Obstacles to Enterprise Agility

Michael James, CollabNet Certified Scrum Trainer

March 15, 2010

I often work with divisions of Fortune 500 companies that are struggling to become agile, starting with Scrum. While each organization is in a distinct business sector using different technology and management cultures, each one shares a common pathology, a kind of “giantism.” This article lists common obstacles to agility in large organizations and explores the possibility that the symptoms of giantism are entirely avoidable.

At first glance, an organization’s challenges will appear to be “too much to do” or “not enough resources” or “changing business climate.” Upon closer inspection, the root causes will turn out to be bad habits, unexamined reflexes and misconceptions.

A division of a well-known company cited as a 1997 success story by a famous Scrum pioneer came to Danube Technologies, Inc. for help in 2009 because market forces revealed it was less agile than its competitors. The Scrum initiative that started in 1997 apparently couldn’t withstand a decade of obstacles to large-scale agility. Sadly, most attempts to adopt Scrum in large organizations do not result in durable, ongoing transformation. Obstacles to Scrum adoption are usually obstacles to business success in general, and established organizations are usually reluctant to let go of them.

Obstacle #1: Naive Resource Management

The PMBOK Guide observes “often the budget needs to be increased to add additional resources to complete the same amount of work in less time.” More specific to software, Fred Brooks (in The Mythical Man-Month) makes an apparently contradictory claim: “Adding manpower to a late software project makes it later.” To resolve this paradox, let’s examine the definition of “resource.”

When the work is new product development, the relevant resources are intangible: task absorption, learning, interpersonal communication and innovation. Scrum teams attempt to maximize these by creating states of individual and group “flow.” According to psychologist Mihaly Csikszentmihalyi, “[In flow] your whole being is involved, and you are using your skills to the utmost.”

Scrum development team members collaborate intensively to build products according to goals they repeatedly negotiate with the product owner, who is responsible for making the team’s business decisions. Results are demonstrated at the end of every fixed length sprint (e.g. every two weeks). During sprint execution, team members develop intrinsic interest in shared goals and learn to manage each other to achieve them. Even with ideal circumstances (including a team room) it takes a team a few sprints of stable membership to hit its stride, and a year or so to reach its potential. A popular description of this organic process is Bruce Tuckman’s “forming, storming, norming, performing” model.

It is naive to think of human beings as resources. Adding people to a team will not reliably increase the intangible resources--and may detract from them. After a year of doing Scrum, one of my clients reported “Once a team is formed, we would rather lose a team member than add one!” In another case, when the Scrum team itself made the hiring decision, adding a new member went well. Even when giving the team hiring autonomy, it’s inadvisable to grow it much larger than seven people.
In some circumstances, adding teams may result in more progress, if we’re mindful of the intangible resources.

**Obstacle #2: Teams Organized by Functional Specialization**

A Scrum Team is a cross-functional group that attempts to build a properly tested product increment every sprint, gradually adding feature scope. The *most popular book about Scrum* uses the phrase “potentially shippable product increment” 18 times. Despite this, I meet people who claim to be “doing Scrum” while executing “analysis sprints” or “design sprints” at the beginning, deferring integration and testing to the end, and holding different teams responsible for each phase! These waterfall habits hide risk until it’s too late to respond.

In Scrum, every Sprint requires a mix of development activities. We gain agility through continuous requirements analysis, continuous design, continuous integration and continuous testing, all feeding into each other.

**Obstacle #3: Teams Organized by Architectural Components**

Single-component teams reduce the business’s ability to re-prioritize product capabilities, increase the coordination bottlenecks through managers and product owners, and introduce integration risks.

The component team approach would be efficient if new product development were as predictable as manufacturing. In practice, priorities change and estimates prove incorrect, thus it’s difficult to get proper focus from the right people at the right time to build capabilities affecting multiple components.

