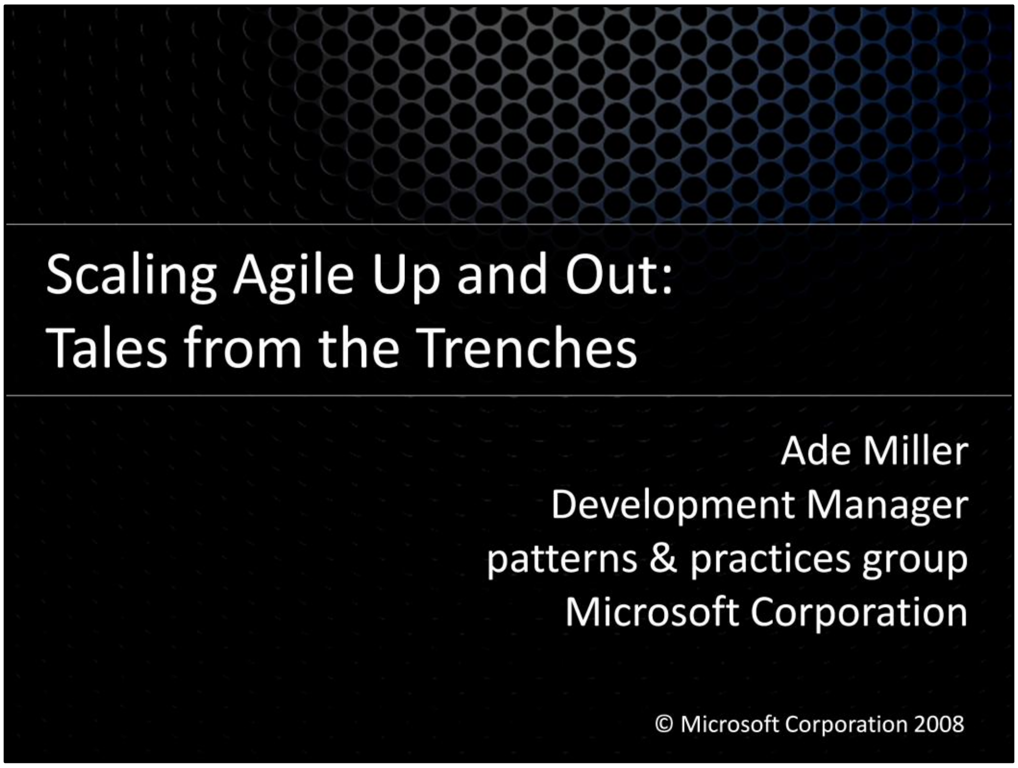


Scaling Agile Up and Out: Tales from the Trenches

Ade Miller
Development Manager
patterns & practices group
Microsoft Corporation

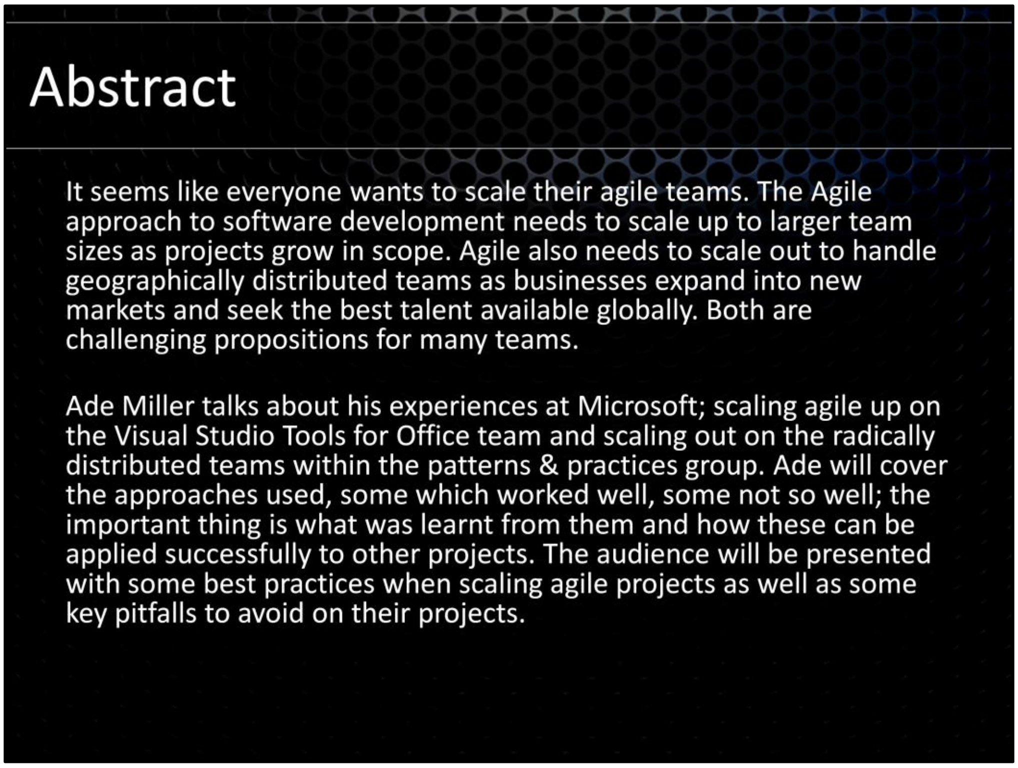
Ade Miller is currently the Development Manager for p&p
• Service Factory: Modeling Edition
• (EntLib, Prism, Unity)

Prior to that he was a developer and then Development Lead on the Visual Studio Tools for Office Team
• VSTO 2005
• VSTO 2008 Beta

He has been evangelizing agile and engineering best practices since walking in the door at Redmond five years ago

Microsoft is an extreme case… This is what happens with you turn the dial to 11 (to see what breaks)

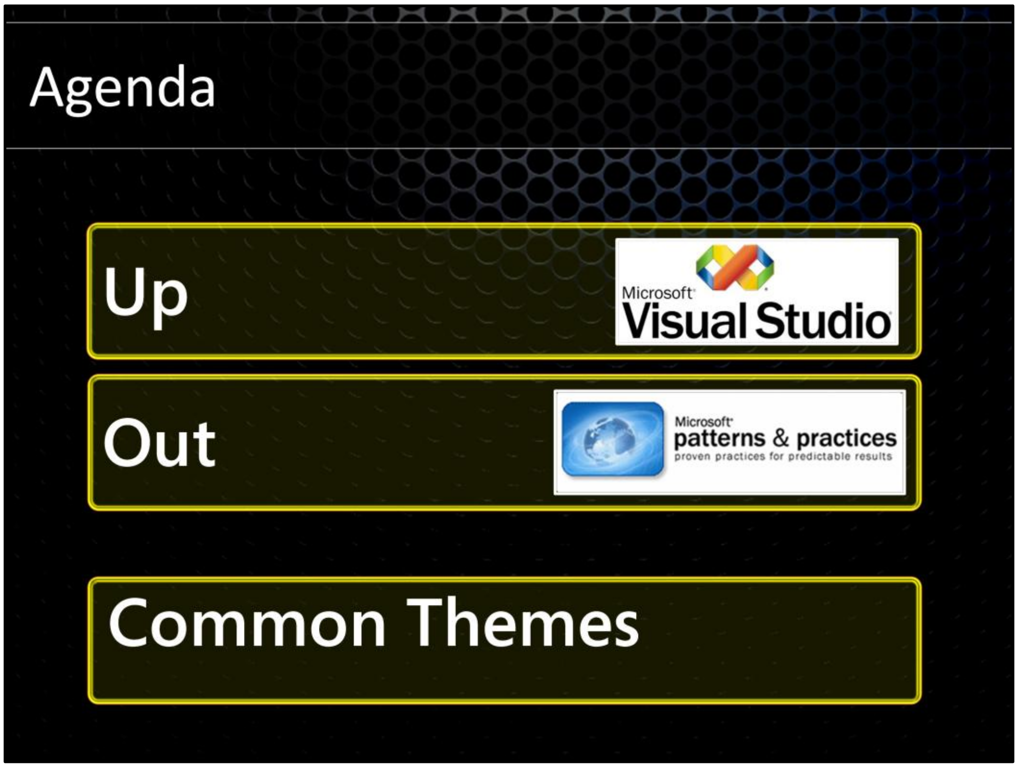
Helpful information for all scales of project



Abstract

It seems like everyone wants to scale their agile teams. The Agile approach to software development needs to scale up to larger team sizes as projects grow in scope. Agile also needs to scale out to handle geographically distributed teams as businesses expand into new markets and seek the best talent available globally. Both are challenging propositions for many teams.

Ade Miller talks about his experiences at Microsoft; scaling agile up on the Visual Studio Tools for Office team and scaling out on the radically distributed teams within the patterns & practices group. Ade will cover the approaches used, some which worked well, some not so well; the important thing is what was learnt from them and how these can be applied successfully to other projects. The audience will be presented with some best practices when scaling agile projects as well as some key pitfalls to avoid on their projects.



Agenda

Up

Out

Common Themes

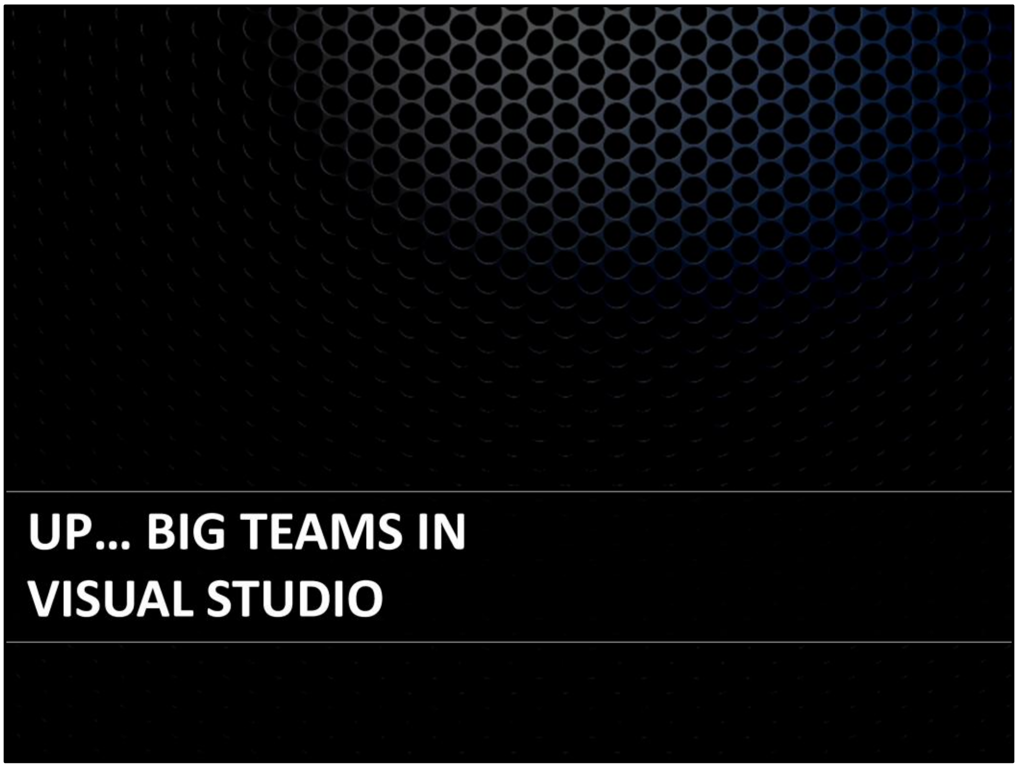
Microsoft Visual Studio
Microsoft patterns & practices
proven practices for predictable results

How many of you working on very large projects or very distributed teams today?

• If there's a big interest in one of these topics over the other we can focus on the relevant one.

A lot of what I'm going to cover today is about very large or very distributed projects

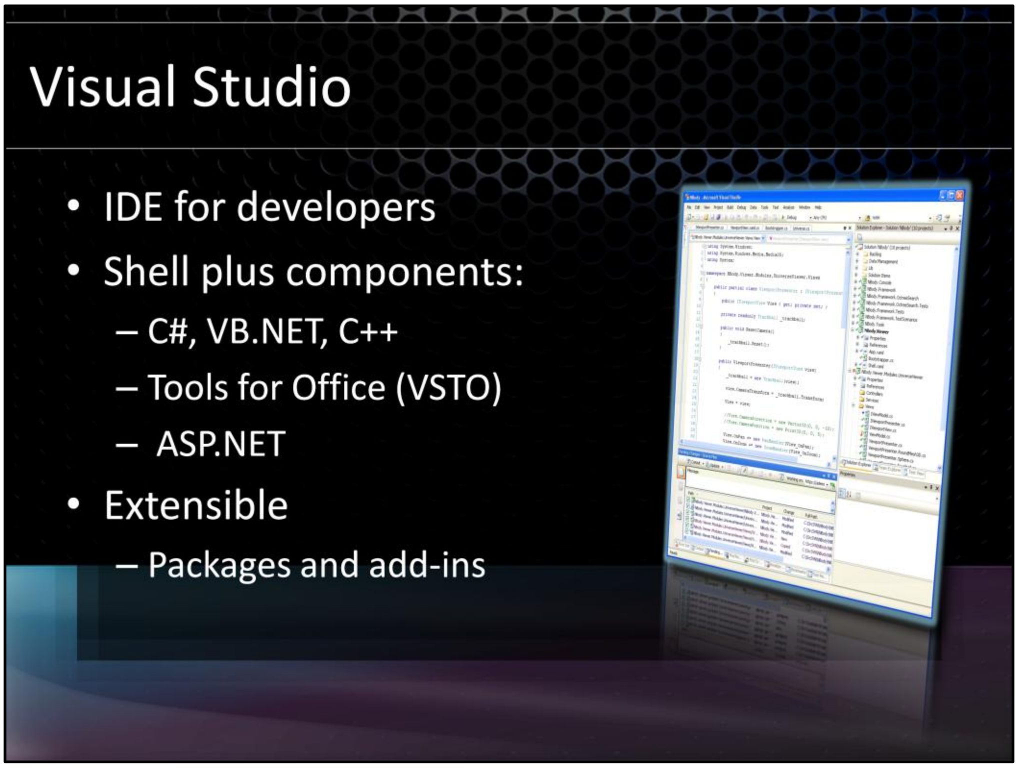
• This makes things break!
• Thinks that broke for us are probably pain points for you too
• Hopefully there are thing you can learn and apply to your projects



UP… BIG TEAMS IN VISUAL STUDIO

Overall (agile) process used by the Division

Things our Product unit, VSTO, did on a small team level



Visual Studio

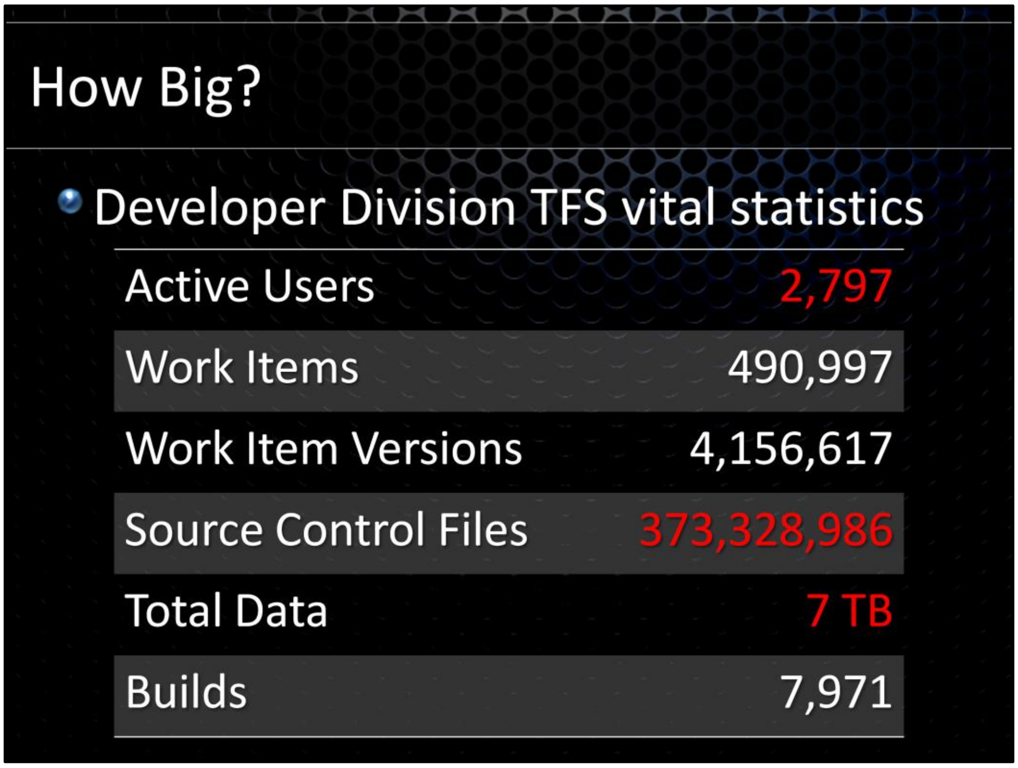
- IDE for developers
- Shell plus components:
  - C#, VB.NET, C++
  - Tools for Office (VSTO)
  - ASP.NET
- Extensible
  - Packages and add-ins

VS is a large complex product

- Been around a long time
- Large legacy code base; C++, COM, C#, ???

