A Practical Approach to Large-Scale Agile Development

An HP case study: The transformation of LaserJet FutureSmart Firmware

Mike Young, Hewlett-Packard
Agile Leadership Conference – Houston
Overview

Â Embarked on a large-scale agile journey the past 3 years
Â Found that most published agile material (especially with real-world experience) is from small-scale software systems
Â HP LaserJet FutureSmart Firmware: not just device drivers. An embedded software system, with 400+ engineers around the world, with complex features and large code base (10M+ LOC)

Â High-end LaserJet printers and MFP’s
Â Enterprise Security
Â Digital Sending
Â HP Open Extensibility Platform
Â Embedded Web Server
Â ePrint
Â Corporate Manageability (Web JetAdmin)
HP FutureSmart Firmware Geography

- Vancouver
- Boise
- Fort Collins
- Roseville
- Bangalore
- Porto Alegre
Decided to publish our experience

- Publishing our experience in a book at Addison-Wesley (part of their “Agile Software Development Series”)
- Can pre-order now online at Amazon, B&N, etc. (available in May)
- Definitely don’t have all the answers. But lots of learnings on what worked and didn’t work for us
State of the development process - 2008

Costs out of control:
- Development costs growing 2.5X from 2004-2008 and the business was still constrained
- Up to 10 different branches (driven by each product release window) in MFP CPE driving millions/year in CPE investments

Couldn’t add enough resources:
- 80-90% of resources just porting existing FW to new products and qualifying
- Unable to add new products to the plans due to lack of FW resources
- 20% of resources developing plans that quickly became obsolete

Lengthy build integration & testing cycles:
- 6 weeks + to get through a complete testing cycle (mainly manual)
- Build integration taking 15-20% of resources a week to get fixes to main
- Manual testing a key driver and constraint for adding products

Products lagging the competition:
- Ongoing customer issues with consistency and lack of features
- Marketing had essentially given up asking for FW innovations
Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck  James Grenning  Robert C. Martin
Mike Beedle  Jim Highsmith  Steve Mellor
Arie van Bennekum  Andrew Hunt  Ken Schwaber
Alistair Cockburn  Ron Jeffries  Jeff Sutherland
Ward Cunningham  Jon Kern  Dave Thomas
Martin Fowler  Brian Marick

© 2001, the above authors
this declaration may be freely copied in any form, but only in its entirety through this notice.
Firmware Development Transformation

Agile Development with Mini-Milestones (Sprints)

### MM30 Objectives

<table>
<thead>
<tr>
<th>Rank</th>
<th>Theme</th>
<th>Exit Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Quality threshold</td>
<td>PM open &lt; 1 week-domain coverage &gt; 100% pass, UI 24hr response, Tests for CAT rescopes</td>
</tr>
<tr>
<td>1</td>
<td>Quarterly Bit Release</td>
<td>A) 100% pass, B) Duration error rate per $10k: 0.3 (sim), 0.35 (emul), 0.4 (product)</td>
</tr>
<tr>
<td>2</td>
<td>CE stability and test coverage (P2P)</td>
<td>A) 100% pass, B) Coverage for high-value PTO1-PTO3 reqts, C) L4 CAT, D) CAT in place with at least 95% CAT equivalence, E) L4 test coverage for all PTO1-PTO3 reqts, F) Duplicate L4 tests to new products (100% identical test/others)</td>
</tr>
<tr>
<td>3</td>
<td>PTO4 dependencies and key features</td>
<td>A) Calibration dependencies, B) Print for an hour at speed to finisher with stapling, C) Copy for an hour at speed, D) Initial test pass (enhanced for coverage), E) Approved to push out to MM31, F) PIN test suite execution (simplified, needs MM support), G) PLM CFT support for 4 line display, H) Send to developer, 3rd party SW available, I) Send to developer, J) High-level analysis of performance on ARM lowered priority.</td>
</tr>
</tbody>
</table>
| 4    | Build for next gen products | A) Build single ARM system feasibility proof, B) Build a new product |}