The opposite of the component team is the feature team. A feature team spans both components and disciplines, and is thus capable of building business value features in thin, fully-tested vertical slices. The feature team approach replaces last century’s efficiency thinking with a concept of team autonomy, ownership and responsibility. The product backlogs of feature teams are driven by business value, not technology dependency. If you are still living with Gantt charts, you probably don’t have feature teams yet.

Creating feature teams comes at a cost: developers must learn new skills, which will slow them down initially. Fortunately, most developers enjoy learning new skills. Techniques such as appointing a technical “component guardian” for each area can help protect architectural integrity as teams are learning. As with any scaled development, continuous integration (automated tests that run much more frequently than once per day) is crucial to avoid undetected regression failures.

Once an organization has mastered feature teams, a next step could be general purpose feature teams with reduced affinity to feature areas. While such a step may be years away, this century will belong to the learning organization.

**Obstacle #4: Distraction**

A typical large organization wastes millions of dollars in unnecessary task switching. Teams lack focus and the months of continuous membership stability to reach the self-managing state needed for high performance and quality. Some people perpetually find themselves on multiple critical paths at the same time, severely constraining total productivity. For effective scaling, these people must become mentors rather than task executors. To increase their influence, they must relinquish some control.

Traditional management practices that reinforce specialization will exacerbate the issue. Work should be offered to a Scrum Team through the Sprint Planning Meeting. When a manager bypasses this by assigning work to an individual, there’s less possibility of mentoring others in the critical skills. In a Sprint Planning Meeting I observed recently, the first question was, “Who do we have on this Sprint?” A group of people being yanked around individually isn’t going to engage in the collaborative learning expected of a Scrum team. Organizations dig these ruts deeper each day.

In the higher ranks, the problems of multitasking, distraction, anxiety and inability to gain influence by letting go of control are even worse. The product owner at the same meeting rushed in nearly an hour late with a harried look on her face, then rushed out again after about 10 minutes; I was told this was routine.
That afternoon, an executive was unable to focus on a meeting with 60 of his employees because he was also trying to manage another situation via his BlackBerry. A casual observer of these scenes wouldn’t guess these frantic event-driven slaves were the ones responsible for long range strategy.

Effective lateral communication between Scrum teams helps product owners calmly focus on responsibilities such as prioritizing work and being the final arbiter of requirements questions.

**Obstacle #5: Reluctance to Continuously Refine, Reprioritize and Rescope**
In product development, new scope discovery commonly outpaces velocity. To manage this reality, product owners should conduct a product backlog refinement meeting with the team every sprint. As large product backlog items (or “epics”) emerge, find the high-value aspects of them and split those into separate smaller items (typically “user stories”). The product owner controls scope by deciding which items will fit in the release projection, given [historic trends of velocity and scope discovery](#).

**Obstacle #6: Rampant Technical Debt**
An organization’s management issues are ultimately visible in the source code. This “technical debt” causes production problems and high cost of change. Ideally, regression testing is automated using the same programming language our production code rather than proprietary tools that reinforce specialization. Skills such as Test Driven Development (TDD) proved their value in the 20th century. In the 21st century, agile engineering skills will be expected of any developer. Teams that don’t have these skills initially must attempt [thinner vertical slices of work until they’ve learned them](#).

**Obstacle #7: Lack of Commitment to Transformation**
Scrum allocates one person per team (the ScrumMaster) to devote full-time attention to exposing and overcoming obstacles such as these. This requires courage, imagination and support from management. Too few ScrumMasters devote proper attention to this, and management often fails to support those who do.

“Too much to do” or “not enough resources” or “changing business climate” aren’t good excuses for avoiding the agile practices specifically intended for those circumstances. Change agents will be better equipped to overcome obstacles to agility by realizing the root causes are often bad habits, unexamined reflexes and misconceptions.