Just one part of what Developer Division does

- VSTS
- .NET runtime



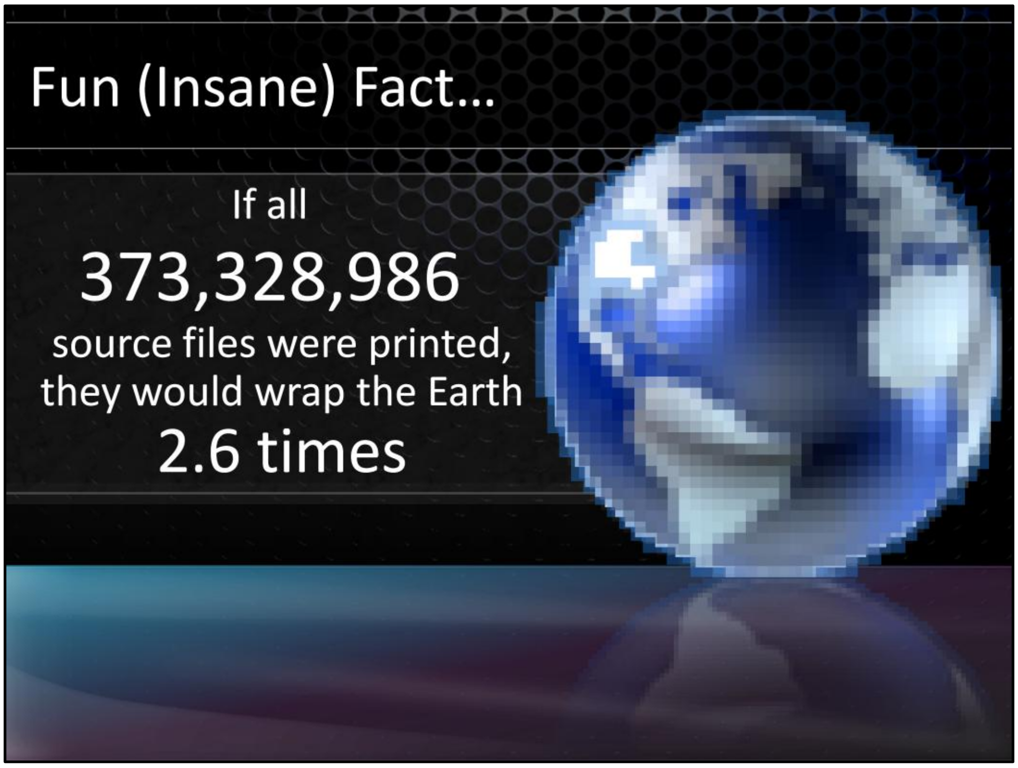
How Big?

- Developer Division TFS vital statistics

Active Users	2,797
Work Items	490,997
Work Item Versions	4,156,617
Source Control Files	373,328,986
Total Data	7 TB
Builds	7,971

The key takeaway here is that these a very big numbers

- Active users give you some idea of the number of people who commit to the tree on a frequent basis



Fun (Insane) Fact…

If all

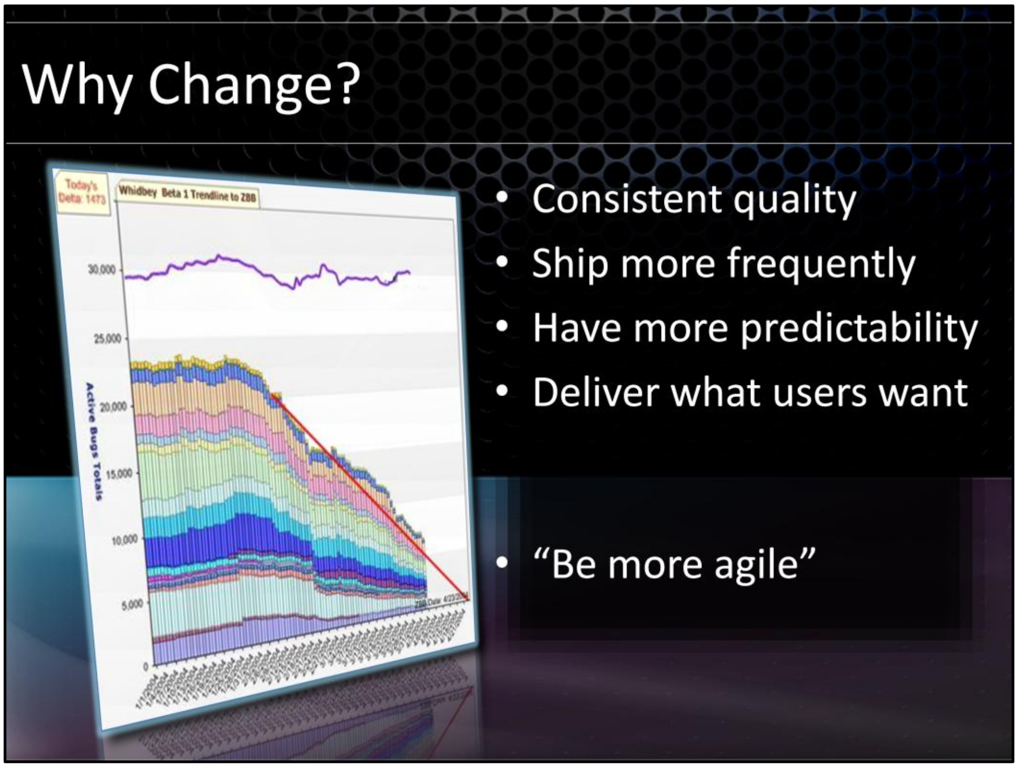
373,328,986

source files were printed, they would wrap the Earth

2.6 times

What’s the font size? That depends who you ask

- Marketing will say 72pt
- Legal will tell you 8pt



Why Change?

- Consistent quality
- Ship more frequently
- Have more predictability
- Deliver what users want

- "Be more agile"

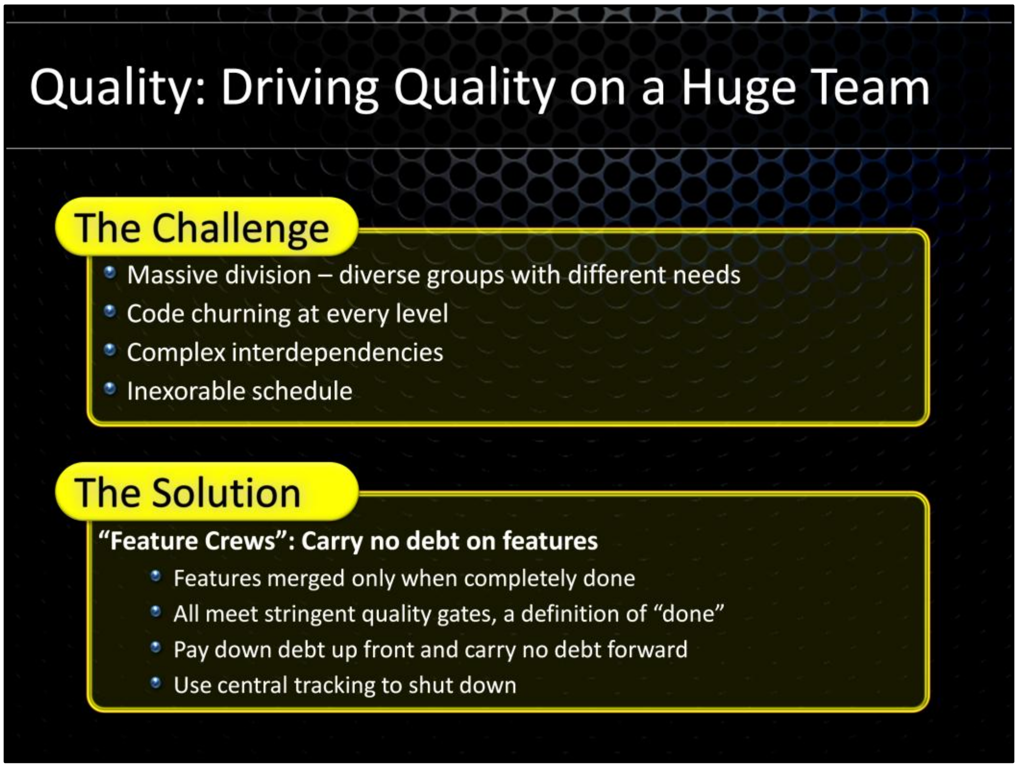
Today's Delta: 1473
Whidbey Beta 1 Trendline to ZBB
Active Bugs Totals
30,000
25,000
20,000
15,000
10,000
5,000
0

The previous release had a long bug tail

- I joined as a developer right where the red line starts… you can imagine I had a fun time.

These are in the order I'm going to talk about them in because it make a more cogent story

- This is NOT a priority order



Quality: Driving Quality on a Huge Team

The Challenge

- Massive division – diverse groups with different needs
- Code churning at every level
- Complex interdependencies
- Inexorable schedule

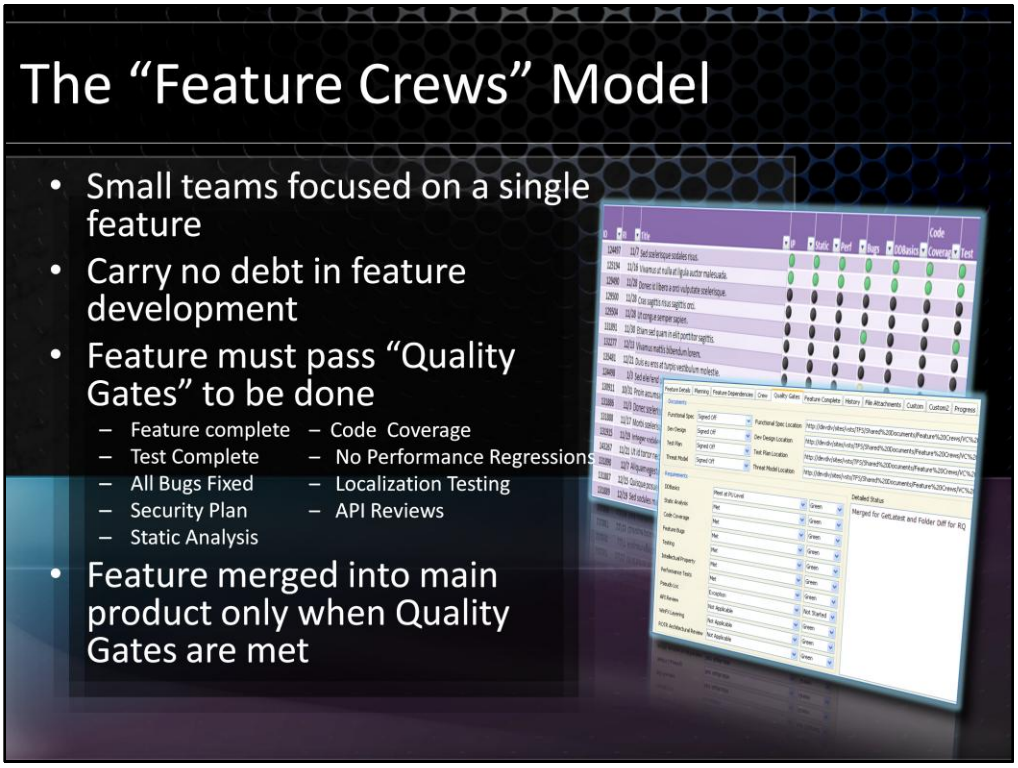
The Solution

“Feature Crews”: Carry no debt on features

- Features merged only when completely done
- All meet stringent quality gates, a definition of “done”
- Pay down debt up front and carry no debt forward
- Use central tracking to shut down

How do we get that many people working on a single codebase and not end up with a big mess?

- Isolation
- Only finished work ends up in the main branch



The “Feature Crews” Model

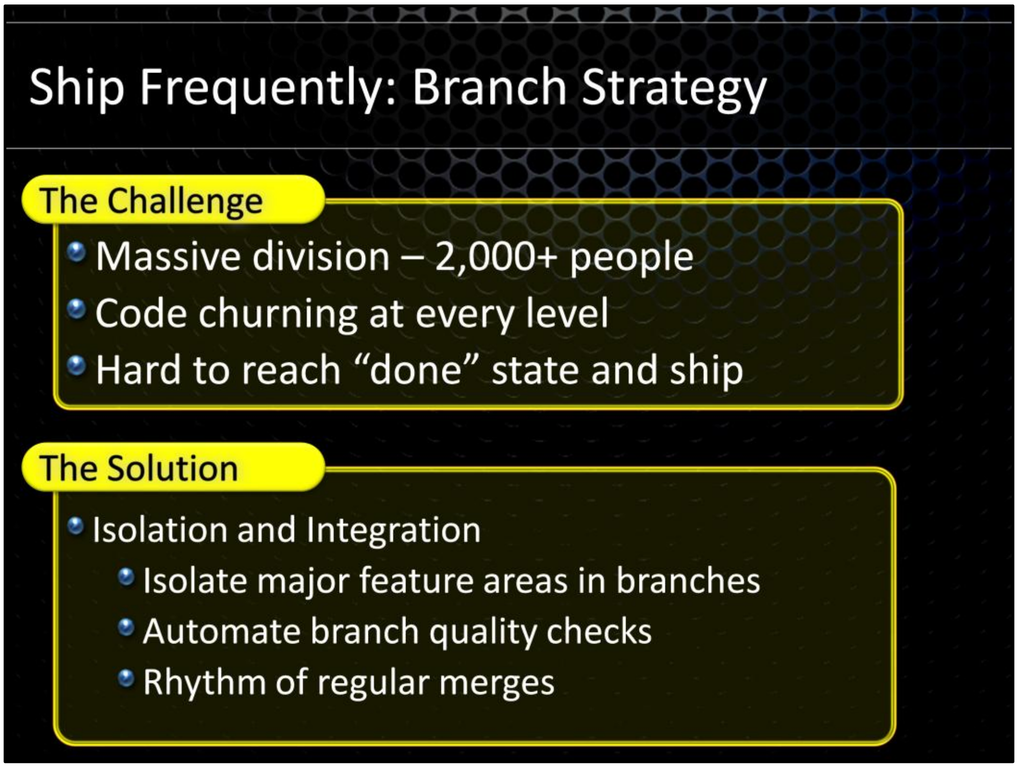
- Small teams focused on a single feature
- Carry no debt in feature development
- Feature must pass “Quality Gates” to be done
  - Feature complete
  - Test Complete
  - All Bugs Fixed
  - Security Plan
  - Static Analysis
  - Code Coverage
  - No Performance Regressions
  - Localization Testing
  - API Reviews
- Feature merged into main product only when Quality Gates are met

Quality gates are a contract between teams and the business

Feature crews are a framework that allow teams to work as they see fit

- Some agile some not
- Provided they meet the gates
- Provided they report their status

Enable agile behavior on a much larger scale without dictating to individual teams



Ship Frequently: Branch Strategy

The Challenge

- Massive division – 2,000+ people
- Code churning at every level
- Hard to reach "done" state and ship

The Solution

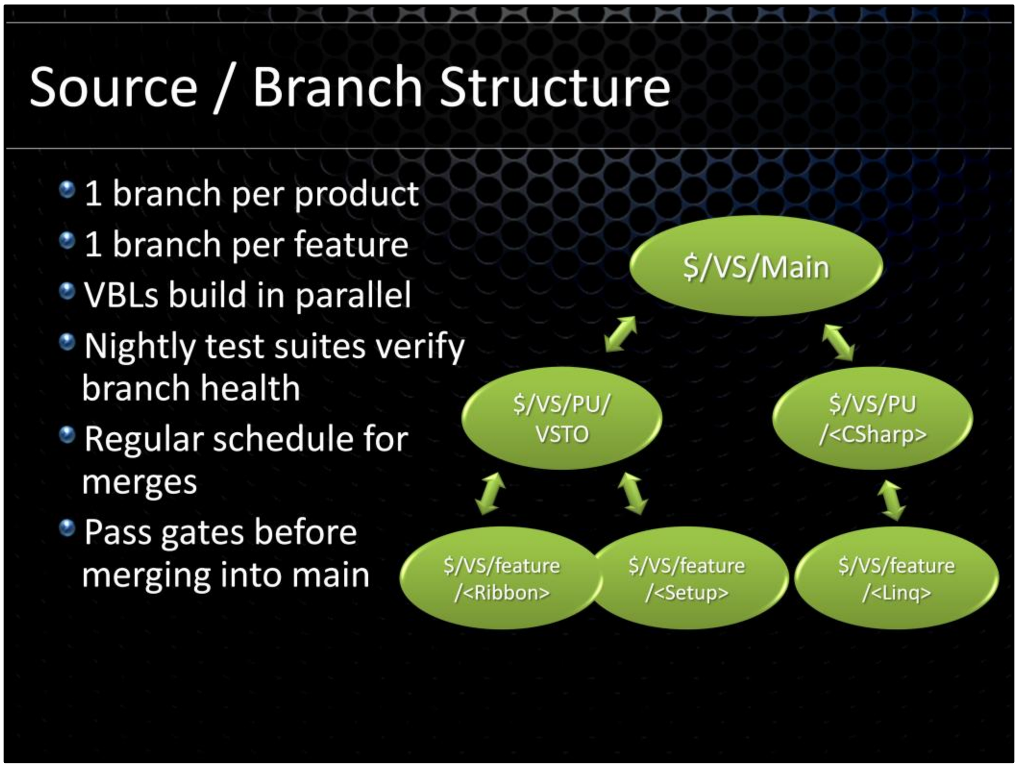
- Isolation and Integration
  - Isolate major feature areas in branches
  - Automate branch quality checks
  - Rhythm of regular merges

Example:

- 2000 developers each break the build once a year
- Everyone checks into a single branch
- The build is broken 3 times a day!