### Continuous Integration and Test System

- **Stage 1:**
  - **L1** (10-14/day)
- **Stage 2:**
  - **L2** (12x/day)
  - **L3** (6x/day)
  - **L4** (1x/day)

### Organizational change management

1. **Af9 Products**
   - Oct '09: New feature development, Bug fixing
   - Mar '10: New feature development, Bug fixing
2. **Soi 0 Products**
   - Oct '09: New feature development, Bug fixing
   - Oct '10: New feature development, Bug fixing

### Architected for Product Variability

- Robust, scalable architecture
- Adaptable test cases
- Robust, scalable architecture
- Adaptable test cases
Making large-scale agile happen

∆ Take advantage of the large-scale positives
  - scale allows tools and automation that streamline dramatically
    GIT / Integration Queueing (IQ), Virtual Machine Provisioning System (VMPS),
    HP QualityCenter (reqts, test mgmt/reporting, change requests), HP Halo

∆ Build a culture
  ∆ This was our 3rd try at making large-scale agile successful
  1st) Tops-down (nice metrics, little buy-in)
  2nd) Bottoms-up (good excitement, limited reach)
  3rd) Active sponsorship with innovation/improvements
      driven by all

∆ Key drivers in the organization
  ∆ Management sponsor (our director of engineering)
  ∆ ScrumMaster
  ∆ Lead architect

∆ Eliminate status meetings wherever possible
  ∆ Automated metrics
  ∆ Clear roles/responsibilities
  ∆ Clear priorities
HP FutureSmart Firmware Top 6 Agile Principles

1. Reduce overhead and waste

2. Don’t overfill your plate

3. Cater to the bottleneck

4. Integrate early and often

5. Planning rhythm

6. Doers define agile/lean practices

- JIT process changes
- “No” now means more overall “yes”
- Global short-term resource moves
- How to not sacrifice quality?
- Ranked objectives, regular cadence
- What is painful? What would you do?
Aligning agile with Business Needs

What are your development cost and cycle time drivers?

What is your value proposition?

- Reduce cost? Maximize # of products? Clear customer differentiation?

→ Focus on minimizing biggest cost drivers that are not key to your value proposition
Development cost driver improvements

2008

- Code integration: 10%
- Detailed Planning: 20%
- Porting code: 25%
- Current product support: 25%
- Manual Testing: 15%
- Capacity for Innovation: ~5%

2011

- Continuous Integration: 2%
- Agile Planning: 5%
- One main branch: 15%
- One branch CPE: 10%
- Most testing automated: 5%
- Capacity for innovation: ~40%
Cycletime driver improvements

2008

- Build bosses
  - 1 week
- Number of builds
  - 1-2
- Feedback on Main
  - 1 commit/day
- Full Manual regression testing
  - 6 weeks

2011

- Continuous Integration
  - 3 hours
- Continuous Integration
  - 10-15/day
- Autorevert
  - ~100 commits/day
- Auto regression testing
  - 24 hours
Aligning architecture with Business & Agile

- Architecture can make or break any business, and any agile effort.
- If changing architecture, make sure it’s tied to the critical success needs of your business

Key enablers in FutureSmart Firmware Architecture
- Code identifies hardware it is running on and then configures itself
- MFP design center (job queue, performance, contention)
- “Feature enable/disable” concept (support new business model: customer buying a device and subscribing to future upgrades to keep all HP devices consistent and new)

- Most large architecture changes are done “big-bang” style.
  Then a big quality hole to dig out of. This is when we first started 4-week Sprints (prove out each end-to-end “slice”)
    - Demos at Sprint Checkpoints are powerful (excitement, focus, reward)
Agile Architecture Rollout

Thin slice approach to bringing up a new architecture

Agile architecture process

1. Define a Thin Slice
2. Informal Review
3. Prove with Prototype
4. Review/Adjust

Thin slice approach to new architecture
Iterative approach to Agile Management

Having real-time metrics is essential for the speed of agile. But **don’t manage by metrics.** Use the metrics to **understand where to have conversations** about what is not getting done.
# Sprint Objectives

