Agile Development in a Large Company

IEEE NJ Coast Section Technical Talk
June 10, 2009

John Palframan
palframan@avaya.com

Research for this topic on Avaya projects was done in collaboration with Randy Hackbarth – Avaya Labs, and David Weiss – Iowa State University
What Is Iterative and Agile Programming
Before Agile – The Waterfall Model

Where Waterfall Has Problems

- Risk mitigation is postponed
  - Design up front, but then may encounter implementation difficulties
  - Testing not done until the end, issues may require a redesign

- Scope Management
  - Requirement always change, especially with a long development cycle

- Building the right thing
  - Clients usually don’t know what they want until they see a prototype
  - Requirements are usually oriented mainly towards functionality
  - Studies show a large proportion of early requirements are never used

- Because of rework due to the above, cost and time estimates are difficult

- Documentation is difficult to keep up to date

“From Waterfall to Evolutionary Development”, Trond Johansen, Tom Gilb
“Understanding The Pros And Cons Of The Waterfall Model Of Software Development”, Melonfire
“Managing The Development Of Large Software Systems”, Dr. Winston W. Rovce
Waterfall Studies

- **UK study of 1,027 projects:**
  - 87% failure rate
  - 82% cited waterfall-style scope management as the "single largest contributing factor for failure"

- **1995 study of $37M worth of DoD projects**
  - 46% of the systems so egregiously did not meet the real needs (although they met the specifications) that they were never successfully used
  - 20% required extensive rework to be usable

- **Study of 6,700 projects:**
  - Four out of the five key factors contributing to project failure were associated with and aggravated by the waterfall model
  - Inability to deal with changing requirements
  - Problems with late integration

- **Study of over 400 waterfall projects:**
  - Only 10% of the developed code was actually deployed
  - Only 2% was actually used

"Agile and Iterative Development: A Manager's Guide", Craig Larman
Iterative Development

http://en.wikipedia.org/wiki/Iterative_development
Agile Development

• A group of software development methodologies based on iterative development
• Requirements and solutions evolve through collaboration between self-organizing cross-functional teams

http://en.wikipedia.org/wiki/Agile_software_development
Traditional vs Agile

http://www.versionone.com/Resources/AgileBenefits.asp
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, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick

http://agilemanifesto.org/
Principles behind the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.

3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. **Working software is the primary measure of progress**.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. **Simplicity**—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

http://agilemanifesto.org/
% of Companies Using Agile Using Selected Agile Methodologies

http://www.versionone.com/Resources/AgileMethodologies.asp
Agile – Commonalities

- Frequent Deliveries
- Base kept at a high quality level throughout development
- Team is empowered, self organized, accountable
- Large Amount of Communications/Interactions Among Members
- Stakeholder Involvement
- Transparency
- Disciplined project management process
XP – Kent Beck

- Deliver high-quality software (Simple Designs, Pair Programming, Test-Driven Development, Refactoring, Continuous Integration, Collective Code Ownership, Coding Standards)
- Deliver quickly and continuously (every 1-3 weeks)
- High customer involvement (acceptance tests, on-site)
- Rapid Feedback Loops
- Continuous Planning and Testing

http://www.extremeprogramming.org/
Other Methodologies

**FDD**
- Domain Object Modeling
- Developing by Feature
- Component/Class Ownership
- Design/Build by Feature Teams
- Inspections
- Config Management
- Regular Builds
- Visibility of progress and result

**DDSM**
- Business Needs/Value
- Risk Analysis & Mitigation
- Planning
- Active User Involvement
- Empowered Teams
- Frequent Deliveries
- Testing Throughput Development
- Stakeholder Collaboration

**Crystal**
- Tailor to characteristics
  (team size, criticality, priorities..)
- Teamwork
- Communications
- Simplicity
- Ongoing process improvement

**Lean**
- Eliminating Waste
- Amplifying Learning
- Deciding as Late as Possible
- Delivering as Fast as Possible
- Empowering the Team
- Building Integrity In
- Seeing the Whole
Product Owner (re)prioritizes backlog each sprint

Sprint Planning Meeting

Transparency in planning and module development Base breakage minimized

Sprint Retrospective

Scrum: 15 minute daily meeting. Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?

