Speed, Innovation and Simplicity through Software Architecture

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

February 2014 Industrial Engineering and Management Guest lecture "If you are not moving at the speed of the marketplace you're already dead – you just haven't stopped breathing yet"

Jack Welch

On Education ...

White youst high the becatives isotally a straight by the bis of t

PeterefeloBooker (fermenanresidentefitearrandageinersity)

Three Key Take-Aways

- Increasing SPEED trumps ANY other improvement R&D can provide to the company – the goal is continuous deployment of new functionality
- Software engineering is at an inflection point from "integration-oriented" to "composition-oriented" software engineering
- Software architecture is key to build delightful products in the context of software ecosystems

Overview

Vem är jag? Wie ben ik? Who am I?

- Trends in Software: Need for Speed
- Innovation Experiment Systems
- Software Ecosystems
- Architecture & Scale
- Implications for ICT Professionals
- Conclusion



Software Center @ Chalmers

- Mission: Improve the software engineering capability of the Nordic Software-Intensive Industry with an order of magnitude
- Theme: Fast, continuous deployment of customer value
- Dual success metrics
 - Academic excellence
 - Tangible industrial impact







GRUNDFOS







MALMÖ UNIVERSITY

Overview

- Vem är jag? Wie ben ik? Who am I?
- Trends in Software: Need for Speed
- Innovation Experiment Systems
- Software Ecosystems
- Architecture & Scale
- Implications for ICT Professionals
- Conclusion

Trend: Products to Services













Innovation Approaches



This requires continuous experimentation with customers

You should wake up every morning terrified with your sheets drenched in sweat, but not because you're afraid of our competitors. Be afraid of our customers, because those are the folks who have the money. Our competitors are never going to send us money.

- Jeff Bezos

Trend: Need for Speed

Value Creation Shifts

Emerging companies highlight importance of user contribution and social connectedness



Level of User Contribution

Founded	1984	1995	2004
1M users	~6 years	30 months	10 months
50M users	N/A	~80 months	~44 months

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)

No efficiency improvement will outperform cycle time reduction

Need for Speed - Principles



Team

•2 pizza's

self-selected, directed and managedquantitative 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?
- Trends in Software: Need for Speed
- Innovation Experiment Systems
- Software Ecosystems
- Architecture & Scale
- Implications for ICT Professionals
- Conclusion

What Do These Product Have in Common?











Example: Apple

The Myth	The Reality
Inspired innovation	Create and winnow 10 pixel-perfect prototypes
Inspired design	Build a better backstory (intricate layers of business design behind the products
Brilliantly inspired marketing	Engineer the perfect customer experience to create customer experience and buzz

Reference: http://blogs.hbr.org/cs/2011/08/steve_jobs_and_the_myth_of_eur.html

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



Decisions should be based on DATA, not opinions

We have an unprecedented opportunity to run A/B tests with online users and innovate more quickly based on actual user response. Microsoft needs to shift the culture from planning the exact features to planning a set of possible features, and letting customers guide us.

- Ray Ozzie

Stages and Techniques

	Pre- Development	Non-commercial deployment	Commercial deployment
Optimization	Ethnographic studies	Independently deployed extensions	Random selection of versions (A/B testing)
New features	Solution jams	Feature alpha In-product surveys	Instrumentation/ collecting metrics
New Products	Advertising Mock-ups BASES testing	Product alpha Labs website In-product advertising	Surveys Performance metrics

Pre-Development: Concept Testing

• What

- Technique to test a concept before start of development.
- Active Customer: Respondents are exposed to graphical or written product description. Answer a survey
- Passive customer: Cross-sell, upsell or market a nonexisting product (e.g. AdWords) to measure market interest
- To think about
 - Measure the conversion funnel
 - Consider A/B testing on your ad & pages

Quickbooks Online

- Cross-sell and upsell "ads" in the online product
- Measure conversion of those ads
- Decide on development of feature/product based on data

Pre-Development: 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

Example: Quickbooks



Development: Feature Alpha

- What
 - Release last stable release of product with one new feature (under development) to selected customers
- Variations
 - Operate the product on copy of customer data
 - Allow user to turn feature on or off
- To think about
 - NEVER risk customer data
 - Measure everything test alternative implementations

Example: Quickbooks



Development: Labs Site

- What
 - Create destination for customers to try out new potential products, extensions and mock-ups
- Alternatives
 - Labs destination embedded in product or at company level
 - Combine with 3rd party developer efforts
- To think about
 - Brand impact of immature products
 - Energizing the innovation muscle of the company