Michael James is a software process mentor and Certified Scrum Trainer at CollabNet with a focus on the engineering practices that enable agile project management practices. Having worked as a software developer (formerly “architect”) for more than 20 years, he has deep experience with automated testing that predates the eXtreme Programming movement, including formal, phased, high-ceremony processes based on DOD-STD-2167A; chaotic non-processes of the dot-com era; and agile processes such as Scrum and XP. As an agile coach, James has found that teams learn best when he can work alongside them “in the trenches.” This one-on-one interaction allows him to share agile practices and insight while tackling real organizational challenges.

Copyright © 2010 gantthead.com All rights reserved.

The URL for this article is:
http://www.gantthead.com/article.cfm?ID=255033
Who are we?

- MJ is Scrum team member and software developer
  - See also 6-page illustrated guide to Scrum
  - ScrumMaster’s Checklist
    - http://ScrumMasterChecklist.org

- Who is CollabNet?
  - On site process and technical consulting
  - ScrumWorks project management tool
    - ScrumWorks Basic is free*
    - ScrumWorks Pro is easy to afford
  - Subversion, TeamForge
  - Agile process and technical training

* No cost download and license: http://danube.com/scrumworks

Activity: Propose a Strategy for Developing a Set Top Box

- You are developing a consumer product to deliver home entertainment services such as video on demand.
- Your company’s previous attempts at this kind of thing have run late, over budget, and had quality problems.
- The following people are available:
  - 5 C/C++ programmers, DSP experience
  - 15 Java programmers
  - 3 C/C++ programmers, microcontroller experience
  - 4 C/C++ programmers work on encryption in secure facility
  - 3 experts at configuring broadcasting system
  - 6 testers on the other side of the planet

- When you think you have some good ideas about team composition, suggest some small stories related to the epic of “high and low definition video on demand over a custom encrypted network with keys in the smart card driven by the remote control.”
What is New Product Development?

Product development is knowledge creation.

* The Knowledge-Creating Company (Harvard Business Review Classics), Ikujiro Nonaka

Adding Ingenuity to Productivity

**productivity**: the effectiveness of productive effort, esp. in industry, as measured in terms of the rate of output per unit of input: workers have boosted productivity by 30 percent.

**ingenuity**: the ability to solve difficult problems, often in original and creative ways
Letting Go Of Predictability

- Scrum is intended for work involving uncertain requirements and technology (e.g., new product development). Please check off the techniques you think are most applicable to this zone and cross out the ones that aren’t.
  - plan-driven approaches
  - value-driven approaches
  - empirical feedback loops
  - defined processes
  - people regarded as “resources”
  - communication primarily through artifact handoff
  - slack
  - economies of scale
  - intrinsic motivation
  - efficiency orientation
  - learning teams
  - static hierarchy
  - testing primarily at the end
  - heterogeneous teams
  - co-ordination roles (managers as go betweens)
  - frequent retrospection
  - static job titles
  - behaviorism (punishment/reward motivation)
  - rework
  - detailed roadmaps
  - team ingenuity
  - continuous planning/designing/building/testing
  - static best practices
  - metrics
  - precise estimation
  - communication primarily face to face
  - big plan/analysis/design up front

Fundamentals
Exercise: Sprints

In The Definitions of Scrum

- Is there such a Sprint as an “analysis Sprint”?
- Is there such a thing as a “testing Sprint”?
- What is a “stabilization Sprint”?
- If someone says a project requires a lot of infrastructure and architecture work that will take eight weeks to complete, should the first Sprint be eight weeks long?
  - Is the architecture an adequate deliverable?

Running (and Tested) Features

(Technical Debt is High Cost of Future Change)

Robust “done”

Weak “done”

Δ = Technical debt

Waterfall

Running (and Tested) Features

Time


Robust Definition of “Done”

The Potentially-Shippable Product Increment

Each Sprint should yield a potentially-shippable product increment*. If Product Owner said “Ship what you showed me,” would you be ready within one stabilization Sprint?

* “Potentially shippable” does not always mean potentially sellable.” -- Jeff Heinen, QPass Inc.
Weak Definition of “Done”

Stuff we procrastinated:
- refactoring
- load testing
- security testing
- user documentation
- technical debt
- UAT...