Where do these isolated crews work?

- In a separate (isolated) branch – separate code tree
- Regularly merge completed work into main branch



Source / Branch Structure

- 1 branch per product
- 1 branch per feature
- VBLs build in parallel
- Nightly test suites verify branch health
- Regular schedule for merges
- Pass gates before merging into main

$$\begin{tikzcd}
 & \text{\$/VS/Main} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{\$/VS/PU/VSTO} \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow] & & \text{\$/VS/PU/<CSharp>} \arrow[d, leftrightarrow] \\
\text{\$/VS/feature/<Ribbon>} & \text{\$/VS/feature/<Setup>} & \text{\$/VS/feature/<Linq>}
\end{tikzcd}$$

Product broken into Product Units (PUs)

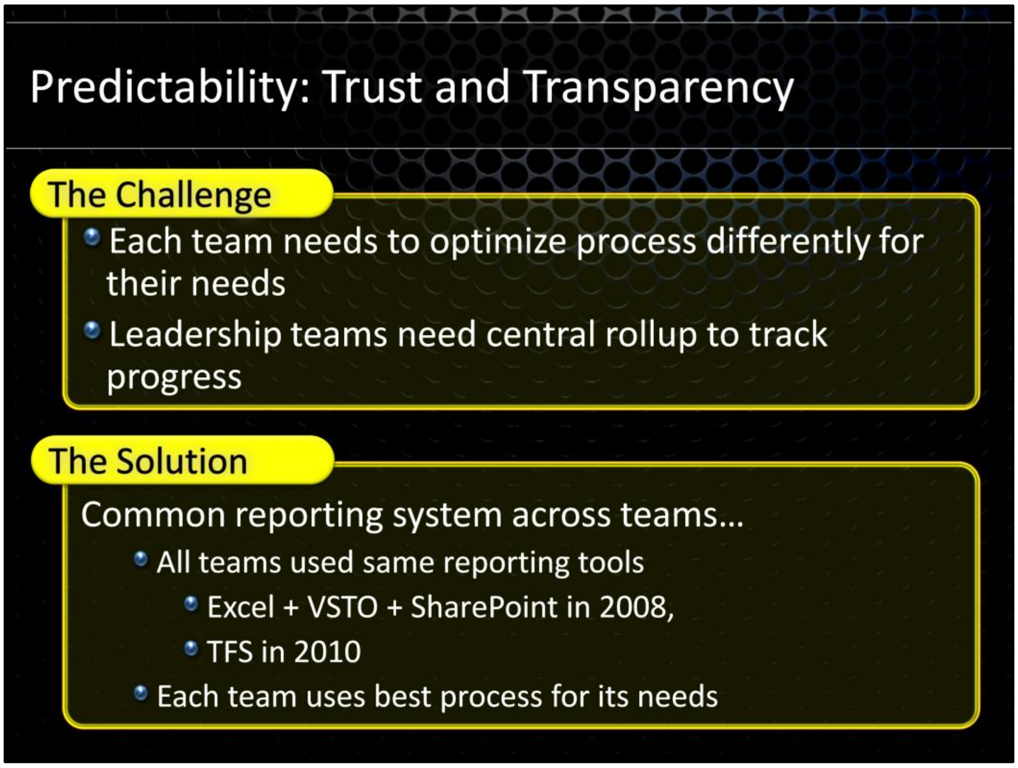
PUs break work down into Features

Code merges up the tree until it reaches the root. That's were we create product builds

Build everything every night and test

Merge up/down tree frequently

Quality gates must pass before merging



Predictability: Trust and Transparency

The Challenge

- Each team needs to optimize process differently for their needs
- Leadership teams need central rollup to track progress

The Solution

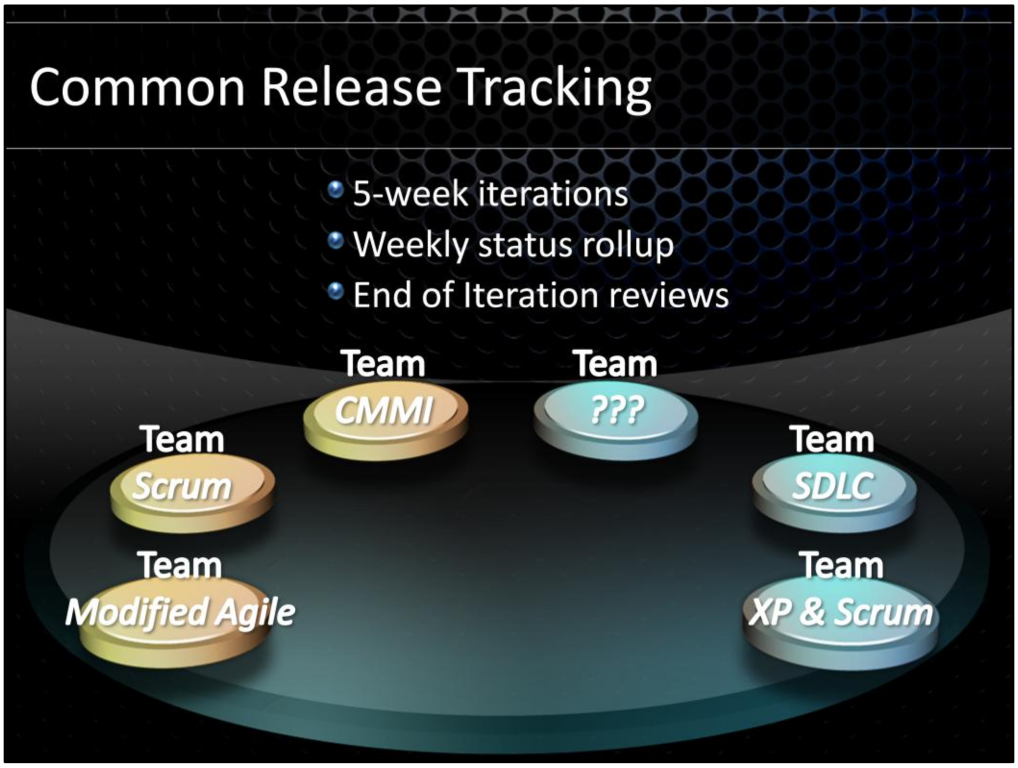
Common reporting system across teams…

- All teams used same reporting tools
  - Excel + VSTO + SharePoint in 2008,
  - TFS in 2010
- Each team uses best process for its needs

How do you surface progress to management?

Not enforce a process on 2000+ very different people and teams

A common reporting structure and tools



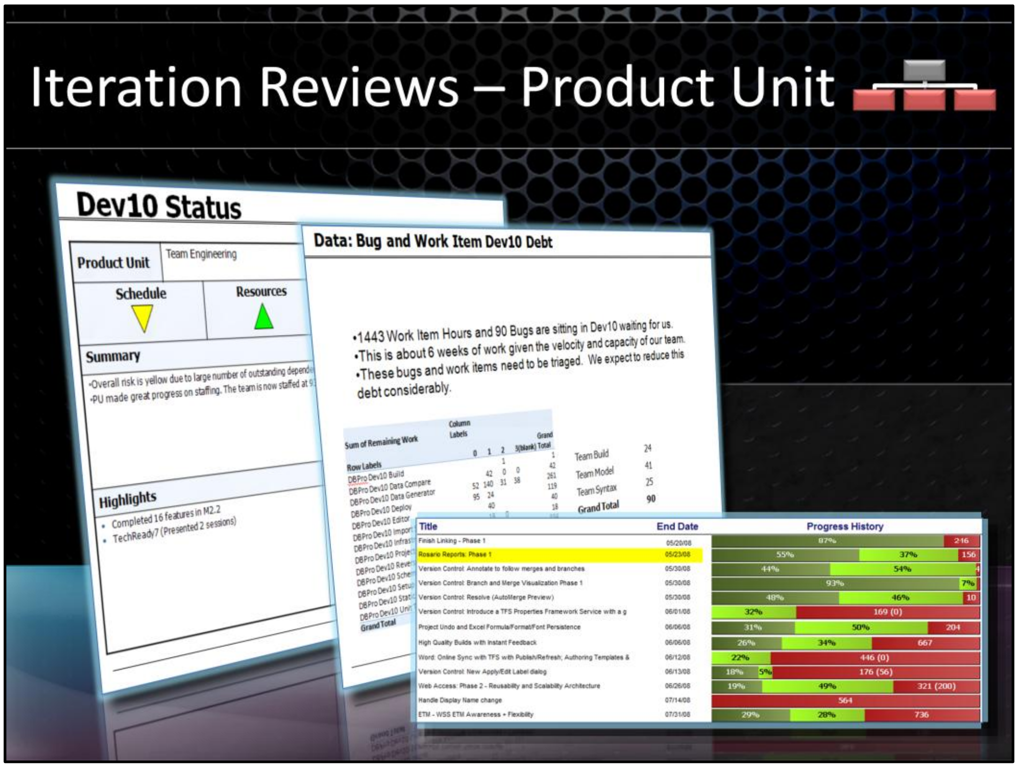
Common Release Tracking

- 5-week iterations
- Weekly status rollup
- End of Iteration reviews

Team CMMI
Team ???
Team Scrum
Team SDLC
Team Modified Agile
Team XP & Scrum

Teams use different processes but deliver on a common cadence

Common reporting structure and end of iteration reviews



Iteration Reviews – Product Unit

Dev10 Status

Product Unit	Team Engineering
Schedule ▽	**Resources** ▲

Summary

•Overall risk is yellow due to large number of outstanding depend
•PU made great progress on staffing. The team is now staffed at 9

Highlights

- Completed 16 features in M2.2
- TechReady7 (Presented 2 sessions)

Data: Bug and Work Item Dev10 Debt

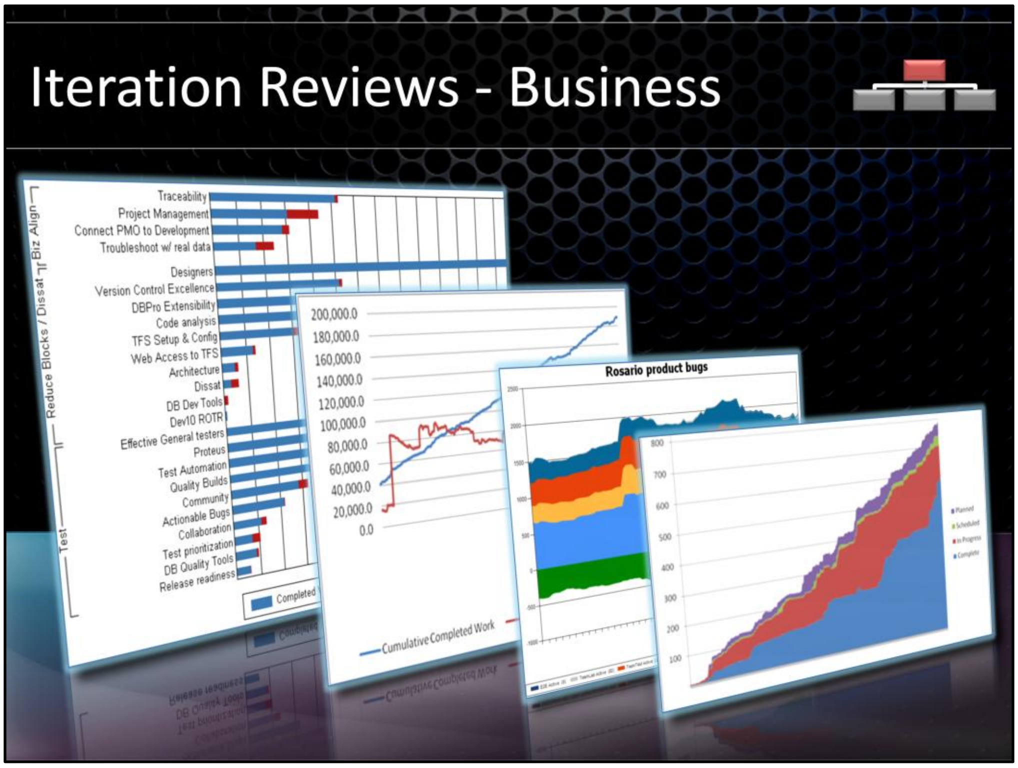
•1443 Work Item Hours and 90 Bugs are sitting in Dev10 waiting for us.
•This is about 6 weeks of work given the velocity and capacity of our team.
•These bugs and work items need to be triaged. We expect to reduce this debt considerably.