Scrummaster: removes obstacles

Collaborative Project Team

“Done”

New functionality is demonstrated at end of sprint

http://www.controlchaos.com

Scrum in 10 Minutes: http://www.youtube.com/watch?v=Q5k7a9YEoUI&fmt=22
SCRUM - Tools

- Product Backlog
  - Prioritized list of features, infrastructure items, wish-list items
  - Rough estimates of business value and development effort

- Sprint Backlog
  - User Stories broken into tasks & staff hours
  - Tasks prioritized and team members sign-up to develop

- Burn Down for tasks completed (and Burn-Up for projects)
  - Public, up-to-date display of progress on sprint backlog

- Project Management Tools such as Rally or Scrumworks

Challenges of Deploying Agile In a Large Enterprise

- Large distributed projects
- Code base could be 10s of millions of lines of code
  - Code ownership
  - Frequent builds
- Interworking with many other products (being developed simultaneously) with consistent policies, look and feel, “common” resources….
- Mission critical reliability
- Millions of lines of legacy code, perhaps with no automated tests
- May only have indirect contact with end customers
What Is Avaya
Avaya – Enterprise Communications

Converged Communications
Unified Messaging
Contact Centers
Communications Enabled
Business Processes

http://www.avaya.com
Converged Communications

VoIP (Voice over IP i.e. Internet) and Traditional Switches, Gateways, Phones, Control Interfaces (API) for

Targeting industry segments with special features: e.g.

- Financial:
  - Highly reliable 99.999% = 5 minutes downtime a year

- Hotel
  - Access to hotel services
  - Wakeup calls

- SIP, IP, Digital & Analog
- Cross Network
- Management
- Scale
- Security
- Simplification
Unified Communications

Unified Access
Mobility
Telephony
Messaging
Conferencing
Video
Presence
Multi Vendor

Unified Communication

Modular Messaging Web Client

Mailbox Usage: 2%

Message Folders:
- Inbox
- New MM Message
- New Email Message
- Directory
- Options
- Help
- Logoff

21015 02/21/2007 05:41 AM Voice Mail Message (9 seconds)
Avaya Customers

- Global Enterprises
- Branches of Those Enterprises
Avaya’s Competitors (2007)

- **IP Telephony**: AVAYA, Cisco, Nortel, Alcatel
- **Unified Messaging**: AVAYA, Nortel, NEC
- **Audio Conferencing Web Conferencing**: AVAYA, Cisco, Compunetix, IBM, Adobe

Market Share

<table>
<thead>
<tr>
<th>0%</th>
<th>20%</th>
<th>40%</th>
<th>60%</th>
<th>80%</th>
<th>100%</th>
</tr>
</thead>
</table>

© 2009 Avaya Inc. All rights reserved.
Avaya R&D Organization Characteristics

- **Culture**
  - Many different development cultures due to acquisitions
  - Siloed – Business Units have P&L responsibility
  - Individual projects are run independently
- **Very Schedule Driven, though oscillates between**
  - Cost (drive to outsource)
  - Quality (drive for improved customer satisfaction)
  - Schedule (Making a name for AVAYA after spin, coordinating interdependent products to be released together)
- **Experience**
  - Combination of very experienced staff and new hires
  - Takes 1-2 years to become productive on a complex product
- **Geography**
  - Distributed Development across 16 time zones

Avaya Research and Development

- 2300 staff + additional contractors/outsource organizations
- 30% from Lucent Bell Labs, 40% new hires since 2001, 30% acquisitions
Avaya – R&D Software Characteristics

- Real Time
- Embedded
- Critical Reliability
- A lot of legacy code through Bell Labs and Acquisitions
- Interdependent (e.g. Switch, Presence Server, SIP server, Interface server etc all work together).
- Networked
- Highly Configurable
- Large (some bases take days to compile and smoke test)
- Complex
- Integrated with 3rd party and open source code
- Many languages (mostly C, C++, JAVA, PHP, C#), hardware platforms, and operating systems (Linux, Windows, VxWorks..)
Avaya Code Base Growth - Selected Projects
How Products are Developed

- Prototyping
- Requirements
- Development Planning
- Development Infrastructure
- Software Architecture

- Current Engineering
- Field Support

- Design
- Coding & Unit Test
- Integration & Functional Test
- System Test
- Alpha and Beta Programs
- TRADITIONAL & AGILE MODELS USED
How Does Avaya Approach Agile Development
Primary Goals In Avaya for Agile Development
(as reported by projects)

1. Make it easier to obtain and incorporate user feedback in order to improve customer satisfaction

2. Make development progress visible by doing development in small steps, thereby giving the development team clear, short-term projects.

3. Make it easier to develop correct, high-quality software by using a series of small steps

4. Build a productive and knowledgeable team.

5. Make Progress in the context of an incomplete view

6. Improve predictability
Percentage of Avaya Iterative Projects with Stated Goal

Goal

- User feedback and customer satisfaction
- Development progress visible
- High-quality software by using a series of small steps
- Build a productive and knowledgeable team
- Make progress in the context of an incomplete view of the product
- Improve Predictability

Percentage

© 2009 Avaya Inc. All rights reserved.
Cumulative Number of Avaya Projects With Agile Practices
Areas We Track in R&D for “Architecture Guided Iterative Development” – Traditional Practices