How big is YOUR “weak done” cloud?

Sprint

Sprint

Sprint

Sprint

Sprint

Product

Product

Product

Product

Shippable Product?

Smiley face

Smiley face

Smiley face

Smiley face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face

Frowning face
Large Scale Development, The Bad News

- "As project size grows the success rate goes down." – Per Kroll*
- "Scaling agile projects is the last thing you should do." – Martin Fowler
- One high-performing team in an optimized environment will outperform x average teams.
- Are you in the business of developing product, or keeping people busy?

"The matrix structure tends to raise the management head count for larger projects. Because software productivity declines as the management count goes up, this form of organization can be hazardous for software."†

* IBM Methods Manager, paraphrased in DDJ (http://www.ddj.com/architect/204200877)
† Software Assessments, Benchmarks, and Best Practices, Jones (2000)

Scaling Exercise

5 C/C++ programmers, DSP experience
15 Java programmers
3 C/C++ programmers, microcontroller experience
4 C/C++ programmers work on encryption in secure facility
3 experts at configuring broadcasting system
6 testers far far away

Teams Separated Along Layers

Common, and Not Recommended
Group Teams By Related Features

- One code base with continuous integration
- Informal "dotted line" working groups span teams as necessary to fend off technical debt, co-ordinate integration (early and often).

Scaling: Start Small

- Single Product Owner and Product Backlog
- Start with a single "Staging" team
  - Team prepares for scaling by building real product, establishing continuous integration and TDD practices, etc.
  - Team decides when it is ready to scale, if necessary

Staging Team Seeds New Teams

- Original staging team members (workers, not managers) seed new teams
- One single integrated product increment each sprint
- Each team needs a ScrumMaster, especially in the beginning.
**Example Scaled Planning Meeting**

- In the morning, original seed team tentatively commit sensible PBIs to the Sprint with PO, just as they did in Sprint Planning Part 1 when they were the only team.
  - In some cases seed team may involve PO in tentative plan of which spawned teams will do which items.
- In the afternoon, seed team members involve the spawned teams in task breakdown, often on separate taskboards.
  - PO remains available for clarification or renegotiation.
  - One large conference room subdivided for multiple teams is ideal.
- Final thumbs up from all team members at end of day commits the Sprint goals.

**Alternative: Multiple Virtual Products**

- Treat each team’s effort as a virtual “product” even when they are a single product from a marketing and build perspective.
- Separate backlogs.
- Separate planning and review meetings.
- One Product Owner can handle about three teams, or uber Product Owner can delegate to team Product Owners in traditional hierarchy.

**Scaling: Integration is Difficult**

- It must all come together as a single product increment.
- Try to integrate as often as possible, avoid big-bang integrations.
  - Each team integrates as often as possible, optimally this is continuous on check-ins.
  - Bugs produced by check-ins/integrations should be discrete enough such that the source should be easily identifiable.
  - The more continuous your integration, the easier it is to identify the source of bugs

- “Daily builds are for wimps” - Kent Beck
Example Implementation Roadmap

1. Implement Scrum skeleton (cross-functional teams, monthly deliveries, etc.)
2. Start measuring velocity (e.g., story points per iteration)
3. Co-locate team members (ideally in team rooms)
4. Simplify/agility source code configuration management practices
5. Begin using continuous integration (Detect merge/build failures immediately)
6. Build "sandboxes." (Independent environments for TDD)
7. Develop habit of test-driven development (Detect regression failures immediately)
8. Develop habit of radical refactoring, continuous design (Reduce impact of changes to requirements changes)
9. Begin using continuous integration (Detect merge/build failures immediately)
10. Assess and scale

Synchronizations Across Teams

- Scrum of Scrums is one of several practices.
- Who attends the Scrum of Scrums?
**Obstacles**

