Building Software Large and Small

Notes from the Field

Dennis Lee
About Me

• Grew up in Philippines
• Cornell University – 1992
• University of Washington – 1999
• Amazon.com – 1999-2006/2008-now
• Marchex – 2006-2008
• In China – 2010 – 2013
• Kindle Bookstore - today
My experience at Amazon

- Supply Chain systems (2.5 years)
- Website Merchandising (1.5 years)
- Website Operations (2 years)
- Grocery Delivery Logistics (1.5 years)
- Amazon China Website (3 years)
- Kindle (current)
Software Design Process Fails

The Health Insurance Marketplace online application isn't available from approximately 1 a.m. to 5 a.m. we make improvements. Additional down times may be possible as we work to make things better. The app and the Marketplace call center remain available during these hours.

Find health coverage that works for you

Get quality coverage at a price you can afford.
Open enrollment in the Health Insurance Marketplace continues until March 31, 2014.

APPLY ONLINE
APPLY BY PHONE
Theory vs. Practice

• Theory –
  – Agile – incremental delivery, people over process
  – Waterfall = BAD

• Practice
  – Multi-year projects, Gantt charts, “sprints”, unhappy customers --- effectively waterfall

• Why?
Goal of Software Process

• Deliver an agreed upon working piece of software to the customer at an agreed upon time.
Relative Cost of Bug Fixes

Process

• Goal: Deliver an agreed upon working piece of software to the customer at an agreed upon time.

• Observations:
  – *Bugs found earlier in the development process are less costly to fix*

• Solution:
  – *Invest more earlier in the process to get it right*
Communication Gets Expensive

- Communication paths with two programmers: 1
- Communication paths with three programmers: 3
- Communication paths with four programmers: 6
- Communication paths with five programmers: 10
- Communication paths with ten programmers: 45
Process

• Goal: Deliver an agreed upon working piece of software to the customer at an agreed upon time.

• Observations:
  – Bugs found earlier in the development process are less costly to fix
  – Different pieces of the system require different skillsets
  – Communication is expensive

• Solution:
  – Invest more earlier in the process to get it right
  – Break problem into sub-systems and defer having teams talk until needed
Learning from Mistakes

• Typical System – built quickly to meet pressing need. No time to [...]. Just get it out.

• Typical Issues:
  – Doesn’t scale
  – Isn’t flexible enough
  – Design is ugly
Process

• Goal: Deliver an agreed upon working piece of software to the customer at an agreed upon time.

• Observations:
  – Bugs found earlier in the development process are less costly to fix
  – Different pieces of the system require different skillsets
  – Communication is expensive
  – Large amount of time is spent to retrofit systems for scalability, flexibility and poorly designed

• Solution:
  – Invest more earlier in the process to get it right
  – Break problem into sub-systems and defer having teams talk until needed
  – Design scalability, flexibility, security, testability, etc. into the system up-front
My First Intern’s Project

• Goal: system to kill the buy box on an item in real time

• Schedule (12 week internship):
  – Ramp up (2 weeks)
  – Build Database (2 weeks)
  – Build Service (3 weeks)
  – Build UI (3 weeks)
  – Integrate and Test (2 weeks)

• What really happens:
  – Things took more time
  – By the end of the internship – everything was “done” – but still need to integrate and Test
Vicious Cycle

1. Delayed release
2. Uncertainty on hitting the target
3. Add more features to make sure we hit the target
Delivering Value

• Value is judged by Customers when they try out the product
• We are likely building the wrong product
• Everything Changes

Summary: No one can predict the future
Process

- Goal: Deliver an agreed upon working piece of software to the customer at an agreed upon time.
- Observations:
  - Bugs found earlier in the development process are less costly to fix
  - Different pieces of the system require different skillsets
  - Communication is expensive
  - Large amount of time is spent to retrofit systems for scalability, flexibility, and poorly designed
  - Integration is non-trivial
  - *No one can predict the future*
- Solution:
  - Invest more earlier in the process to get it right
  - Break problem into sub-systems and defer having teams talk until needed
  - Design scalability, flexibility, security, testability, etc. into the system up-front
  - ????
Modified Solution

• Invest more earlier in the process to get it right
  ➢ Deliver value to the customer as early as possible

• Break problem into independent pieces and defer having the teams talk until needed
  ➢ Invest in cross functional teams that can execute on all levels of the stack

• Design scalability, flexibility, security, maintainability, testability, etc. into the system up-front
  ➢ Accept that these are “problems from success”

  ➢ Invest to make retrofitting as cheap as possible – Quality
Modified Goal

• Deliver an agreed upon working piece of software to the customer at an agreed upon time

• Deliver Value to Customers as quickly as possible
Keeping software quality high

• Code in repository is “ready-to-deploy” all-the-time
• Constantly write new tests and modify tests to adapt.
• Estimated test and refactor tax: 50% of dev time
• Major production issues always have a post-mortem:
  – Always ask – could we have caught this in test?
  – Tests are written
  – Monitoring is updated
• Dedicated people to work on defects
• Interrupt stories if we have a bad quality week
• Infrastructure projects are scheduled with other stories
In Practice – My Teams

• Minimize formal specs
• Deliver often
• Customer focus
• Minimize Work-in-progress
• Lots of experimentation
• Keep the code clean and well tested
Case Study
Amazon Fresh Picking Rewrite

• Goal: Improve Efficiency of Picking

• Requirements going in:
  – Gather weight and dimension data for all items
  – Virtually pack items into totes
  – Weigh the tote to check for picking errors
  – Scan check picking supplies (e.g., ice packs)
What did we do?

• Gather weight and dimension data for all items – Yes

• Virtually pack items into totes - Yes – but...

• Weigh the tote to check for picking errors - No

• Scan check picking supplies (e.g., ice packs) - No
Summary

• Deliver Often
• Limit Work-in-Progress
• Stay in touch with your customers
• Keep Code Clean and Testable
Open Question

- Testing/Verification – can we make it much easier and more natural?
  - Proof systems are a start but it’s still hard to specify
  - Nothing “nags” at you – it’s too easy not to do
  - It still takes too much discipline and “try harder”
Traditional

Components are developed separately. Manager likes to say “complete each component then move on”

Integration late in project

Unexpected behaviors discovered very late

System testing at end

Schedule

Eye done
Ear done
Leg done
Hand done

Agile

Simple end-to-end functionality is achieved at an early stage

Regular system testing throughout

Each component gets gradually richer features and behavior
High confidence in product from an early stage

Questions?

Contact: dennisl@amazon.com