• **Architecture Guided**
  - Decompose work into minimally interdependent modules
  - Develop interface specifications so teams (and projects) can work independently
  - Ensure project characteristics, (e.g. performance, reliability, capacity) are reachable
  - Use common code, algorithms, databases across and other technologies products
  - Architecture Scorecards to ensure consistency
  - Architecture reviews across divisions
  - Prototyping allows for early feedback and technology risk assessment

• **Baselined High-Level Requirements**
  - Determine how products interoperate
  - Enough detail for an initial architecture

• **Plan and Track the Entire Release, Not Just Iterations**
  - Teams and functional areas must converge
  - Plan for Alpha and Beta tests in complex customer environments

---

“5 Levels of Agile Planning: From Enterprise Product Vision to Team Stand-up”  Hubert Smits

© 2009 Avaya Inc. All rights reserved.
Areas We Track in R&D for “Architecture Guided Iterative Development” – Traditional Practices

- **Common Repositories & Tools**
  - All teams have access to code, defect reports, test results, code coverage
  - Common procedures and checks for check-in code

- **Reviews and Inspections**
  - Cross product architecture reviews
  - Meetingless design reviews across teams
  - Inspect code or test plans, especially reviews of inexperienced or outsourced staff by experienced staff

- **Automated Build Management with Sanity Tests.**
  - Most projects build daily, some do continuous builds, allowing immediate feedback to developers to fix base problems
  - Developers deliver tests along with code to build up sanity tests
  - Legacy code may not have automated tests
  - Automated propagation of the tested compiled base to all sites

- **Manage 3rd Party Deliverables** (acceptance plan, quality assessment, SCM)
  - Commercial and open source
  - Early warning if something is amiss
Areas We Track for “Architecture Guided Iterative Development” – Avaya Agile Practices – SCRUM Based

• **Collaborative Project Team**
  - In Avaya this is generally limited to R&D (e.g. services, documentation, product introduction, etc not included)
  - SCRUM of SCRUMS used to scale to multiple teams and multiple locations
  - Cross location team could employ open phone!

• **Empowered Product Owner**
  - Usually Product Manager, usually not co-located with R&D
  - Established an onsite team lead when there are offshore members

• **Empowered Team Lead**
  - ScrumMaster

• **Time-Boxed Iterations with Demo**
  - Most projects have 4 week iterations
  - Value of Demo is "huge". Rotating responsibility for end-end demo hardened the demo and allowed centralized dispersal of feedback
  - Demos use simulators and web conferencing for distributed environment
  - Demo is good at getting visibility into remote or offshore work

* - SRUM Practice
Areas We Track for “Architecture Guided Iterative Development” – Avaya Agile Practices – SCRUM Based

• **Ease of Communication/Transparent Status***
  - Widespread use of Rally, Wikis, Sharepoints for metrics, code, methodology, designs, interface specs, architecture etc.
  - Other functional areas sometimes involved in development sprints

• **Prioritized Feature List, with Cut-Offs***
  - Mixed success; some projects unable to prioritize

• **Code Refactoring***
  - Mixed success – intense schedule pressure

• **Plan &Track Each Iteration with Business Goal, Fixed Resources and Content***
  - Each team understands how they mesh with the other teams

• **Automated Test Framework***
  - Consistency of testing across teams (legacy code tests only partially automated)
  - Some Test Driven Design
  - Various SV strategies (work with development, lag an iteration, just take it at the end)

• **Iteration Retrospectives***
  - Teams find they can implement 2-3 process changes per iteration
Areas We Track for “Architecture Guided Iterative Development” – Avaya Agile Practices – SCRUM Based

• **Document Just Enough**
  - Combination of traditional documents and specs embedded in the code (e.g. JAVADOCs, D’Oxygen) – keeps documentation accurate for those at other sites.

• **Customer Feedback Throughout Lifecycle***
  - Product Management usually acts as proxy, some support interaction, some customer focus groups
  - Conference calls allow whole team to participate with forums such as Customer Advisory Panels
  - Rotating developers through customer support, regardless of location, improves understanding of customers

• **Brief Daily Stand-up Meeting***
  - Manage across time zones (time shift, rotate times, fewer, conference bridge)
  - Issues are identified early which allows quicker mitigation.
  - The meetings often identify concrete areas where extended team members and other domain experts can assist.
  - The meetings keep the team focused
  - The development staff must account for themselves.
  - Knowledge is dispersed throughout the team.
  - Helped SV team when they were not co-located to understand product earlier
Bottom Line

- Use of Agile Methods Continues to Grow
- Increased Quality and Productivity Not Yet Quantitatively Verified
- No Change In Time To Market (releases coordinated with non-Agile products, and timed to twice a year)
- Hampered By:
  - Limited Involvement Outside of Development
  - Legacy Code
  - Lack of Legacy Automated Tests
  - Interdependence with non-Agile projects
- Can Scale to Large Distributed Teams
thank you

John Palframan
palframan@avaya.com
Some References