1. Naive Resource Management
2. Teams Organized by Functional Specialization
3. Teams Organized by Architectural Components
4. Distraction
5. Reluctance to Continuously Refine, Reprioritize, and Rescope
6. Rampant Technical Debt
7. Lack of Commitment to Transformation
8. Failure to Address Geographic Distribution Issues

* See also: *Obstacles to Enterprise Agility*, Michael James (2010)

---

**Distraction**

- “Not all organizations want to be great. That’s the hard truth. For those that do—that really do—the principles of great performance show quite clearly what it takes to get there.”*

* Talent Is Overrated: What Separates World Class Performers..., Geoff Colvin

---

- It’s possible for a person to have an overwhelming number of things to do and still function productively with a clear head and a positive sense of relaxed control.
  - *Getting Things Done: The Art of Stress-Free Productivity* (David Allen)
Clear Goals

- Almost every project could be done better, and an infinite quantity of information is now available that could make that happen.
  - Getting Things Done: The Art of Stress-Free Productivity (David Allen)

Challenge The 3%

- “In a well run change initiative, about 3% of the workforce will never change, no matter what you do.... If you don’t challenge the 3% and move them up or out, you will end up two years from now being very frustrated that you haven’t permanently changed your culture.”

* Topgun instructor pilot Stephen Harden, quoted at http://saferpatients.com
**Microefficiencies, Localized Optimizations**

**Steer the “What” not the “How”**

- Functional departments (IT, HR, Procurement, etc.) attempt to exploit *economies of scale* with “one size fits all” solutions and policies, restricting team innovation.

- Large companies allow innovation by giving up this control, allowing teams to act like small businesses:
  - Lockheed Skunkworks
  - world’s fastest airplane
  - highest-flying spyplane
  - first stealth aircraft
  - IBM PC
  - Microsoft XBox
  - Semco

---

**Optional Subheading: 23 pt Verdana Bold**

- “life at most companies seems almost intended to defeat all the principles of deliberate practice.”
  - *Talent Is Overrated: What Really Separates World Class Performers From Everybody Else*, Geoff Colvin

---

**Great Teams Face Outward**

- “The problem with coordination roles was painfully visible when we worked with a telecom messaging product. This product used Scrum and had ScrumMasters creating the conditions for self-organizing teams, but they refused to remove the project manager role. The result? The project manager became responsible for the coordination among teams and even for the communication to the Product Owner. He became stressed and overloaded with work.

- When we told him that his role is not needed, he laughed, and pointed out the amount of work he was doing. He did not realize that his role attracted the work, and that the work—the major bottleneck in the product group—would disappear if his role was removed.*

* *Scaling Lean & Agile Development*, Larman/Vodde, 2008
Stop Managing, Start Facilitating

• Another product group did remove project managers and let the teams coordinate their work. One of the line managers of this group reflected on this change and said, "Nobody ever missed them, and I have fifty percent more ‘free’ time."

* Scaling Lean & Agile Development, Larman/Vodde, 2008

Stop Managing, Start Facilitating

• When Rosing started at Google in 2001, "we had management in engineering. And the structure was tending to tell people, No, you can't do that." So Google got rid of the managers. Now most engineers work in teams of three, with project leadership rotating among team members. If something isn't right, even if it's in a product that has already gone public, teams fix it without asking anyone.

• "For a while," Rosing says, "I had 160 direct reports. No managers. It worked because the teams knew what they had to do. That set a cultural bit in people's heads: You are the boss. Don't wait to take the hill. Don't wait to be managed."


Coping with Geographical Challenges

• Why would you want a planet-sized impediment?

• Visit each other and work together long enough to be on an informal, first-name basis.
  – Formality and concerns about “looking good” are the enemy.

• Organize teams to reduce coupling over geographic boundaries.

• One onsite representative for several offsite team members in other timezone.

• Continuous Integration and TDD.

• Remote pair programming still experimental.
  – http://www.vimeo.com/1195398
• “So far as directing the actors, and the crew is concerned: well I direct just as little as possible, and I get as much from others as I possibly can.”
  – John Huston