## MM30 Objectives

<table>
<thead>
<tr>
<th>Rank</th>
<th>Theme</th>
<th>Exit Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Quality threshold</td>
<td>- P1 open &lt; 1wk - CAT 100% pass - L2 24hr response - Tests for CAT escapes</td>
</tr>
<tr>
<td>1</td>
<td>Quarterly Bit Release</td>
<td>A) Final P1 change requests fixed 2 remaining.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B) Duration error rate per 10K: 0.3 (sim), 0.35 (emul), 0.4 (product)</td>
</tr>
<tr>
<td>2</td>
<td>CEstability and test coverage (PTO3)</td>
<td>A) L2/ L3/ L4 CAT 100% passing w/ proper coverage (3 superbundles/ wk)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B) All L2 pillars 98% pass w/ coverage for high-value PTO1-PTO3 reqts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C) L4emu test pillars in place LLAW, copy/ PDL, PrintDevice</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D) L3 CAT in place with at least L4 CAT equivalence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E) L4 test coverage for all PTO1-PTO3 reqts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F) Duplicate L4 tests to new products 100% exec (no DS okay)</td>
</tr>
<tr>
<td>3</td>
<td>PTO4 dependencies and key features</td>
<td>A) Calibration dependencies</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B) Print for an hour at speed to finisher with stapling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C) Copy for an hour at speed 35ppm (40ppm is at speed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D) Enter/ exit powersave Approved to push out to MM31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E) Falcon test suite execution Emulator still needs RM support</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F) Automated RM no bash prompt Approved to push to MM31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G) RUI/ CTF support for 4-line display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H) Send to Folder, 3rd-party SW avail for Send to Email</td>
</tr>
<tr>
<td>4</td>
<td>Build for next-gen products</td>
<td>A) Build single ARM system Feasibility proven. 2 DLUs to re-compile.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B) High-level analysis of performance on ARM lowered priority.</td>
</tr>
<tr>
<td>5</td>
<td>Fleet Integration plan</td>
<td>Align on content for slivers of end-to-end agile test. Overall plan in place.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Need sliver details or will just deliver same as to PTOs.</td>
</tr>
</tbody>
</table>

- Ranked
- Published broadly
- Used by every engineer and manager for daily tradeoffs
- Okay to stretch, but shouldn’t have anything unachievable
- Use for everything (arch., features, products, qual, process)
Continuous Integration – where we started


Metrics page we called the “red/green show” that showed hour-by-hour when the integration system was up and ready to accept new commits.

Got us to small/frequent integrations, but anyone could bring the system down anytime. We were red much of Boise daytime every day. Many people committed changes late at night and weekends.

Also had issues with repeatable test results during integration. Chat room when build failures – posted an “L1 craps” visual.
Continuous Integration and Test System

Stage 1:
- \( L_1 \) (10-14x/day)
- Small groupings of passing individual commits
- Sam, Julie, Hank, Becky, Doug
- Fix

Stage 2:
- Auto-Revert
- Merge conflict

Stage 3:
- \( L_2 \) (12x/day)
- Manual Intervention

Stage 4:
- \( L_3 \) (6x/day)

Stage 5:
- \( L_4 \) (1x/day)
## Testing Levels

**Integration**

- **L0**
  - Pre-commit testing. Done by the developer on his/her own development machine before committing to the main trunk (at the bare minimum, includes running "qbar" locally).

- **L1**
  - Commit testing. Happens automatically at the time a developer commits new code to the main trunk of the SCM (broken up into Stage 1 and Stage 2 for easier auto-revert). This is now our "Integration Queuing" (IQ) system.

**Stability**

- **L2**
  - Quick-turn broad-based testing by pillar (runs every 2 hours, across multiple test machines in parallel). Intended as a quick feedback loop to find broad-based failures from new commits in as narrow of a commit window as feasible.

- **L3**
  - Same purpose as L2, but on real hardware so can know that the full build, firmware download, boot, and general capability are working end-to-end. Runs every 4 hours.