Sum of Remaining Work	Column Labels				
Row Labels	0	1	2	3(blank)	Grand Total
DBPro Dev10 Build			1		1
DBPro Dev10 Data Compare		42	0	0	42
DBPro Dev10 Data Generator	52	140	31	38	261
DBPro Dev10 Deploy	95	24			119
DBPro Dev10 Editor		40			40
DBPro Dev10 Impor					18
DBPro Dev10 Infras					
DBPro Dev10 Proje					
DBPro Dev10 Reve					
DBPro Dev10 Sche					
DBPro Dev10 Setu					
DBPro Dev10 Stat					
DBPro Dev10 Unit					
Grand Total					

Team Build	24
Team Model	41
Team Syntax	25
Grand Total	**90**

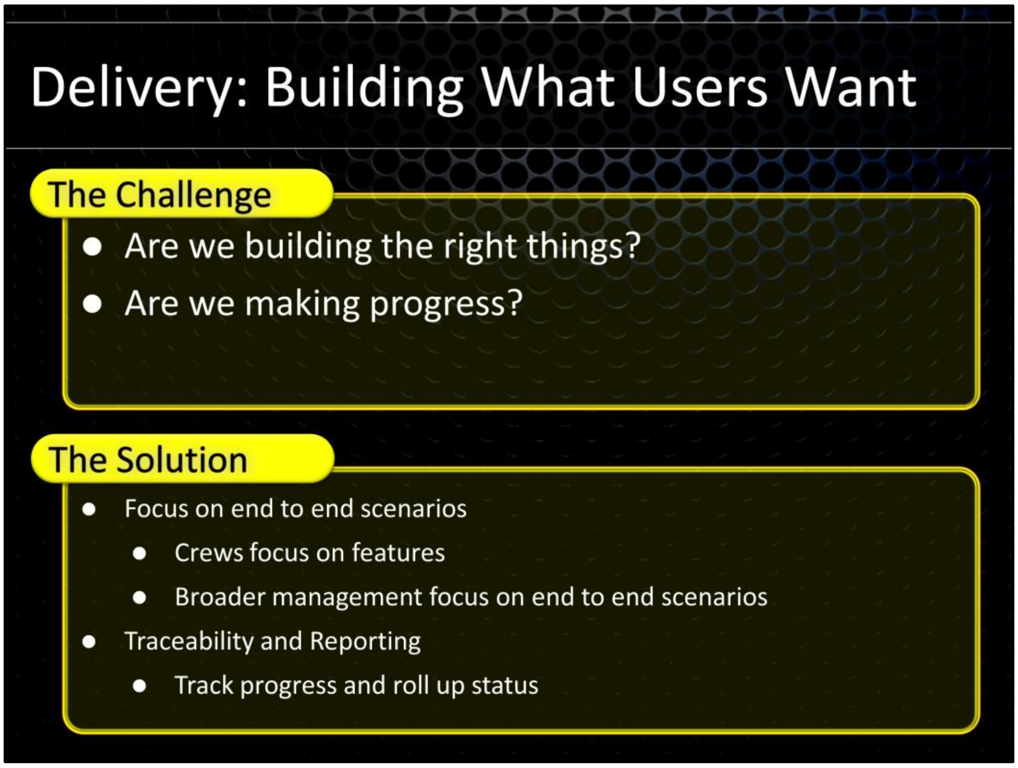
Title	End Date	Progress History		
Finish Linking - Phase 1	05/20/08	87%		246
Rosario Reports: Phase 1	05/23/08	55%	37%	156
Version Control: Annotate to follow merges and branches	05/30/08	44%	54%	4
Version Control: Branch and Merge Visualization Phase 1	05/30/08	93%	7%	
Version Control: Resolve (AutoMerge Preview)	05/30/08	48%	46%	10
Version Control: Introduce a TFS Properties Framework Service with a g	06/01/08		32%	169 (0)
Project Undo and Excel Formula/Format/Font Persistence	06/06/08	31%	50%	204
High Quality Builds with Instant Feedback	06/06/08	26%	34%	667
Word: Online Sync with TFS with Publish/Refresh; Authoring Templates &	06/12/08		22%	446 (0)
Version Control: New Apply/Edit Label dialog	06/13/08	18%	5%	176 (56)
Web Access: Phase 2 - Reusability and Scalability Architecture	06/26/08	19%	49%	321 (200)
Handle Display Name change	07/14/08			564
ETM - WSS ETM Awareness + Flexibility	07/31/08	29%	28%	736

Feature Crews report up to their Product Unit



Iteration Reviews - Business

Product Units report up to the business as a whole



Delivery: Building What Users Want

The Challenge

- Are we building the right things?
- Are we making progress?

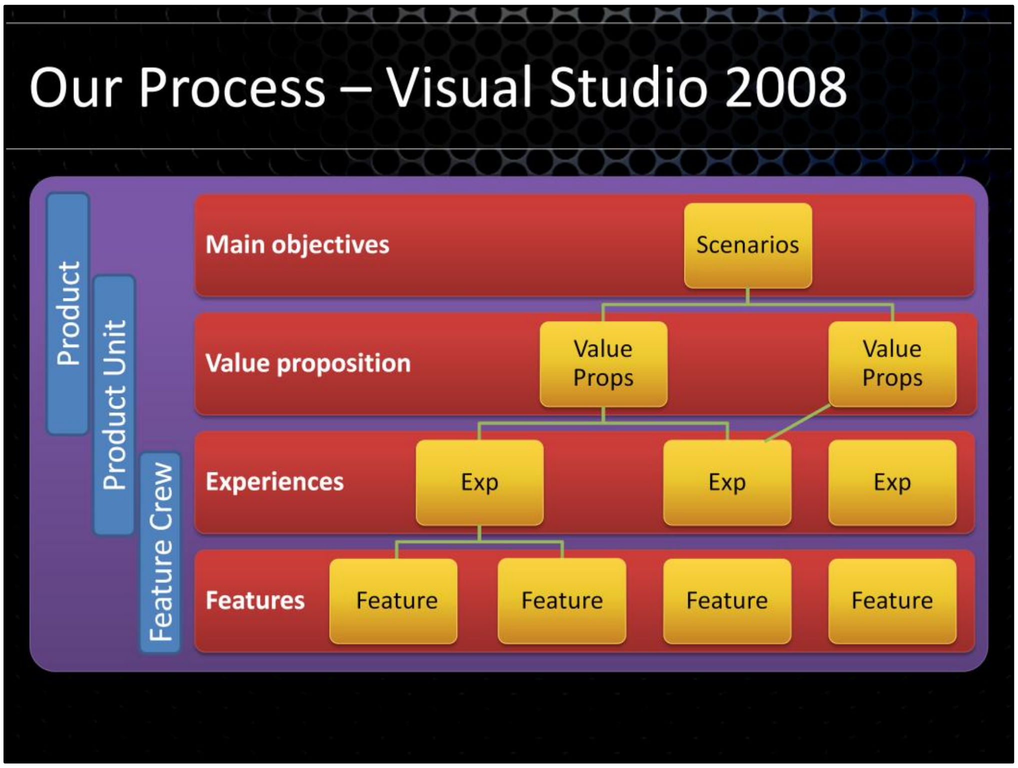
The Solution

- Focus on end to end scenarios
  - Crews focus on features
  - Broader management focus on end to end scenarios
- Traceability and Reporting
  - Track progress and roll up status

We still have to deliver the right thing!

If the feature crews are thinking about features then who thinks about the bigger picture?

End to end scenarios



Our Process – Visual Studio 2008

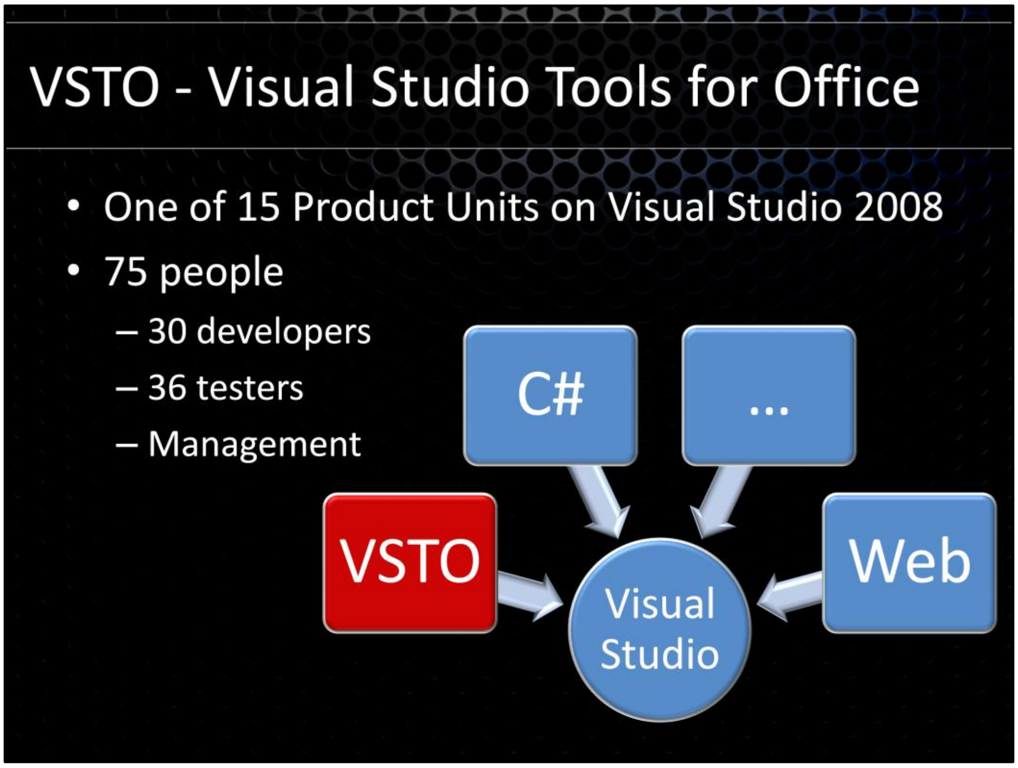
Crews understand and own their feature

Crews understand how it fits into the overall experience and value proposition

If crews want to make changes which impact the overall experience then they need to surface that at the Product Unit level

Product Units sync up across the whole product

DRIVING AGILE INSIDE BIG…
VISUAL STUDIO TOOLS FOR OFFICE



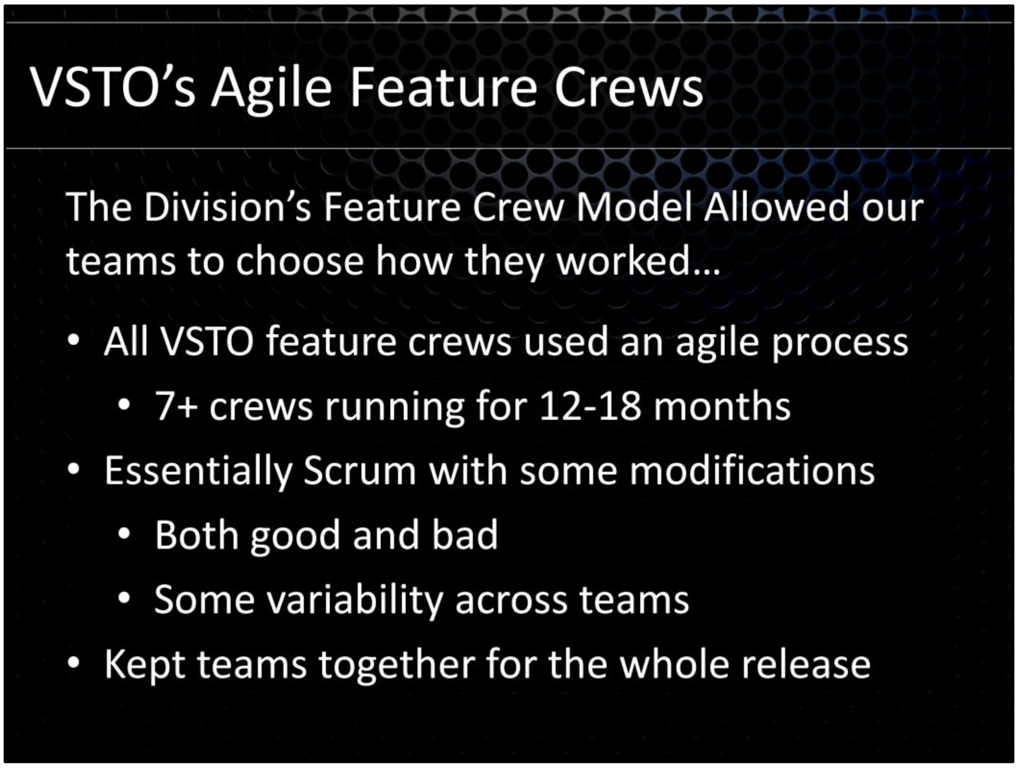
VSTO - Visual Studio Tools for Office

- One of 15 Product Units on Visual Studio 2008
- 75 people
  - 30 developers
  - 36 testers
  - Management

C#
...
VSTO
Web
Visual Studio

So how did the VSTO Product Unit leverage the Feature Crew process and run agile teams?

VSTO was just one of the Pus contributing to the Visual Studio 2008 product.



VSTO's Agile Feature Crews

The Division's Feature Crew Model Allowed our teams to choose how they worked…

- All VSTO feature crews used an agile process
  - 7+ crews running for 12-18 months
- Essentially Scrum with some modifications
  - Both good and bad
  - Some variability across teams
- Kept teams together for the whole release

Not going into great detail about how teams adopted Scrum at an individual level

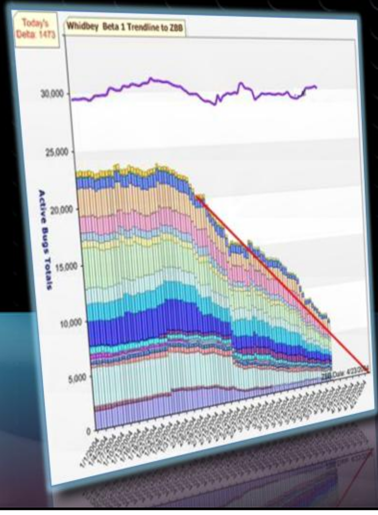
- Similar to many other stories you've heard before

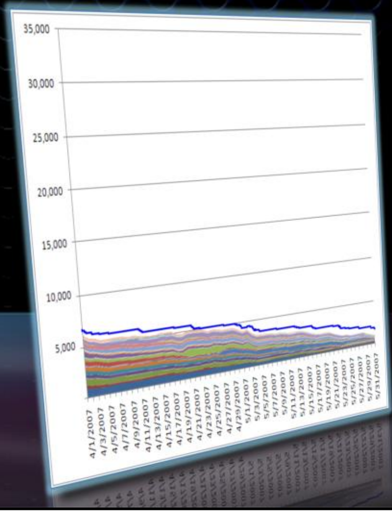
What's important is how we scaled

Did It Work?

VS 2005 Beta 1
Product Bugs only

VS 2008 Beta 1
<u>ALL</u> bug debt

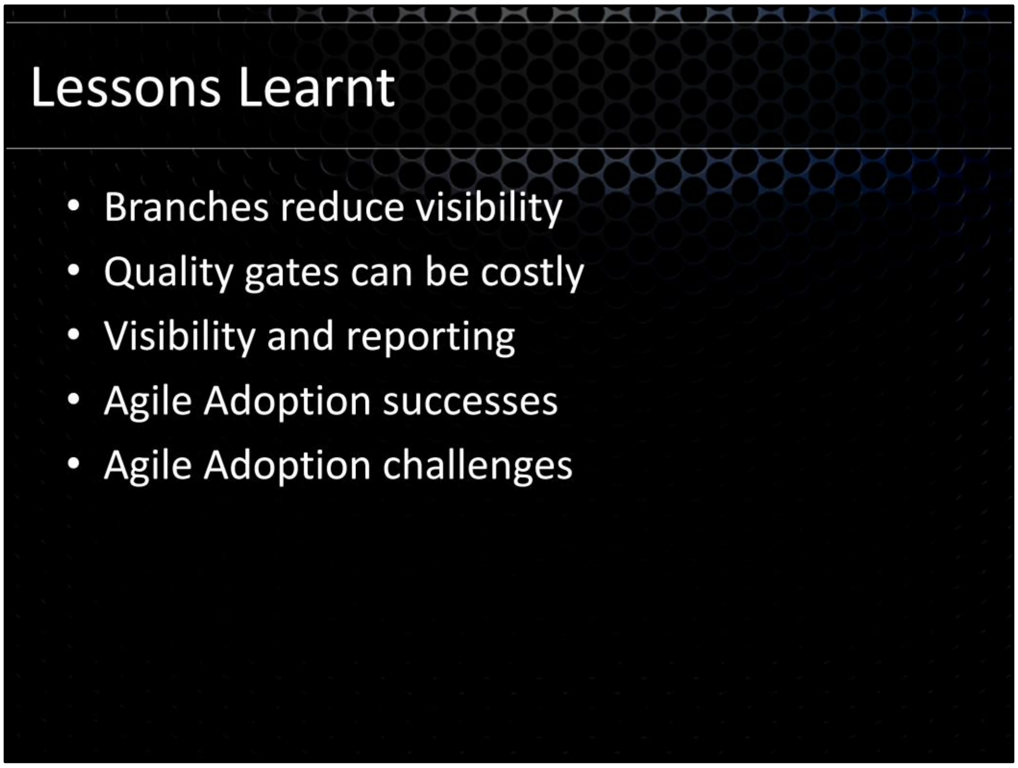




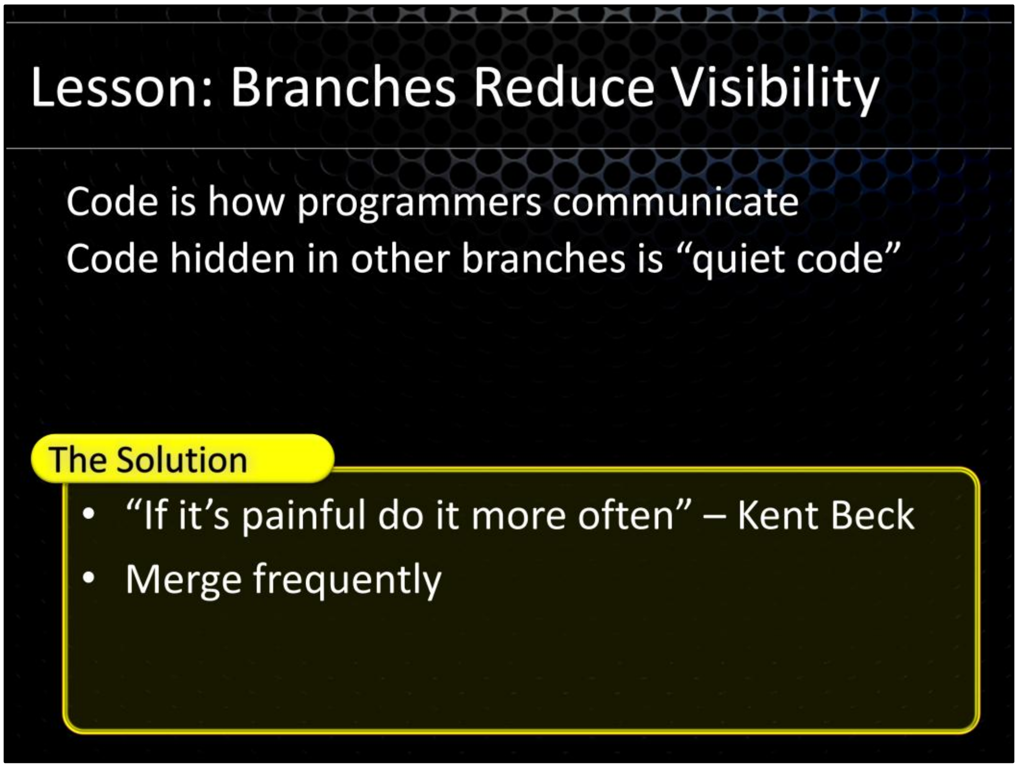
Biggest thing that I can show you

Improved time to market - 4 years down to 2

Improved visibility and more confidence in a predictable schedule


Lessons Learnt

- Branches reduce visibility
- Quality gates can be costly
- Visibility and reporting
- Agile Adoption successes
- Agile Adoption challenges



