transaction cost

Introduced kanban board, daily standup, colored post-it notes for different classes of service, notations on the post-its

Poor requirements causing downstream waste resulted in an upstream inspection to eliminate issues with poorly specified requests
In general, empirical observation of ragged flow or visibility of waste generates a quality circle resulting in a kaizen event.
Kanban innovates on typical agile/iterative development by introducing a late binding release commitment.

- Kanban system breaks constraint of typical agile/iterative 2-4 week cycle.
- Requests can take up to 100 days to process but releases still made every 14 days.
- Decision on content of release made 5 days prior to release.
- No estimation is done on individual items.
- Effort to estimate is turned back to productivity (analysis, coding, testing).
How Software Kanban Differs from Typical TPS Implementation

- No FIFO queuing
- Tasks prioritized by “cost of delay” or resource availability
- Cost of delay is heterogeneous
- Resources are often specialist, not generalist or cross-trained at prev/next stations
- Task durations have much wider variability – no tight 3 sigma limit, no takt time concept
Colors are used to designate qualities of service for work items.

**Issues are the exception** – attached to work items that are blocked for external reasons and call attention to problems preventing smooth flow.
Kanban has allowed us to observe known industrial engineering issues

- Overly large CRs caused ragged flow, blew out lead time
- Larger variation in CR size has required larger queues and buffer – extending lead time
- Ragged flow causes idle time – even on bottleneck stations (e.g. test)
- Non-constraints also exert ragged flow behavior due to non-instant availability (e.g. integration build)
- Big items are now broken up, breaks the Kanban limit but pull system means no new items enter WIP until overflow is pulled through. Result is smoother flow even with big items
Cumulative Flow

WIP growth due to additional resource allocation (good) and some sloppy management of kanban limits (bad)

Business encouraged to re-triage backlog
How many issues and blocked work items do we have?
Executive Dashboard

<table>
<thead>
<tr>
<th>Lead Time and Due Date Percentages</th>
<th>Lead Time (Average # of Days)</th>
<th>Due Date Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead Time, Engineering Ready to Release (CRs &amp; Bug Fixes)</td>
<td>30</td>
<td>32.5</td>
</tr>
<tr>
<td>Lead Time, Engineering Ready to Release (CRs Only)</td>
<td>30</td>
<td>32.6</td>
</tr>
<tr>
<td>Lead Time, Engineering Ready to Release (Bugs Only)</td>
<td>30</td>
<td>32.5</td>
</tr>
</tbody>
</table>

**Change Request Backlog:**

- **# Requested**
  - **Graph Legend:**
    - Request Count
    - Backlog All Count
    - Backlog GTM Count

**Throughput And Production Rate:**

- **# Deployed**
  - **Graph Legend:**
    - Running 3-Month Average
    - Deployed: Missed SLA
    - Deployed: On Time
Lead Times are lengthening again due to environment rebuild and business requested delay waiting for expedite request.
Due Date Performance Detail

Lead Time Distribution

Smoothed Lead Time Distribution

MARCH
Due Date Performance Detail

**Lead Time Distribution**

- Outliers
- Majority of CRs range 30 -> 55

**Smoothed Lead Time Distribution**

- Majority of CRs range 30 -> 55
Lead Time: Touch Time Ratio as an indicator of process waste and scope for improvement has been problematic to measure accurately.

More important is that thinking about lead time: touch time has focused line management attention on elimination of waste and reduction of variation.
Summary

- Culture Change
  - Trust, empowerment, objective data measurement, collaborative team working and focus on quality

- Policy Changes
  - Late-binding release scope, no estimating, late-binding prioritization

- Regular delivery cadence

- Continuous Improvement
  - Increased throughput, high quality, process continually evolving, kanban limits empirically adjusted
And finally, staff take a pride in their achievements.
Thank you!

David.anderson@corbis.com
http://www.agilemanagement.net/

Rick.Garber@corbis.com
About the presenters

David Anderson is Senior Director of Software Engineering with Corbis. He has 25 years experience in the software development business starting with computer games in the early 1980’s. As a pioneer in the agile software movement David has managed teams at Sprint PCS and Motorola delivering superior productivity and quality. More recently at Microsoft he developed the MSF for CMMI Process Improvement methodology.


David’s team at Corbis are currently focused on introducing more Lean ideas including use of kanban, oobeya, and visual control techniques to demonstrate high levels of productivity, improved lead times and quality while using new and traditional software engineering techniques such as software factories, modeling, architecture to enable postponement and the use of real option theory in managerial decision making.

Rick Garber is Manager of IT Process Engineering with Corbis in Seattle, WA where he leads process improvement initiatives for Corbis’ software engineering, IT services, business intelligence and global infrastructure teams. Rick has played a key role in the definition and implementation of a kanban system for sustainment engineering at Corbis.

Previously, Rick was an IT consultant/project manager with Equarius (now EMC Microsoft Solutions) in Bellevue, WA. With Equarius and subsequently with Corbis, Rick was part of a team that designed and developed Corbis’ core media management system.

Rick holds a bachelors degree in Industrial Engineering and MBA from Oregon State University, and a Certificate of Advanced Studies in Database Management from the University of Denver. He lives in Kirkland, WA.