**Regression**

- **L4**
  - Full regression test suite of all automated tests. Kicks off at midnight daily and provides complete view of the quality of the system. If passing rate drops substantially, should have seen a dip in L2 testing (otherwise, update the L2 list).
FutureSmart FW Large Scale Agile Development Engine

- 300+ developers effectively working on 10M line code base
- Processing 75,000-100,000 lines of code turmoil with 100-150 separate developer commitments
- 10-15 complete builds a day
- Stage 1 and 2 testing on every commit to keep big defects out
- A full set of Automated Regression (15,000 hours/day with ~90% pass rate) to find defects that make it through stage 1 & 2 filters
- Auto qualification versus costly manual tests

Breakthrough capacity for integrating code and deploying it across the fleet
Taming the Planning Beast

*Where we were in 2008:*
- Business required final feature list 12 months in advance that we could never deliver due to inevitable plan changes.
- Our planning resources were the same as the development resources, so they would stop working on features to do detailed work-breakdowns and scheduling.
- Always in a “locked in” mode; had to say no to any late-breaking requests (or throw “plans” out the window and start over).

*Guiding principle now:*
Every hour we spend planning a feature is an hour we don’t spend delivering it (the real goal).
Taming the Planning Beast: Solution

Tools for very **light-touch predicting** of how far we’ll get on a “1 to N feature list” (without delving into the details or engaging developers).

**Prioritize everything.** Allow priorities to self-drive the organization. As needed, anyone drops what they’re doing and helps the person with the higher ranked user story / feature.

**Full-time System Engineers** who drive **just-in-time feature definition** using user stories with feature requestors to offload the developers and give a clear path forward when ready to engage.

**Commit by delivering, not by estimating!**
FutureSmart Firmware User Stories per Sprint

Input Queue of user stories (not in WIP yet; being evaluated)
Now on quarterly releases (2 feature sprints plus 1 final qual sprint),
so Input Queue is ~2 releases of backlog (80-100 User Stories per release)

1st year throughput (25 user stories/sprint)
2nd year throughput (big drive to complete architecture; not sustainable)

Latest throughput (need more data to know new steady-state; probably 40-50/ sprint)
Getting Mgmt/Mktg Buy-in to Agile Planning

“FW will still commit to basic new product support one year ahead”
- Means prioritizing “product turn-on and delivery/qualification” ahead of new features
- Separate out “new feature requests” from “make my product work”

“You will get 20% more features this way”
- A leap of faith since seeing is believing, but easy to explain the 20% of resources previously used to estimate (which often meant complete wasted effort for things that we ended up not doing)

“You get to decide what we work on first”
- Establish a “1-N feature request list” and the combined marketing teams decide the order (actually helped them feel much more in control than before)
- Single point of contact in marketing to prioritize new features (else the squeaky wheel gets the grease)
- Required a governance model where business leaders reviewed/approved the 1-N request list.

“We’ll actually listen to your last-minute requests”
- Competitor comes up with a cool new feature just before product intro? Large sales opportunity requires a feature to close the deal?
  → Previously, waited 12-18 months because we were already “committed” to everything else.
  → Now, just put it at the top of the list, ahead of all the other “input queue” features (recent example: “HP ePrint”. Able to deliver with quick turnaround)
State of the art FW development model

2008

- Costs out of control
- Couldn’t add resources fast enough
- Lengthy build, integration and testing cycles
- Products lagging the competition

2011

- ~70% reduction in FW development cost per program
- 50% reduction in FW headcount
- Cont. integration, Daily automated regression
- Vintage chart unleashed and capacity for innovation
Recent Learnings

Â Don’t over-rely on automation
- Awesome for integration, stability, and regression
- Doesn’t take the place of exploratory testing

Â Quickly train new management on agile principles
- New engineers “get it” quickly (peer pressure, plus using tools)
- Managers (at all levels) naturally want to maximize confidence and minimize risk (“give me high-confidence estimates and commits”)