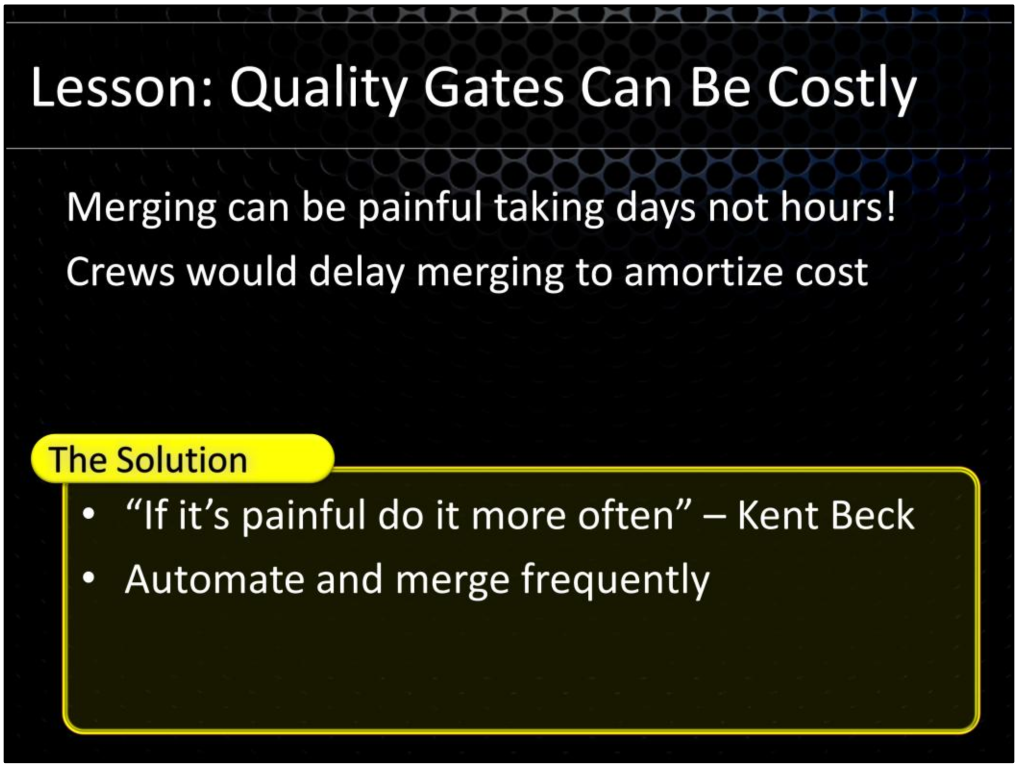
Lesson: Branches Reduce Visibility

Code is how programmers communicate
Code hidden in other branches is "quiet code"

The Solution

- "If it's painful do it more often" – Kent Beck
- Merge frequently

Senior Developers struggled with this the most
- They're the ones who expect to understand how features impact each other



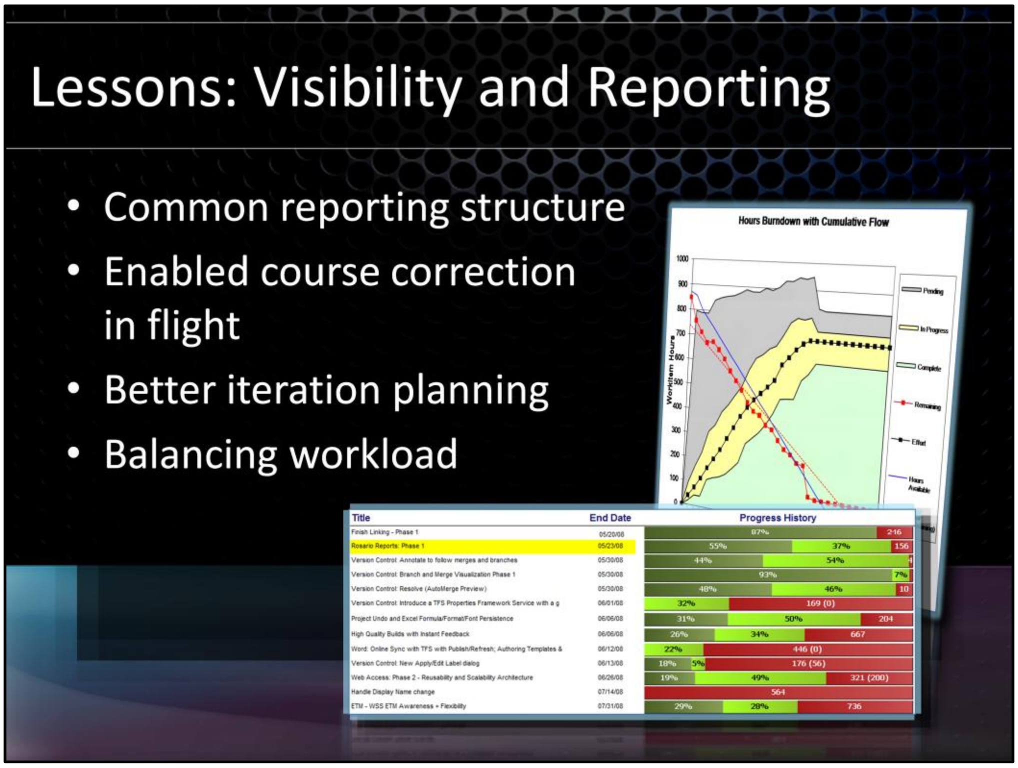
Lesson: Quality Gates Can Be Costly

Merging can be painful taking days not hours!

Crews would delay merging to amortize cost

The Solution

- "If it's painful do it more often" – Kent Beck
- Automate and merge frequently



Lessons: Visibility and Reporting

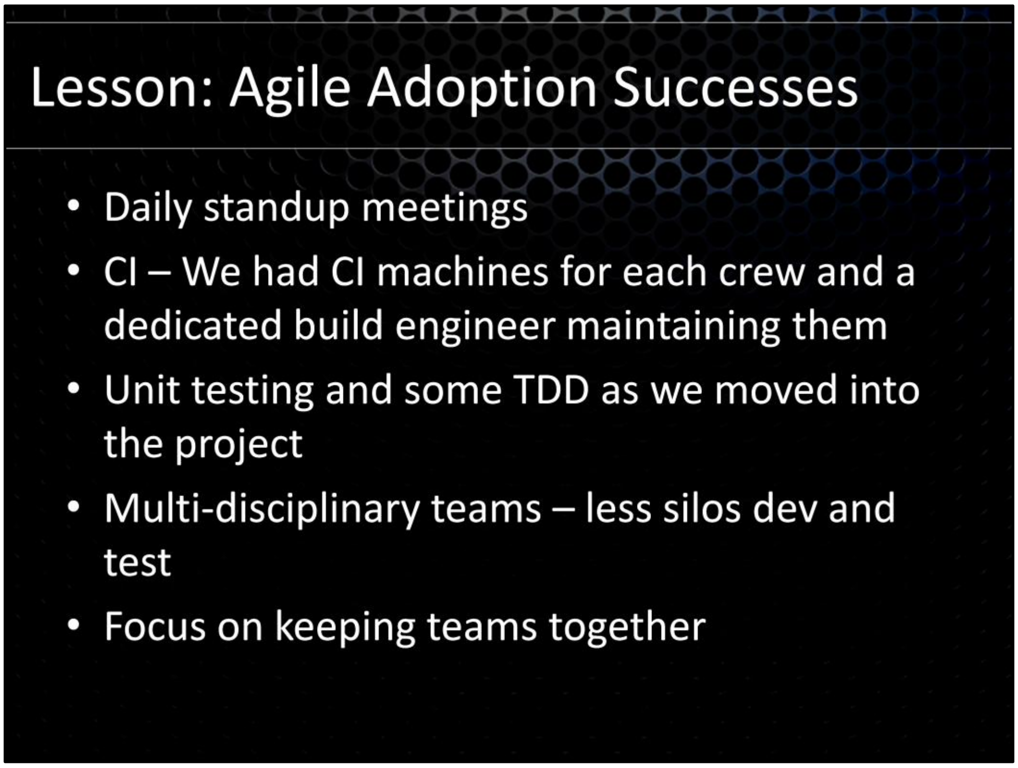
- Common reporting structure
- Enabled course correction in flight
- Better iteration planning
- Balancing workload

Title	End Date	Progress History
Finish Linking - Phase 1	05/20/08	87% 216
Rosario Reports: Phase 1	05/23/08	55% 37% 156
Version Control: Annotate to follow merges and branches	05/30/08	44% 54% 4
Version Control: Branch and Merge Visualization Phase 1	05/30/08	93% 7%
Version Control: Resolve (AutoMerge Preview)	05/30/08	48% 46% 10
Version Control: Introduce a TFS Properties Framework Service with a g	06/01/08	32% 169 (0)
Project Undo and Excel Formula/Format/Font Persistence	06/06/08	31% 50% 204
High Quality Builds with Instant Feedback	06/06/08	26% 34% 667
Word: Online Sync with TFS with Publish/Refresh; Authoring Templates &	06/12/08	22% 446 (0)
Version Control: New Apply/Edit Label dialog	06/13/08	18% 5% 176 (56)
Web Access: Phase 2 - Reusability and Scalability Architecture	06/26/08	19% 49% 321 (200)
Handle Display Name change	07/14/08	564
ETM - WSS ETM Awareness + Flexibility	07/31/08	29% 28% 736

Improved visibility within VSTO improved our ability to balance workload

- ensured we shipped the right things
- ensured we put effort in the right places

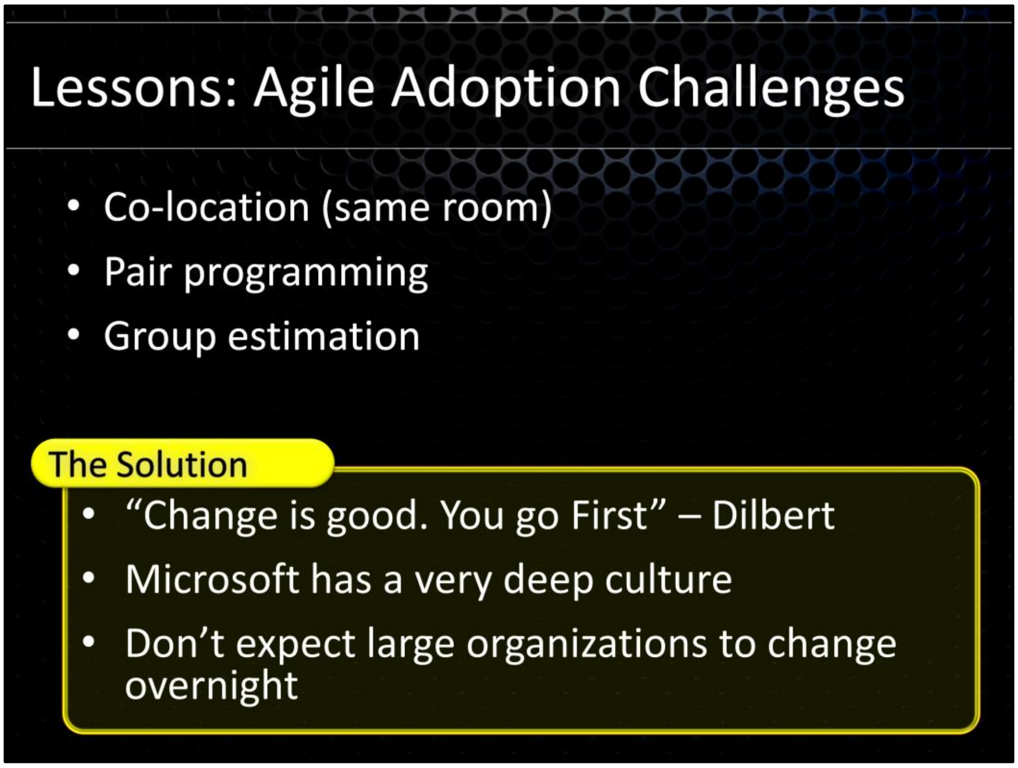
Similarly at the PU/Product level… Common reporting structure improved visibility



Lesson: Agile Adoption Successes

- Daily standup meetings
- CI – We had CI machines for each crew and a dedicated build engineer maintaining them
- Unit testing and some TDD as we moved into the project
- Multi-disciplinary teams – less silos dev and test
- Focus on keeping teams together

If all I've ever achieved is getting people who sit in their own offices every day to do a daily standup then that's progress!

