Achieving Speed in Legacy Systems

Jan Bosch
Professor of Software Engineering
Chalmers University of Technology
Gothenburg, Sweden.
www.janbosch.com

September 2011
Keynote HPL 2011
“If you are not moving at the speed of the marketplace you’re already dead – you just haven’t stopped breathing yet”

Jack Welch
Three Key Take-Aways

• Increasing **SPEED** trumps ANY other improvement R&D can provide to the company – it is the foundation of everything else

• Teams should be **small**, **multi-disciplinary**, **self-selected** and **-directed**, use **data** (not opinions) for decision making and optimize quantitative output **metrics**

• Striving for **continuous deployment** is key to achieving speed in legacy systems
Overview

• Vem är jag? Wie ben ik? Who am I?
• Software trends: Need for Speed
• Challenges of legacy software engineering
• Innovation Experiment Systems
• Example: Intuit Quickbooks
• Implications
• Conclusion
From Research to Industry

- Open Innovation (Intuit, USA)
- Engineering Process (Intuit, USA)
- Head of research lab (Nokia, Finland)
- Professor of software engineering (RuG, Netherlands) (BIT, Sweden)

Innovation
Industrial development
Industrial research
Academia (+ consulting)
Software Center @ Chalmers

• Mission: Improve the software engineering capability of the Swedish Software-Intensive Industry with an order of magnitude
• Theme: Fast, continuous deployment of customer value
• Founding members

![Logos of Ericsson, Volvo, Volvo, Saab]

• Dual success metrics
  • Academic excellence
  • Tangible industrial impact
Overview

• Vem är jag? Wie ben ik? Who am I?

• **Software trends: Need for Speed**

• Challenges of legacy software engineering

• Innovation Experiment Systems

• Example: Intuit Quickbooks

• Implications

• Conclusion
Trend: Products to Services
Trend: Capitalism 3.0
Trend: Need for Speed

Value Creation Shifts

Emerging companies highlight the importance of user contribution and social connectedness.

<table>
<thead>
<tr>
<th>Founded</th>
<th>1984</th>
<th>1995</th>
<th>2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>1M users</td>
<td>~6 years</td>
<td>30 months</td>
<td>10 months</td>
</tr>
<tr>
<td>50M users</td>
<td>N/A</td>
<td>~80 months</td>
<td>~44 months</td>
</tr>
</tbody>
</table>
Need for Speed in R&D – An Example

• Company X: R&D is **10%** of revenue, e.g. **100M$** for a **1B$** product
• New product development cycle: **12 months**

• Alternative 1: improve efficiency of development with 10%
  • **10 M$** reduction in development cost
• Alternative 2: reduce development cycle with 10%
  • **100M$** add to top line revenue (product starts to sell 1.2 months earlier)

No efficiency improvement will outperform cycle time reduction
Team size
- 3x3 = 3 persons x 3 months (Google)
- 2 pizza rule (Amazon)
- Principle: What is required is a team, where the roles are defined and each member has the right skill for that role, and following a lean, agile, method — all focused on the customer.

Release cycle
- Weeks, not months
- Continuous deployment
- Principle: short cycles are key for agility, speed and decoupling

Architecture
- 3 API rule
- Mash-ups and web services
- Principle: architecture provides simplicity, compositionality and is designed in parallel with software development
Web 2.0 Rules to SW Development (2/2)

Requirements and Roadmapping

• Each team (3 persons) announces what they intend to release
• Some (QA) requirements are shared across the board, e.g. performance, latency, etc.
• Principle: the cost of overlapping teams is much lower than the cost of synchronized, planned roadmaps and plans

Process

• CMMi and other process maturity approaches address the symptoms, not the root cause
• Control is a very expensive illusion causing LOTS of inefficiency in the system
• Principle: Architecture, not process, should manage coordination and alignment

From the Cathedral to the Bazaar
Composition-Oriented SE

Team
• 2 pizza’s
• self-selected, directed and managed
• quantitative output metrics

Architecture
• simplicity – 3 API rule
• backward compatibility – no versions!
• focus on compositionality

Release process
• continuous, independent deployment
• all the way to customers – installed base
• measure usage to feed back into development
Overview

- Vem är jag? Wie ben ik? Who am I?
- Software trends: Need for Speed
- **Challenges of legacy software engineering**
- Innovation Experiment Systems
- Example: Intuit Quickbooks
- Implications
- Conclusion
Traditional Software Engineering

software product lines
global software development
software ecosystems

causing

unacceptable complexity and
coordination cost
Challenges

• Slow response to market changes
• Late customer feedback
• Inefficient requirements management
• Low utilization of engineering resources
• Opinion-based, politicized organization