Intuit Labs

Intuit. LABS

APPLICATIONS INNOVATORS PROCESS CONNECT



Evolution: A/B Testing

- What
 - A/B testing is a method of comparing a baseline control sample to a variety of single-variable test samples for improving some metric
- Alternatives
 - "Marketing" testing, e.g. colors, buttons and order of options
 - Alternative implementations of a feature
- To think about
 - Run multiple experiments simultaneously
 - Verify statistical relevance (free online tools exist)

Case: Open Infotainment Labs



Case: Open Infotainment Labs

- Feature development from a nominal lead-time of 1-3 years to 4-12 weeks?
- Working software was continuously validated in "real" environments
 - installed in both a driving simulator and real test cars
 - users evaluated the system
- 4th sprint: A/B experiment
 - Evaluating two layouts of the start screen
 - Implemented as two different launchers in Android
 - Mounted in a vehicle
 - 7 test drivers in total (3 used A, 4 used B)

A/B Testing Examples

- 37signals tested the headline on its pricing page. It found that "30-Day Free Trial on All Accounts" generated 30% more sign-ups than the original "Start a Highrise Account."
- Dustin found that "You should follow me on Twitter here" worked 173% better than his control text, "I'm on Twitter."
- A surprising conclusion from two separate A/B tests: putting human photos on a website increases conversion rates by as much as double.
- CareLogger increased its conversion rate by 34% simply by changing the color of the sign-up button from green to red.
- A software product company redesigned their product page to give it a modern look and added trust building elements (such as seals, guarantees, etc.). End result: they managed to increase total sales by 20%.

We don't have better algorithms. We just have more data.

Peter Norvig Chief Scientist, Google











Rough estimation of waste and benefits

Financial Impact Potential

Ericsson

- R&D budget July 2011 June 2012: 4,864 M\$
- Software R&D (80%): 3891 M\$
- Value of removing 5% waste:
 195 M\$ (1280 MSEK)

AB Volvo

- Revenue 2011: 310 BSEK
- R&D budget 2011 (est. 5%): 16 BSEK
- Software R&D (15%):
 2.4 BSEK
- Value of removing 5% waste:
 120 MSEK

Overview

- Vem är jag? Wie ben ik? Who am I?
- Trends in Software: Need for Speed
- Innovation Experiment Systems
- Software Ecosystems
- Architecture & Scale
- Implications for ICT Professionals
- Conclusion

Towards Web 3.0

3 Atomisation. Globalisation and networking technologies will enable firms to use the world as their supply base for talent and materials. Processes, firms, customers and supply chains will fragment as companies expand overseas, as work flows to where it is best done and as information digitises. As a result, effective collaboration will become more important. The boundaries between different functions, organisations and even industries will blur. Data formats and technologies will standardise.



⁴⁴ My prediction would be that Web 3.0 will ultimately been seen as applications which are pieced together. There are a number of characteristics: the applications are relatively small, the data is in the cloud, the applications can run on any device, PC or mobile phone, the applications are very fast and they're very customizable. Furthermore, the applications are distributed virally: literally by social networks, by email. You won't go to the store and purchase them... That's a very different application model than we've ever seen in computing.

Evolution of Development Approaches



Software Ecosystem?

- Here's a try: A software ecosystem consists of a software platform, a set of internal and external developers and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs.
- Some more detail:
 - Software platform: A hierarchical set of shared software components providing functionality that is required and common for the developers constructing solutions on top of the platform.
 - **Evolution**: Over time, the functionality in the ecosystem commoditizes and flows from unique solutions to the platform.
 - **Developers**: Although internal and external developers use the platform differently, the platform often allows developers to build on top of each other's results.
 - Composition: Users are able to compose their own solutions by selecting various elements into a configuration that suits their needs optimally.

Why Software Ecosystems?

- Increase value of the core offering to existing users
- Increase attractiveness for new users
- Increase "stickiness" of the application platform, i.e. it is harder to change the application platform
- Accelerate innovation through open innovation in the ecosystem
- Collaborate with partners in the ecosystems to share cost of innovation
- Platformize functionality developed by partners in the ecosystem (once success has been proven)
- Decrease TCO for commoditizing functionality by sharing the maintenance with ecosystem

Taxonomy of Software Ecosystems

end-user programming	MS Excel, Mathematica, VHDL	Yahoo! Pipes, Microsoft PopFly, Google's mashup editor	none so far
application	MS Office	SalesForce, eBay, Amazon, Ning	none so far
operating system	MS Windows, Linux, Apple OS X	Google AppEngine, Yahoo developer, Coghead, Bungee Labs	Nokia S60, Palm, Android, iPhone
category platform	desktop	web	mobile