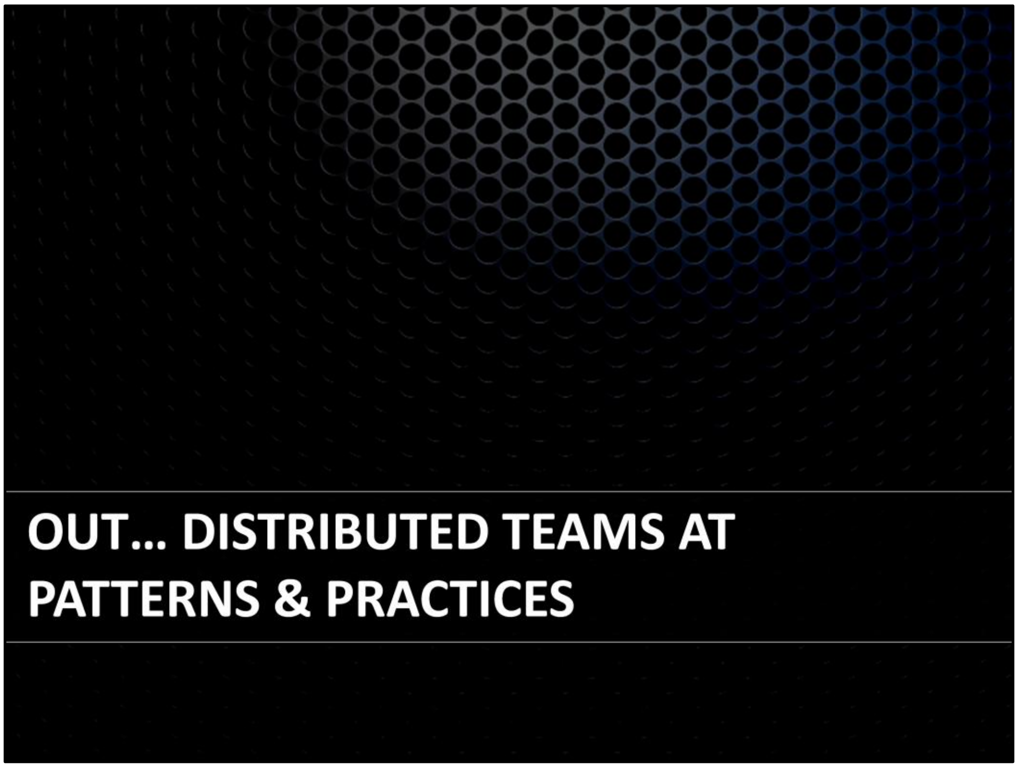


Lessons: Agile Adoption Challenges

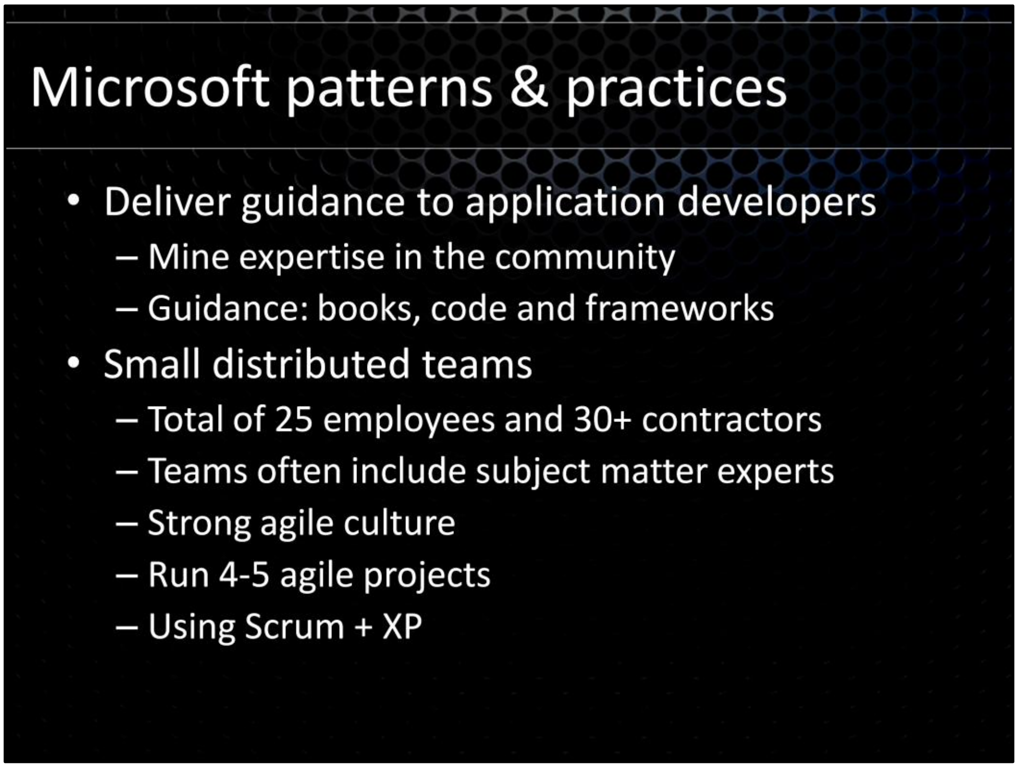
- Co-location (same room)
- Pair programming
- Group estimation

The Solution

- “Change is good. You go First” – Dilbert
- Microsoft has a very deep culture
- Don’t expect large organizations to change overnight



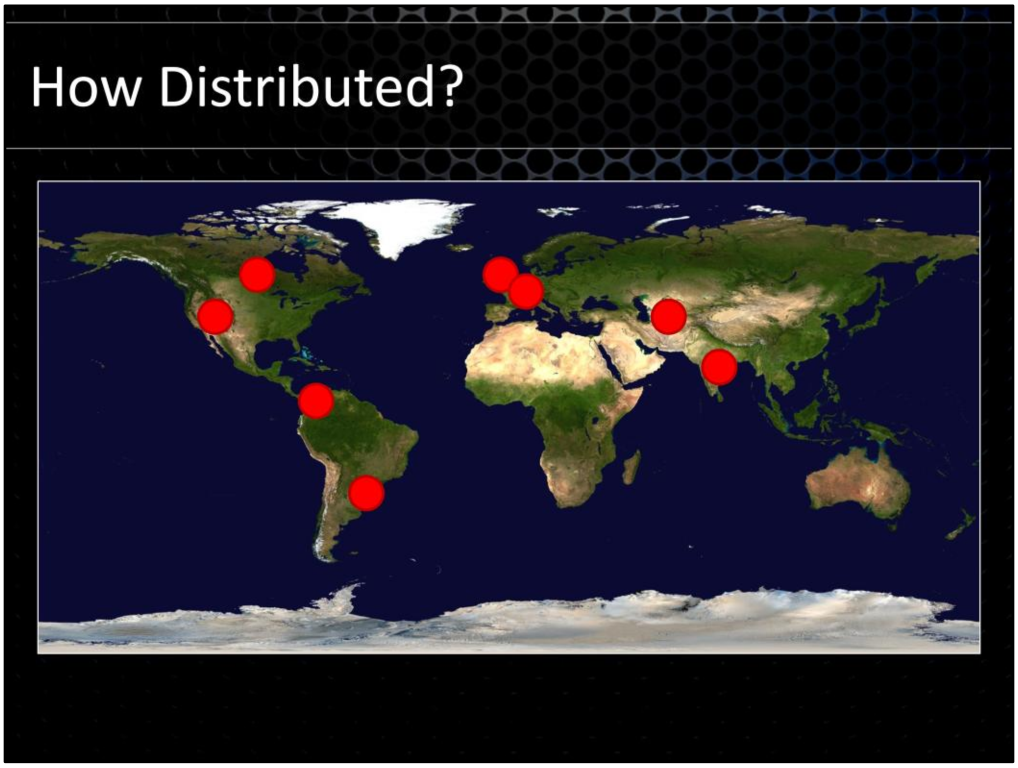
OUT… DISTRIBUTED TEAMS AT PATTERNS & PRACTICES



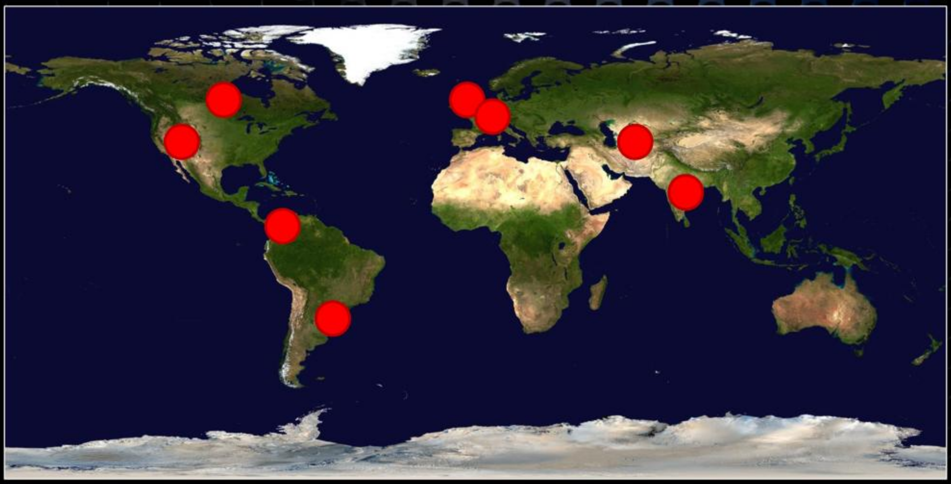
Microsoft patterns & practices

- Deliver guidance to application developers
  - Mine expertise in the community
  - Guidance: books, code and frameworks
- Small distributed teams
  - Total of 25 employees and 30+ contractors
  - Teams often include subject matter experts
  - Strong agile culture
  - Run 4-5 agile projects
  - Using Scrum + XP

http://msdn.microsoft.com/practices



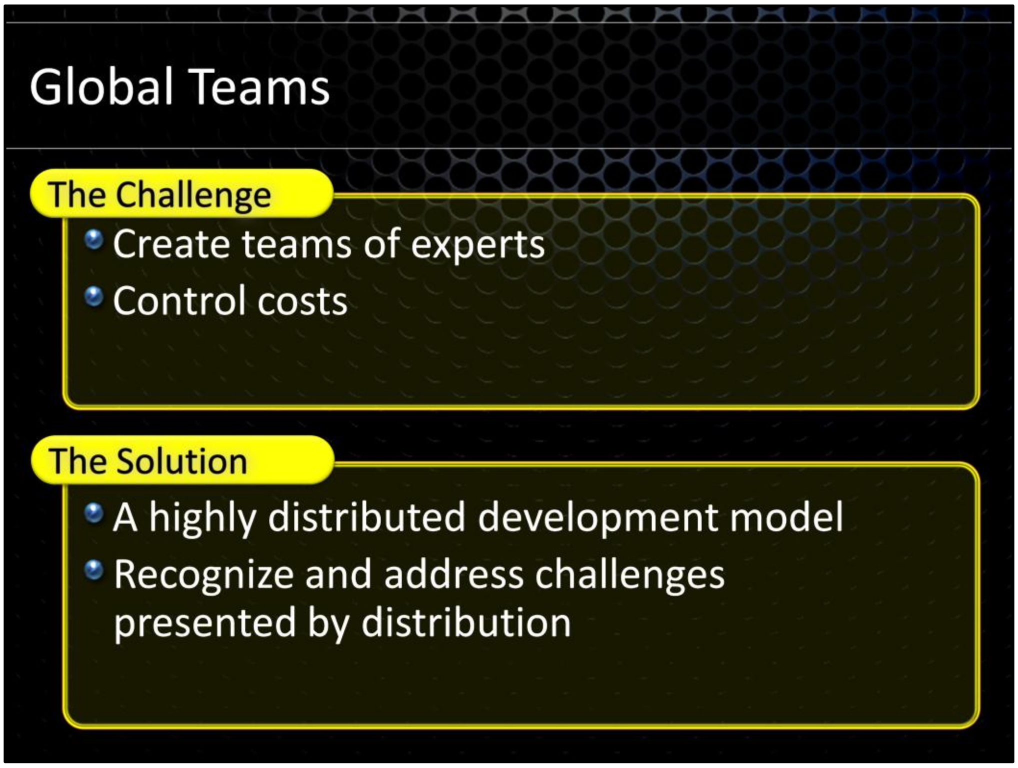
How Distributed?



For a team of 60 or so people we seem to be everywhere!
- Most (but not all) of the Microsoft FTEs work in Redmond

p&p have been doing distributed agile development for about 5 years

p&p is not typical of *all* teams at Microsoft



Global Teams

The Challenge

- Create teams of experts
- Control costs

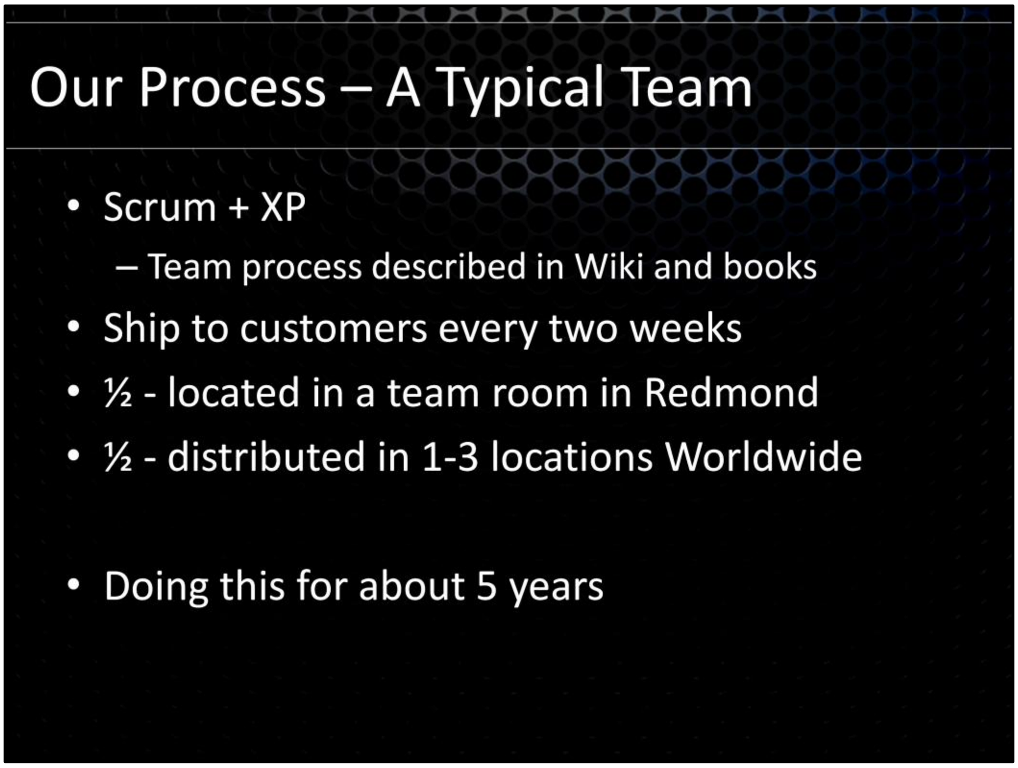
The Solution

- A highly distributed development model
- Recognize and address challenges presented by distribution

So you might start thinking “Duh, any agile team would do these things!”

You need to do these things on any team but failing to do them on a distributed team is a lot worse!!

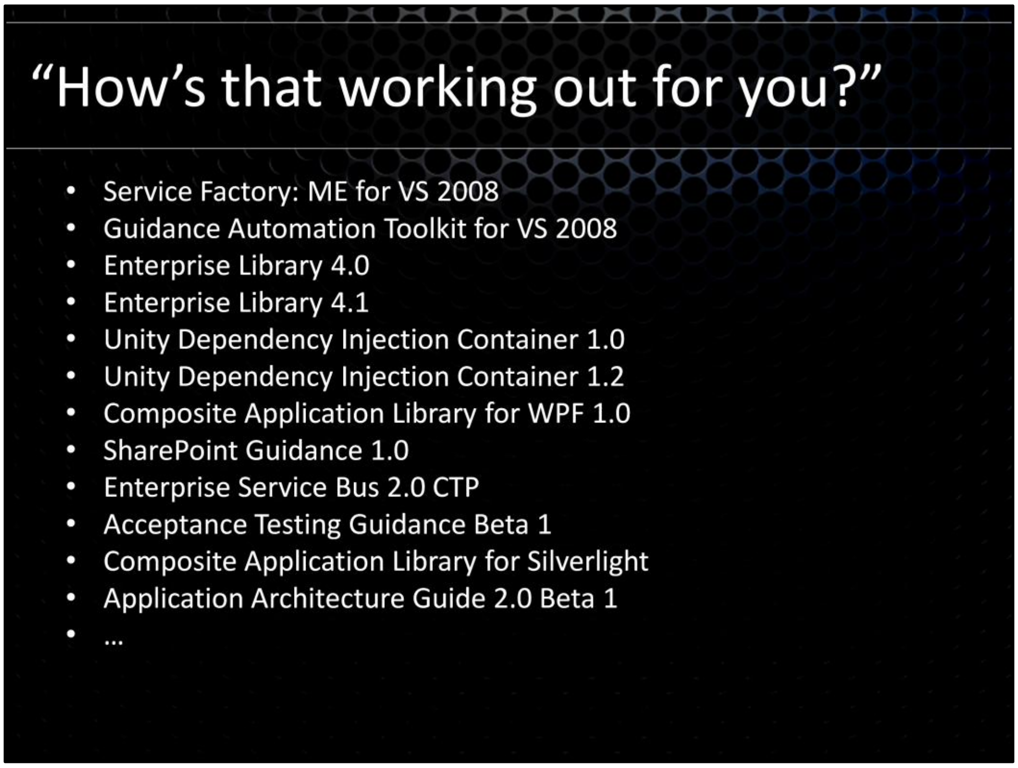
Distributed teams are inherently more dysfunctional… it amplifies any existing problems.