Competitive position erodes over time
Overview

• Vem är jag? Wie ben ik? Who am I?
• Software trends: Need for Speed
• Challenges of legacy software engineering
• Innovation Experiment Systems
• Example: Intuit Quickbooks
• Implications
• Conclusion
What Do These Product Have in Common?
Example: Apple

<table>
<thead>
<tr>
<th>The Myth</th>
<th>The Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspired innovation</td>
<td>Create and winnow 10 pixel-perfect prototypes</td>
</tr>
<tr>
<td>Inspired design</td>
<td>Build a better backstory (intricate layers of business design behind the products)</td>
</tr>
<tr>
<td>Brilliantly inspired marketing</td>
<td>Engineer the perfect customer experience to create customer experience and buzz</td>
</tr>
</tbody>
</table>

R&D as an Experiment System

Learning: the company running the most experiments against the lowest cost per experiment wins

Goal: increase the number of experiments (with customers) with an order of magnitude to ultimately accelerate organic growth

Usage and other data

R&D iteration (4 weeks)

Three types of functionality
- Customer-requested
- Strategy driven
- Experiments

Installed Base (products @ customers)

Decide on new hypotheses to test based on data, ideas, strategy and customer requests

Decisions should be based on DATA, not opinions
## Techniques

<table>
<thead>
<tr>
<th>Pre-Development</th>
<th>Development</th>
<th>Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASES testing</td>
<td>Independently deployed extensions</td>
<td>Random selection of versions</td>
</tr>
<tr>
<td>Advertising</td>
<td>Feature alpha</td>
<td>Instrumentation of usage metrics</td>
</tr>
<tr>
<td>Solution jams</td>
<td>Product alpha</td>
<td>Surveys</td>
</tr>
<tr>
<td>Mock-ups</td>
<td>Labs website</td>
<td>Ethnographic studies</td>
</tr>
<tr>
<td></td>
<td>Product beta</td>
<td></td>
</tr>
</tbody>
</table>
Overview

• Vem är jag? Wie ben ik? Who am I?
• Software trends: Need for Speed
• Challenges of large scale software engineering
• Innovation Experiment Systems
  • Example: Intuit Quickbooks
• Implications
• Conclusion
Intuit Company Information

Who We Are…

A leading provider of business and financial management solutions

- Founded in 1983
- FY 2011 revenue of $3.85 billion
- Intuit is traded on the NASDAQ: INTU
- Employs around ~8,000 people
- Major offices across the U.S. and in Canada and the United Kingdom
- More than 50 million people use our QuickBooks, Payroll, Payments, TurboTax, Digital Insight and Quicken products and services.
Proven formula: lots of delighted customers...

Help small businesses be 20% more profitable... Customers revenues ~20% of U.S. GDP, pay 1 in 12 American workers

Help families find $1,000 annually... $400M in consumer savings

Help people get the maximum tax refund... $33B in tax refunds, 1 out of every 3 tax returns e-filed

Improve FI profit per customer by 20%... IB customers equal to the 5th largest U.S. bank

Help accountants be 20% more productive today... Serve half of all accounting firms

Intuit® Improving 50M Lives
Great Brands and Great Products
Fortune Top 100 Places to Work
Intuit Tops the List: 5 Years Running!
2009 - No. 1
2008 – No. 1
2007 – No. 1
2006 – No. 1
2005 – No. 1
Did You Know?

Intuit has some of the strongest brands in the industry

- 15 million Quicken customers
- Nearly 7 million small businesses are Intuit customers
- More than 14 million federal desktop and Web TurboTax units (Tax Year 2006)

Source: NPD, company estimates
Intuit’s Game Plan To Win

To be a **premier innovative growth** company that empowers individuals and businesses to achieve their dreams

Easy-to-use **“Connected Services”** that create delight by solving important unsolved customer problems & build durable advantage

1. **Social** ... capitalize on our large and growing customer bases to unleash the collective power of user contributions, user behaviors and user data

2. **Mobile** ... deliver “in the pocket” when that is the preferred solution

3. **Global** ... employ the world’s talents to find & solve important problems around the globe
Quickbooks

- Age: 20 years old
- Size: 5-10 MLOC
- Org: 100+ R&D staff
Old Development Process: Yearly Releases

- **January/February**: Development starts for real
- **May/June**: Freeze all development
- **September**: Release
- **Figure out what to build**
- **Get a beta out and fix bugs (and then some more bugs)**
- **Build the key new features**

