Economic Arguments for Agile
Why agile saves money

- Benefit of early delivery
- When you fix bugs
- Fixing scope up-front
- Cost of maintenance
- Team size
- Killer core
Benefit of Early Delivery

- Getting return earlier compounds in value over time (time-value of money)

- Reaching market sooner may have a huge impact
Cost and Benefit of Traditional Development

Money

Months

- cost
- cum cost
- benefit
- cum benefit
Traditional vs Agile
(one project)
Traditional vs Agile (two projects)
Traditional vs Agile
(two projects)

Money

Months

agile

traditional
Fixing Scope Up-Front

- Early decision on scope means loss of flexibility
- The last 20% requires more than 20% of the budget
- Can we know which features are really required?
- What’s the return of a feature built but never used?
When You Fix Bugs

- Bugs are cheaper to fix the earlier you find them (Boehm, Software Engineering Economics, 1981)
Cost of Maintenance

- Old apps never die
- Staff can’t always grow
- Innovation gets squeezed
Team Size

- QSM
  - Consultancy specializing in measuring, estimating, and controlling software dev
  - Database of 4000+ projects
- 2005 study on schedule vs team size
- 564 information systems projects since 2002
  - Divided into small (< 5) and large (> 20)
  - By team size
Team Size, cont.

- For projects of 100,000 SLOCs:
  - Peak staffing of project
    - Average large team: 32 people
    - Average small team: 4 people
  - Total effort (person months) for project
    - 178 p.m for large teams ($2.1 M)
    - 24.5 p.m. for small teams ($0.3 M)
Team Size, cont.

Calendar time to complete project
9.12 months for large team
8.92 months for small team

The one week shaved off delivery cost $1.8M

Explanations?
Communication and coordination inefficiency
Greater rate of defects (5x)
Killer Core

• From a talk by Ken Schwaber at Agile 2006 ("Agile Quality: A Canary in a Coal Mine")

• Agile
  • $value = f(time, quality, scope, cost)$

• Traditional
  • $value = f(time, quality)$
The Meaning of Done

- Change the meaning of “done” to hit a deadline: stabilization phase, user test, alpha, pre-release

- Reduce quality with these common behaviors
  1. overtime and weekends
  2. cut testing
  3. cut reviews
  4. don't following standards
  5. no refactoring
Belief in Magic

• Customer believes magic will happen (“we’ll deliver X by date Y”)

• Developers lie to their customers (“we can make it”), then cut quality to make it so
Core Functionality

• Aka “legacy”, “core”, “base”
• Velocity to work in this code is much lower
• How does this come to be?
Path of Doom

- New development
- 6 months, velocity of 18
- Deadline is 5 months
- Quality cut to make the date
- Team is a hero
Subsequent Projects

- 6 months of work, 5 month deadline, velocity 17, cut quality, hit date
- 6 months of work, 5 month deadline, velocity 15, cut quality, hit date
- 6 months of work, 5 month deadline, velocity 10, cut quality, hit date
- ...
What’s Usually Done

• Build core into each function
• Build new functionality and don't worry about core (fake "done")
• Drop functionality
• Give developers to core teams to increase velocity
• Start rebuilding the core functionality
The Death of a Company

- Ken says: 3-10 years for a company to back itself into a corner with their “core”

- Opens a window of opp for competition (innovative, not hobbled by slower dev)

- Every $1 saved costs $4
  - should be a top management decision
  - reflected on the balance sheet

- Cost of supporting magical beliefs is hidden