Our Process – A Typical Team

- Scrum + XP
  - Team process described in Wiki and books
- Ship to customers every two weeks
- ½ - located in a team room in Redmond
- ½ - distributed in 1-3 locations Worldwide

- Doing this for about 5 years



"How's that working out for you?"

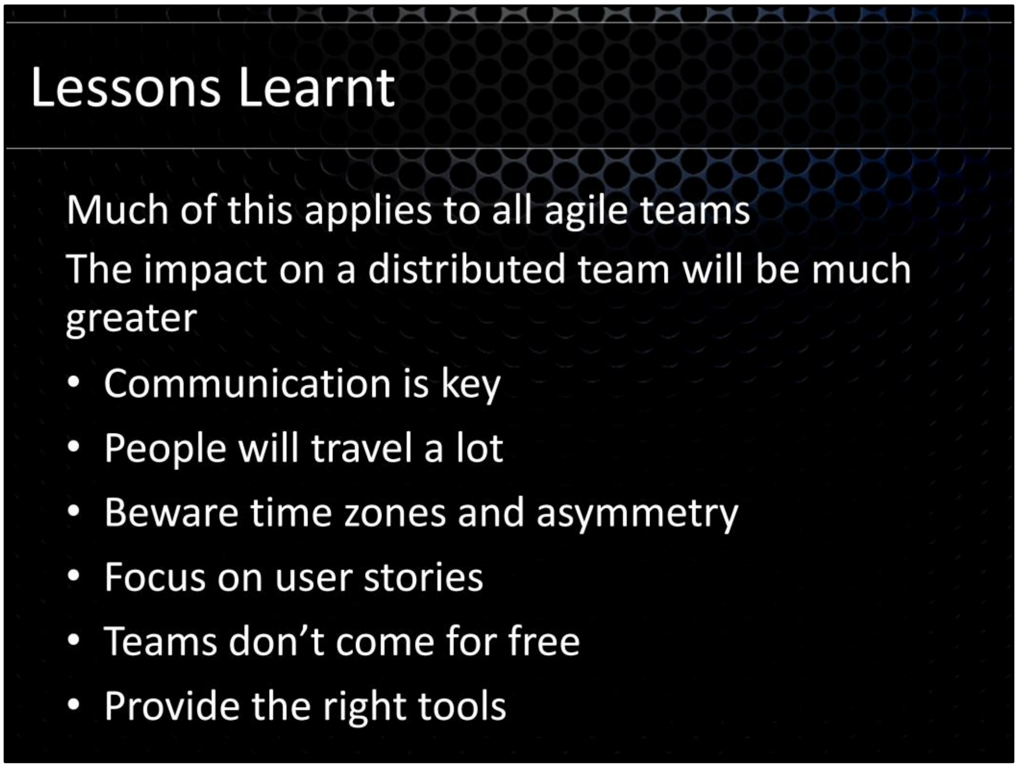
- Service Factory: ME for VS 2008
- Guidance Automation Toolkit for VS 2008
- Enterprise Library 4.0
- Enterprise Library 4.1
- Unity Dependency Injection Container 1.0
- Unity Dependency Injection Container 1.2
- Composite Application Library for WPF 1.0
- SharePoint Guidance 1.0
- Enterprise Service Bus 2.0 CTP
- Acceptance Testing Guidance Beta 1
- Composite Application Library for Silverlight
- Application Architecture Guide 2.0 Beta 1
- …

This is a question Kent Beck likes to ask…

<CLICK>

These are the major deliverables from patterns & practices in 2008 (so far)
- There's more but I ran out of space!

Our customers generally like our stuff… That's not that we can't improve



Lessons Learnt

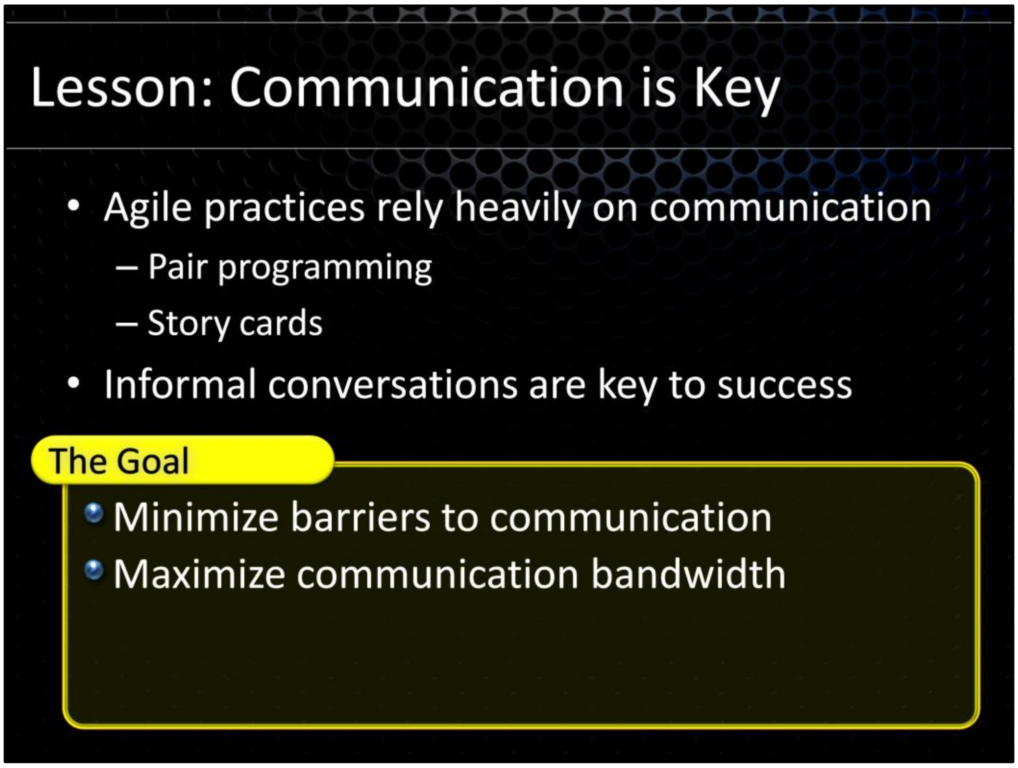
Much of this applies to all agile teams

The impact on a distributed team will be much greater

- Communication is key
- People will travel a lot
- Beware time zones and asymmetry
- Focus on user stories
- Teams don't come for free
- Provide the right tools

As I said before. All agile teams should do these things. Distribution makes this harder!

<CLICK>



Lesson: Communication is Key

- Agile practices rely heavily on communication
  - Pair programming
  - Story cards
- Informal conversations are key to success

The Goal
- Minimize barriers to communication
- Maximize communication bandwidth

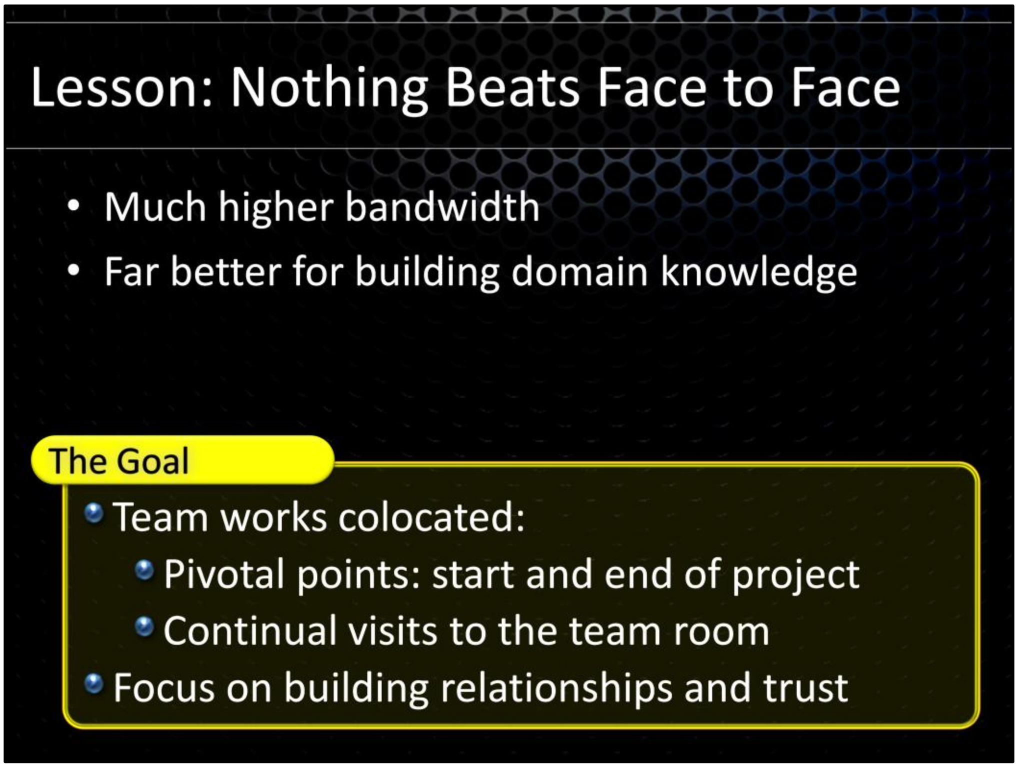
Agile is a bunch of overlapping practices that rely on communication in one way or another.

- Remove one and it weakens the other.

Example… Pair programming. Hard with two developers in different (non-overlapping) time zones.

Example… Story cards are really placeholders for conversations.

The goal… minimize the difference between people in the same room and people elsewhere.



Lesson: Nothing Beats Face to Face

- Much higher bandwidth
- Far better for building domain knowledge

The Goal

- Team works colocated:
  - Pivotal points: start and end of project
  - Continual visits to the team room
- Focus on building relationships and trust

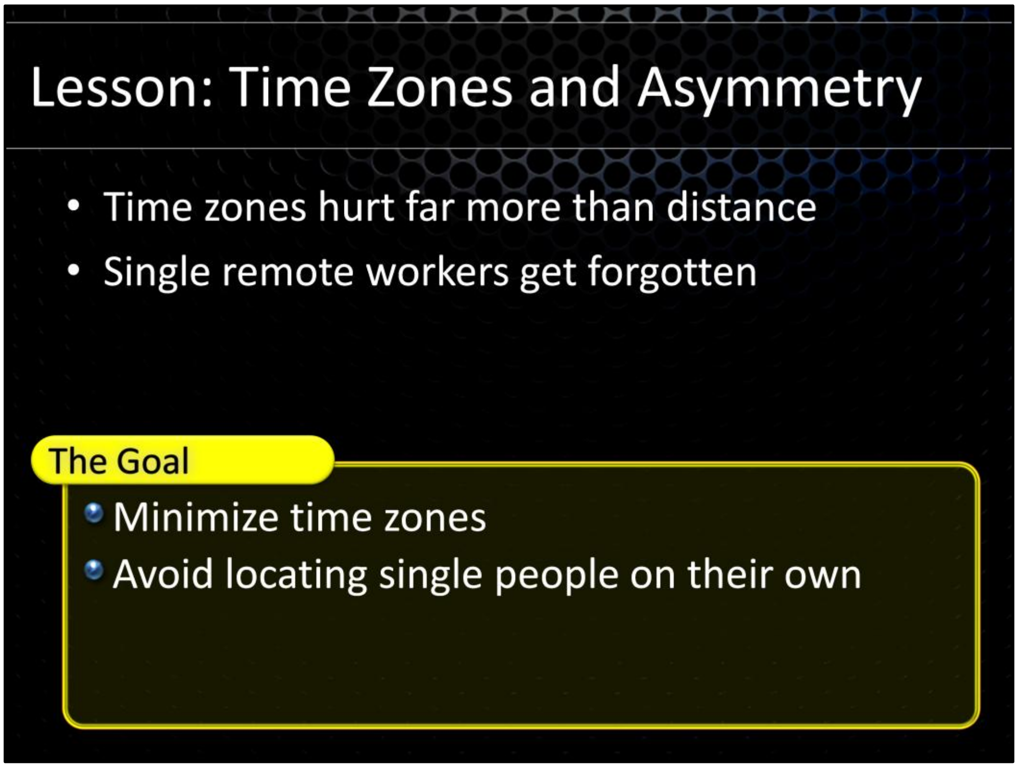
At pivotal points during the project…

- The beginning… requirements, release planning, design, metaphor
- Key points… releases, customer visits
- The end… retrospective, ship party

Really all about building *trust* and inter-personal relationships

So make sure it includes some social aspects!

- Time for hanging out over coffee
- Team lunch
- Rock band party



Lesson: Time Zones and Asymmetry

- Time zones hurt far more than distance
- Single remote workers get forgotten

The Goal

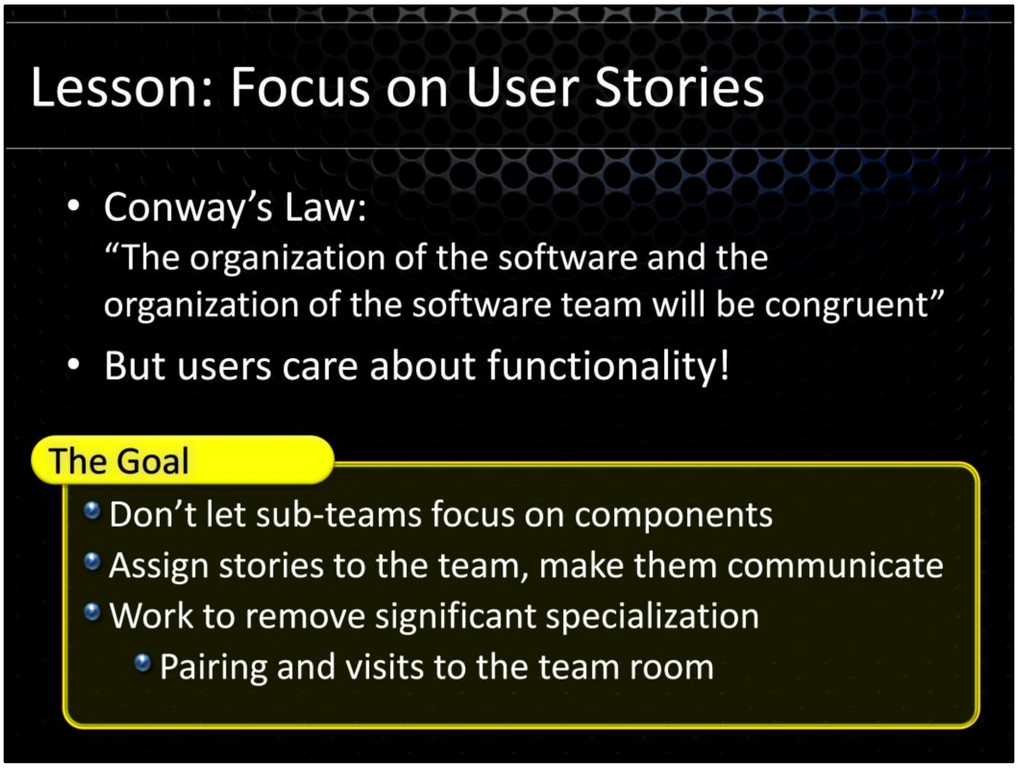
- Minimize time zones
- Avoid locating single people on their own

Time zones just make things harder

- Make the best use of overlapping time
- Use a team room representative (more on that in a moment)

Asymmetric distributions are hard too

- Best avoided
- Team buddy
- Enforce symmetry



Lesson: Focus on User Stories

- Conway's Law:
  "The organization of the software and the organization of the software team will be congruent"
- But users care about functionality!