*Figure out what to build*
Old Development Process: Problems

- Lack of customer feedback
- Heavy, top-down development process
- Inefficient use of engineering resources
- Low engagement of team members
New Process: Overview

Top Down: Strategic Areas to Improve

Solution Jam

Code Jam

1..n iteration

Feature Alpha

Product Release

~50% acceptance  ~50% acceptance
Solution Jam

- Goal: Get as early feedback on an idea or concept as possible
- Length: 1 day
- How:
  - Invite staff to jam
  - Request “pain statements” beforehand
  - Select 10-15 customers based on “pain statements”
  - Staff self-organizes into small teams (3-6 typical)
  - Teams develop mock-up solutions to selected “pain statement”
  - Customers provide feedback on the mock-ups
  - Teams present at end of day
  - Customers + PM&PD leaders select most promising concepts
- Watch out for
  - Opinions instead of data
Code Jam

• Goal: assess difficulty of implementing concept in the legacy code base
• Length: 1-2 days
• How
  • Teams build skeleton implementation of concept in copy of code base
  • Teams assess implementation challenges and dependencies on existing code
  • Teams present and PM&PD leaders select most promising and realistic to implement concepts
• Watch out for
  • Shadow beliefs
Iteration

• Goal: prove one customer hypothesis
• Length: 2-3 weeks
• How
  • Team gets 2-3 weeks to build one end-to-end slice of the concept
  • Works with one or more customers to get feedback on implementation
  • Presents progress and customer feedback for go/no-go decision for next iteration
• Watch out for
  • Not involving the customer throughout
Feature Alpha

• Goal: Evaluate implementation of concept by customer at customer premise
• Length: 2-4 weeks
• How:
  • Prepare “packaged” release of last stable product + concept implementation (not production quality)
  • Provide to customer to evaluate (using copy of customer data)
  • Collect feedback for further improvement
• Watch out for
  • Aim for maximized realism of evaluation
Key Characteristics

• **Self-selected, directed and managed teams**: A major part of R&D is organized in small teams that self-select their members, self-direct (based on customer input) their work and that, to a large extent, manage themselves.

• **Continuous customer involvement**: Customers are involved throughout the development process.

• **Solution and code jams**: One and two-day jams are an integral part of the development process and have a dedicated purpose, i.e. converting an idea into a real, partially validated, concept and exploring the complexities of implementing it in the legacy code base.

• **Feature alphas**: The ability to present the customer with a stable version of the most recent release of a product in the software product line but extended with a specific feature that is under development has proven to be a particularly powerful customer feedback mechanism.
Overview

• Vem är jag? Wie ben ik? Who am I?
• Software trends: Need for Speed
• Challenges of large scale software engineering
• Innovation Experiment Systems
• Example: Intuit Quickbooks

- Implications
- Conclusion
Implications

• Disrupt your industries business model

• Go where the money is: “after-market sales” & services

• Develop the ability of your products to evolve constantly

• Minimize R&D investment between customer proof points (data, not opinions!)
Three Layer Platform Model

**Commoditized Functionality Layer** (optimize for minimizing total cost of ownership)

**Differentiating Functionality Layer** (optimize for maximum platform value for customers)

**Innovation and Experimentation Layer** (optimize for # experiments)

**Platform Interface**

**Challenges**
- Over time platforms lose competitiveness
- Platform becomes competitive disadvantage

**Characteristics**
- Each layer releases independently
- Each layer optimizes different metrics
- Focus R&D effort on highly differentiating functionality
Overview

• Vem är jag? Wie ben ik? Who am I?
• Software trends: Need for Speed
• Challenges of legacy software engineering
• Innovation Experiment Systems
• Example: Intuit Quickbooks
• Implications

• Conclusion
Speed

Increasing **SPEED** trumps ANY other improvement R&D can provide to the company – it is the foundation for everything else

• As a process, methods or tools professional, there is only ONE measure that justifies your existence: how have you helped teams move faster?
• Don’t optimize efficiency, optimize speed
Teams should be **small**, **multi**-disciplinary, self-**selected** and -**directed**, use **data** (not opinions) for decision making and optimize quantitative output **metrics**

- Small and self-*: 2 pizzas
- Multi-disciplinary: product management, experience design, engineering, testing all represented
- Metrics: help teams know how they’re doing and link their metrics to the business strategy
Continuous Deployment

- Striving for **continuous** deployment is key to achieving speed in legacy systems
- Decouple components, decouple teams and decouple organizations
- Invest in the automated testing and deployment infrastructure
- Bring in the customer early and often
- *A la Thoreau: Simplify, Simplify, Simplify*
Not My Job?!
Thank you!

Q&A