Overview

- Vem är jag? Wie ben ik? Who am I?
- Trends in Software: Need for Speed
- Innovation Experiment Systems
- Software Ecosystems
- Architecture & Scale
- Implications for ICT Professionals
- Conclusion

Role of Software Architecture

- Simplify, Simplify, Simplify
- Decoupling
 - Components
 - Teams
 - Organizations
- Lean and agile at scale
- End to end quality requirements
- Fight design erosion



Simplify, Simplify, Simplify

 Each architectural design decision adds design rules and constraints that cause complexity

 Insist on simplicity (3 APIs rule)

- How
 - Push down in the stack
 - Hide
 - Automate
 - Redesign



Decouple Teams and Organizations

- Interconnected teams and organizations asymptotically reduce productivity to *zero*
- Decouple teams and make sure no continuous interaction is needed
- How
 - Continuous deployment
 - No versions
 - No concurrent development



Decoupling: No Versions!



Decouple Components and Teams

1

3

- Sequential feature development (90%)
- Concurrent development, independent deployment enforced (8%)
- Exploratory development (2%)



Strive For Continuous Deployment

- Software engineer checks in code => system compiles, links, tests and deploys the new code
- The automated QA infrastructure, NOT the engineer, is responsible for making sure the system does not go down
- If that's too much, aim for Independent Deployment
- If that's too much, aim for Release Trains

Lean and Agile at Scale

- Achieving lean & agile in large, legacy systems with large R&D organization considered an oxymoron
- Google, Amazon and Intuit are examples that it can be done
- How
 - Small teams
 - Short cycles
 - Direct customer connection
 - Clear success metrics



End to End Quality Requirements



Evolve Architecture; Fight Erosion



Overview

- Vem är jag? Wie ben ik? Who am I?
- Trends in Software: Need for Speed
- Innovation Experiment Systems
- Software Ecosystems
- Architecture & Scale
- Implications for ICT Professionals
- Conclusion

Implications for ICT Engineers



Multi-disciplinary Learn continuously Self-starting Love customers Understand business Drive to metrics Build networks Move fast Entrepreneurial

Shadow Beliefs

• Humans are better than machines in identifying known and new reliability issues – we are building business critical systems, after all!

My experience: data always trumps opinion; test and validation systems pre-deployment and extensive data-collection post-deployment inform decision making

• Software-intensive systems (large, complex, tough requirements) are different and approaches from other domains do not apply

My experience: system failure is devastating in several industries and avoided in Internet systems while adopting agile and continuous

We should avoid or delay adoption of new, more efficient engineering approaches

My experience: getting first to market with new functionality that closely aligns to customer needs is a significant competitive advantage that

drives growth and results in market leadership

Guidelines

- 1. Modularize the system in critical and (less or) not critical parts
- 2. Adopt agile and continuous deployment approaches for the not (so) critical part first
- 3. Deeply engage with customers to develop optimal solutions to their real pain points
- 4. Invest in testing infrastructure that continuously and thoroughly tests systems with no human involvement
- 5. Instrument systems for pre- and post-deployment data collection, concerning at least reliability and usage metrics
- 6. Architect your systems for maximum decoupling and modularization between different components to allow for independent deployment
- 7. Replace commoditizing functionality with Open Source or COTS components; focus R&D on truly differentiating parts

Overview

- Vem är jag? Wie ben ik? Who am I?
- Trends in Software: Need for Speed
- Innovation Experiment Systems
- Software Ecosystems
- Architecture & Scale
- Implications for ICT Professionals

Conclusion

Speed

Increasing **SPEED** trumps ANY other improvement R&D can provide to the company – the goal is **continuous deployment** of new functionality

•If you're not a front-line engineer, there is only ONE measure that justifies your existence: how have you helped teams move faster?

Don't optimize efficiency, optimize speed

Inflection Point

 Software engineering is at an inflection point – from "integration-oriented" to "composition-oriented" software engineering

- Design for automated compositionality, not manual integration
- Minimize dependencies
- Focus on small teams of engineers, give them direction and get out of their way

Software Architecture 2.0

 Software architecture is is central in allowing for independent, continuous deployment to customers

- Architecture happens (in parallel)
- A la Thoreau: Simplify, Simplify, Simplify
- Decouple components, decouple teams and decouple organizations
- Lean and agile at scale

Not My Job?!



Strong LEADERSHIP needed from YOU

Thank you!

