Agile Revisited

Dan North

@tastapod

1990s development

- Large projects

- Functional silos

- Slow, fragmented technology

- Process modelled on Civil Engineering
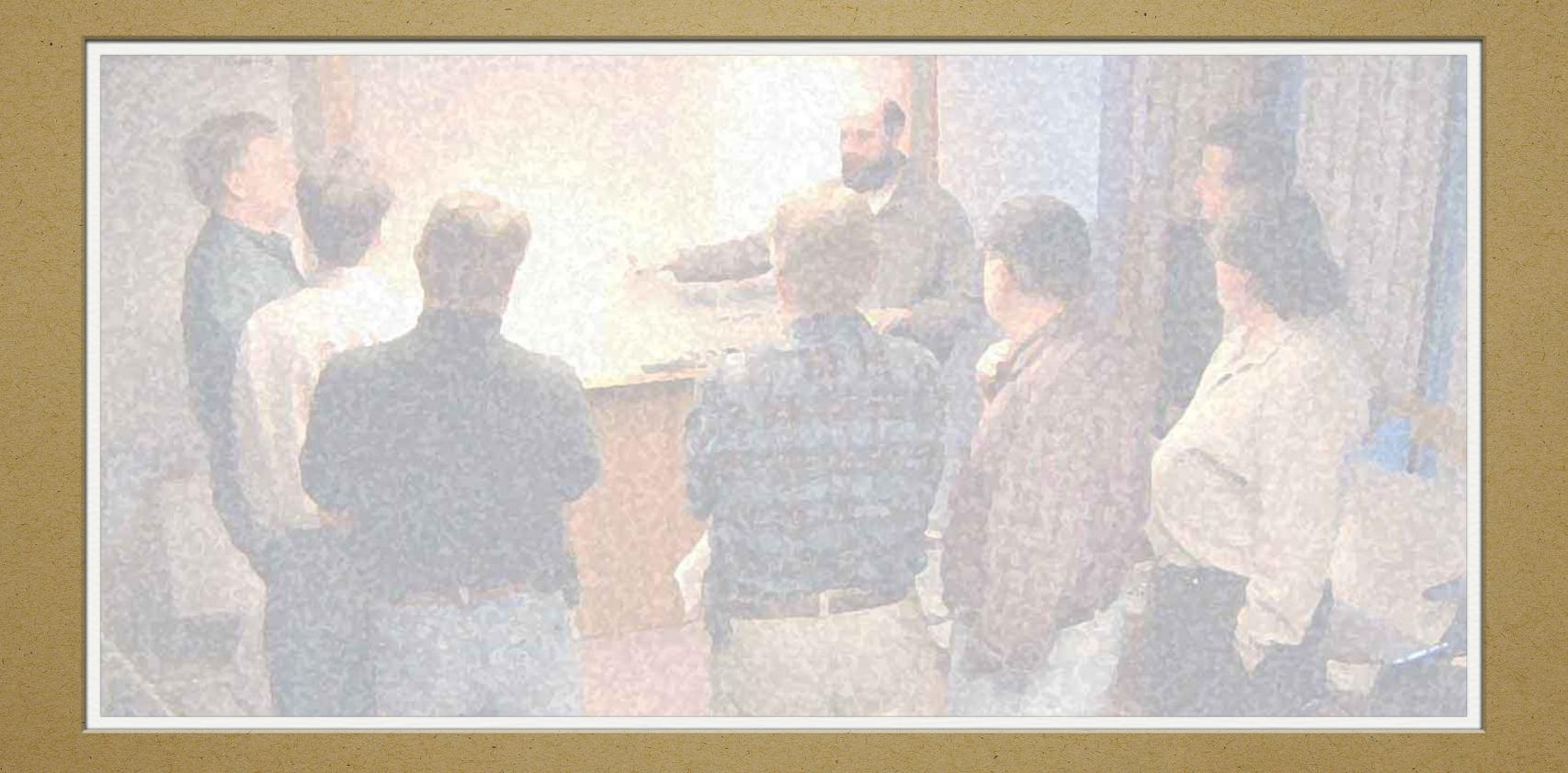
Civil Engineering

- Cost of errors rises exponentially

- Assurance through formal hand-offs

- Plan is intolerant of slippage

- Hand-offs are detailed and expensive

Software Engineering

- *Assumes*: Cost of errors rises exponentially

- *Assumes*: Assurance through formal hand-offs

- Plan is intolerant of slippage

- Hand-offs are detailed and expensive

Agile Then

Early and continuous delivery of software

Working software is measure of progress

Welcome changing requirements

Self-organising teams

Deliver frequently

Technical excellence and good design

The Manifesto

Business and developers working together

Simplicity

Build projects around motivated individuals

Sustainable pace for sponsors, users, team

Value face-to-face communication

Regular reflection and tuning

"**Brand**(vt): give a product a distinctive identity"

The Brand

Early and continuous delivery of software

Welcome changing requirements

Deliver frequently

Working software is measure of progress

Self-organising teams

Technical excellence and good design

The Scrum Agile Brand

Business and developers together

Build projects around motivated individuals

Value face-to-face communication

Simplicity

Sustainable pace for sponsors, users, team

Regular reflection and tuning

@tastapod

enior VP of Card IT at Capital One, w

products a month. Currently Capital C

ers trained on the Agile methodology

o Agile, including Agile team size, co

nd meeting expectations on the first

Agile Now

2010s development

- Smaller projects

- Cross-functional "feature teams"

- Incremental delivery

2010s development

Upstream batching of money

except…

Downstream batching of Release

2010s development

Water-

Scrum

-fall

Agile Next

@tastapod

"Move the people to the work"

…but remember they are people!

- Build your own Light Saber

- Embrace radical diversity

- Figure out what a "team" looks like at scale

Measure Business Impact

- We can build-ship-measure fast enough

- Less is more, like surgery

- Developer "productivity" isn't a thing

- Neither is velocity

Assume technology is instant* and free*

at least compared to the 1990s

Write

Build

Provision

Deploy

Monitor

Embrace Continuous Delivery

- Two weeks is an illusion, two months doubly so

- Outcomes create options, requirements emerge

- Rolling Op-Ex *over* committed Cap-Ex

- Investment collaboration *over* detailed estimation

Would you use a 1990s computer?

Then why use a 1990s method?

Early and continuous delivery of ~~software~~ *value*

~~*Working software*~~ *Business impact* is *measure of progress*

Welcome ~~*changing*~~ *emerging requirements*

Self-organising teams

Deliver ~~frequently~~ continually

Technical excellence and good design

The Manifesto

Business and developers and everyone else working together

Simplicity

Build ~~*projects*~~ *products around motivated individuals*

Sustainable pace for ~~sponsors, users, team~~ all stakeholders

Value face-to-face communication

~~*Regular*~~ *Continual reflection and tuning*

@tastapod

Early and continuous delivery of ~~software~~ value

~~Working software~~ *Business impact* is measure of progress

Welcome ~~changing~~ emerging requirements

Self-organising teams

Deliver ~~frequently~~ continually

Technical excellence and good design

Not bad for a first draft!

Business and developers and everyone else working together

Simplicity

Build ~~projects~~ products around motivated individuals

Sustainable pace for ~~sponsors, users, team~~ all stakeholders

Value face-to-face communication

~~Regular~~ Continual reflection and tuning

@tastapod

Thanks for listening

Dan North

@tastapod

http://dannorth.net