Â Much of the world is still waterfall – must figure out how to help get the whole enterprise to agile.
- Solution Test, User Manuals, Support/Training, Manufacturing
- We have started working on “enterprise deployment” improvements based on ideas from the following:
  
  
Started with product driven releases
Started with product driven releases
Product driven releases
(Delivery/Qualification)

Component A
Component B
Component C

Dev.  Qual.  Intro
Dev.  Qual.  Intro
Dev.  Qual.  Intro
Dev.  Qual.  Intro
Trending towards Enterprise Agility with Fleet Releases & System Qual

- Component A
- Component B
- Component C

Fleet Req

Agile Dev with Quarterly System Releases

Intro

Intro

Intro
Fleet Consistency & Powerful Workflow Solutions with

HP FutureSmart Firmware

LaserJet Enterprise Solutions
HP FutureSmart
A New Breed of Device Firmware

Shared Client on the Network
- Dramatic increase of functionality
- Integration into Solutions Eco-System
- Security

Evolution of Firmware
- Investment protection through fleet consistency
- Easy to manage
- Powerful workflow engine

Introduced Fall 2010
Each launch adds a new branch of code
Adding a new feature requires implementation and testing to each branch

Single, main code branch: New features apply to existing products, therefore becoming forwards compatible
Still requires some regression testing but development work & code check-in is done once only
Reliable Platform, Efficient Development

Single Code-Base
Less errors, fixes apply to all products

AGILE Development
Faster problem identification & resolution

Robust & Reliable Platform

Better Development Tools
Industry standard development & debug environment: Microsoft Visual Studio

Automated Test Tools
Full regression test suite: from 1 run per 6 weeks to now every night
Virtual environment for large scale reliability testing (500+ number of virtual & real devices)
Powerful Workflow Engine
Creating new customer value

Image Preview, Job Editing
- View: scale
- Edit: delete, insert, rotate, reorder

Quick Sets
- Create custom buttons on the home screen to support frequent workflows
- Setup in embedded web server & Web Jetadmin
Powerful Workflow Engine
Creating new customer value

Encrypted Print via UPD
- Password-based
- Complete print path: driver/queue/printer

Security Settings
- Device access & feature-level control
- By user or group of users
- Through embedded web server & Web Jetadmin

UPD = Universal Print Driver
HP Open Extensibility Platform (OXP)
Enable solution applications in your imaging and printing environment

**Solution Portfolio**
- PRINTABLE
- HYLAND SOFTWARE
- omtool
- mimeo
- NUANCE
- KOFAX
- CAPPELLA TECHNOLOGIES
- Jetmobile
- SafeCom
- Pharos Systems
- equitrac
- TROY

**Single Set of API’s**
- ePrint, Exstream, UPD
- Workflow
- WebJetadmin
- Management
- Device Extensions

- Links ecosystem of HP and partner solutions & components talk & share data
- One tool to install, configure and manage HP devices AND solutions
- Platform independent -installed base devices back to 2005
- Solution flexibility through server deployment
- Consistent user interface across the fleet
Agility transformation leading to market place branding and differentiation

http://www.youtube.com/watch?v=JVtV16T9GSo
Taking the first step towards large-scale agile

Are manual tests a big part of your cost driver and cycle-time? Start with test automation.

Are you bringing up a new architecture? Focus on thin slices and demo of real code in monthly checkpoints.

Is monthly/weekly predictability your biggest challenge? Take a snapshot of what everyone thinks they can get done as the sprint starts, and compare it to actuals (everyone will be calibrated in 2-3 sprints).

Is getting your code to meet customer expectations your biggest issue? Start with user stories and test driven design.

Are you having a hard time finishing anything? Control your Work In Process (WIP). Don’t let a sprint get interrupted with new requests.

Do you have a consistent bottleneck? Start catering to it.

Are you doing big-bang integration? Get an “auto-revert” tool in place with even a simple test suite as a barrier to keep bad code out. Reduce your build time.

Are you spending all your time estimating the future to convince the business you can deliver? Invest in System Engineers and simple models for long-range prediction.