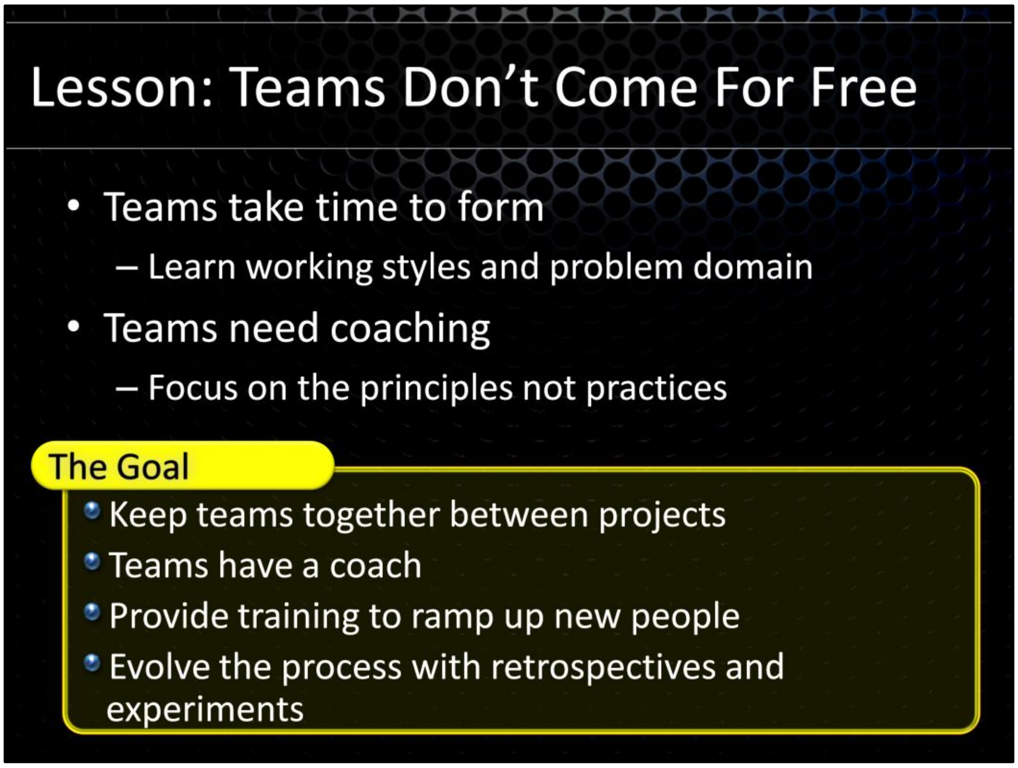
The Goal
- Don't let sub-teams focus on components
- Assign stories to the team, make them communicate
- Work to remove significant specialization
  - Pairing and visits to the team room

Users care about end to end stories not individual components

Conway's Law will tempt you to…
- Assign work by location
- Specialize sub-teams accordingly

Actually you should do the opposite
- Assign stories to the team
- Work to remove significant specialization

WE SHIP SOURCE CODE – This makes the effects of Conway's Law far more visible



Lesson: Teams Don't Come For Free

- Teams take time to form
  - Learn working styles and problem domain
- Teams need coaching
  - Focus on the principles not practices

The Goal

- Keep teams together between projects
- Teams have a coach
- Provide training to ramp up new people
- Evolve the process with retrospectives and experiments

Teams take time to form

- Three months to form a team… (Mitch Lacey)
- Learn working styles
- Learn the problem domain
- Distribution makes this slower

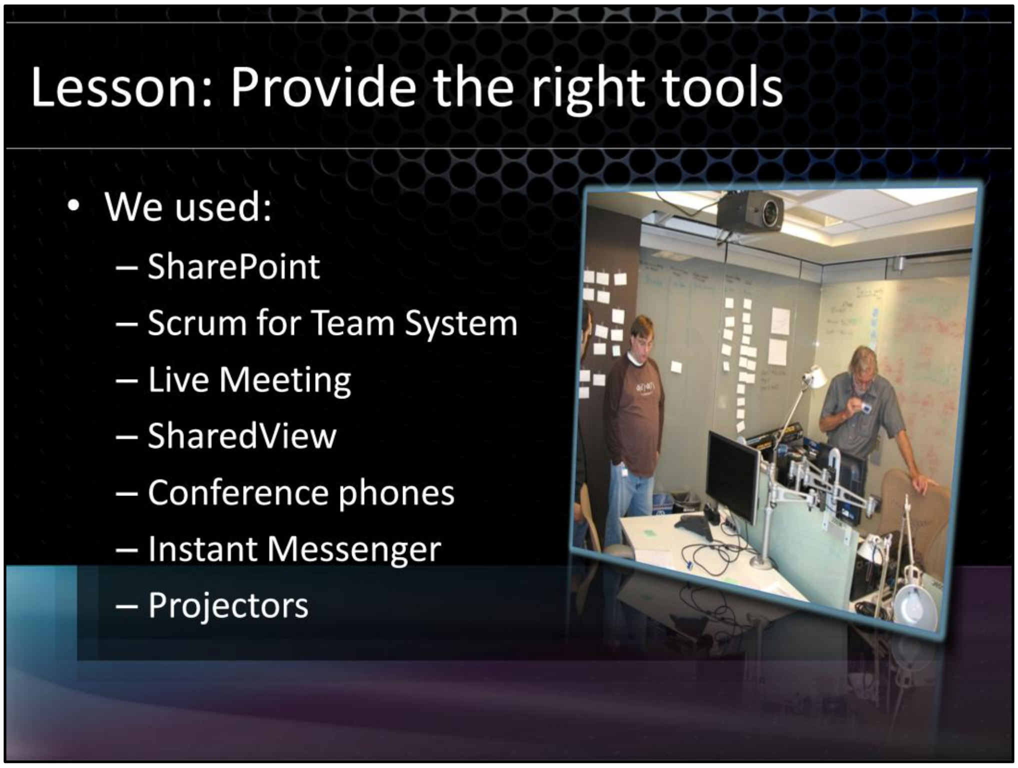
It's easy to abandon key practices if there's nobody there to keep you on the straight and narrow.

Have someone who knows they're the coach!

- Coaches are *not* practice cops
- Remind team of underlying principles
- Help them adapt and modify

Provide the team with training so everyone knows what to expect.

- @ p&p this is a wiki and books.
- Hold retrospectives!



Lesson: Provide the right tools

- We used:
  - SharePoint
  - Scrum for Team System
  - Live Meeting
  - SharedView
  - Conference phones
  - Instant Messenger
  - Projectors

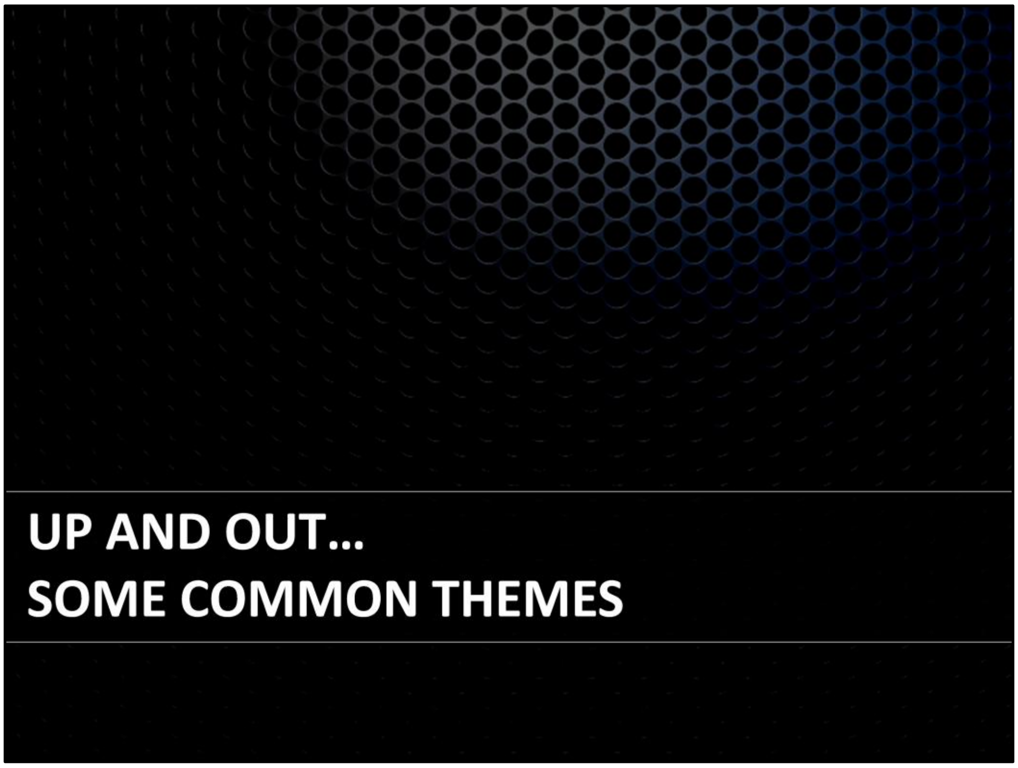
Tools help but they're not a silver bullet

The reason agile teams use things like sticky notes is because they're easy to adapt and don't constrain you.

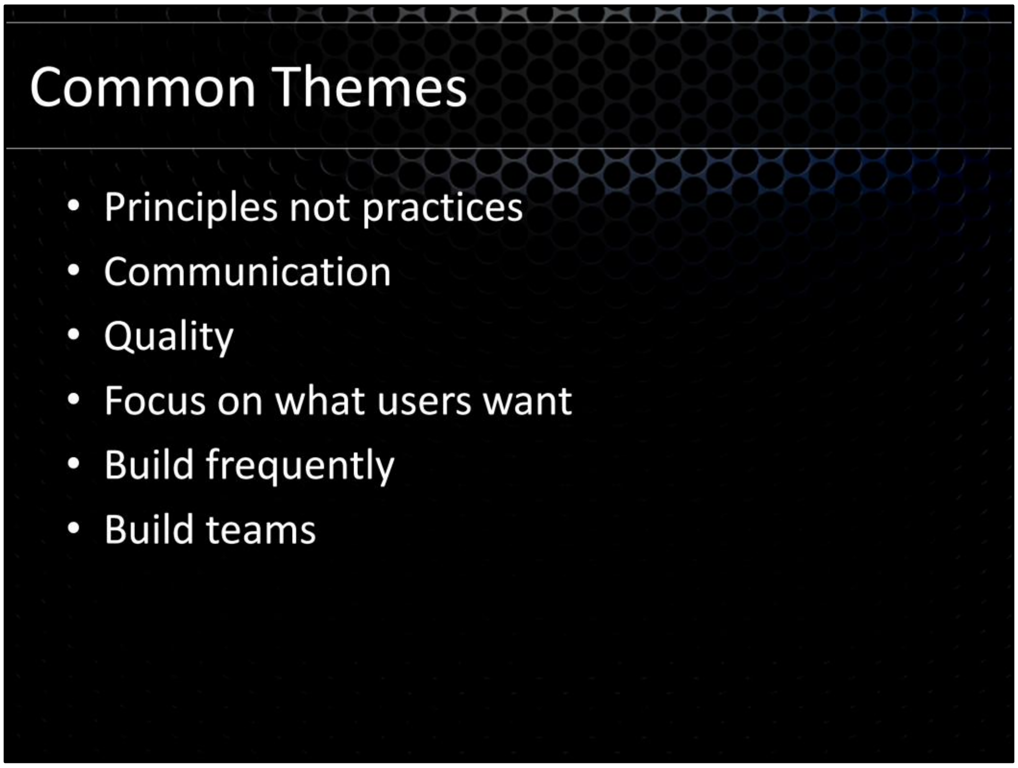
When you provide tools that support distribution make sure they don't constrain you.
- Don't replace something simple with something overly complex!

These are the tools p&p uses. You may use different tools that fulfill the same function.

This is NOT a Microsoft sales pitch!

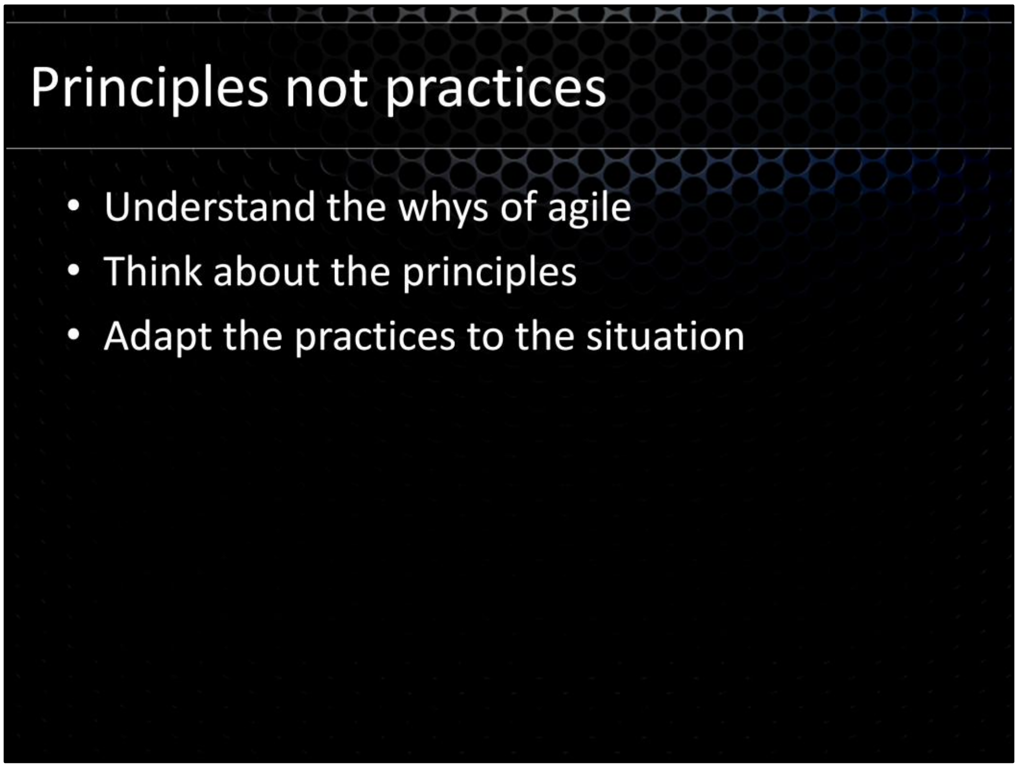


UP AND OUT…
SOME COMMON THEMES



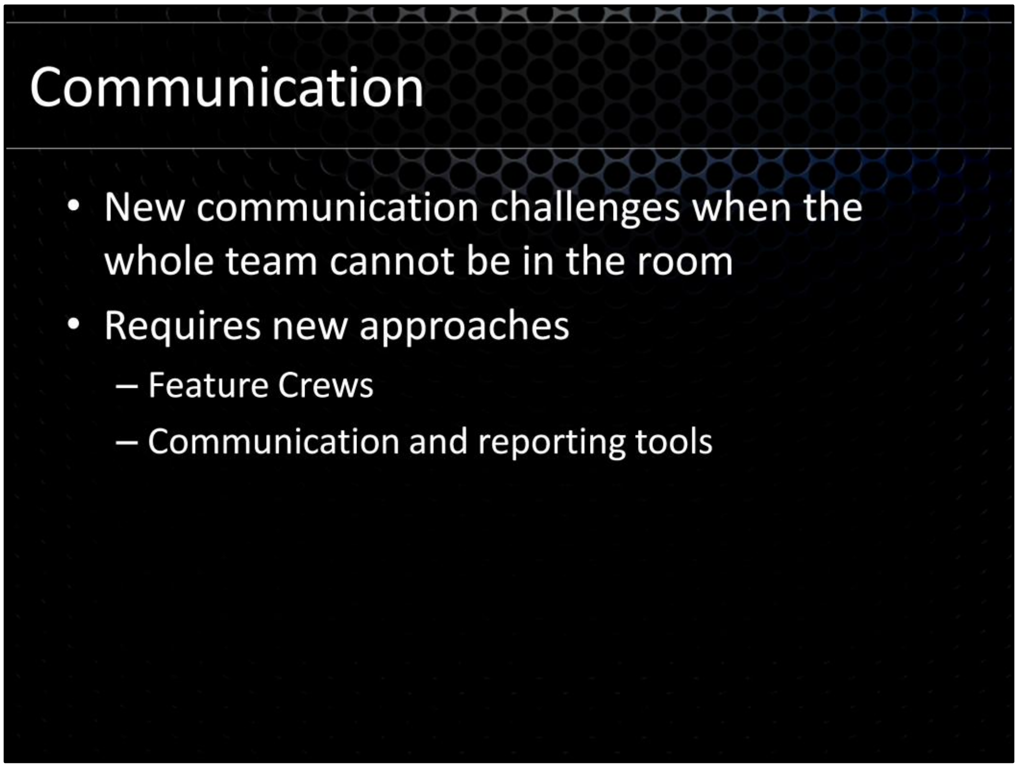
Common Themes

- Principles not practices
- Communication
- Quality
- Focus on what users want
- Build frequently
- Build teams



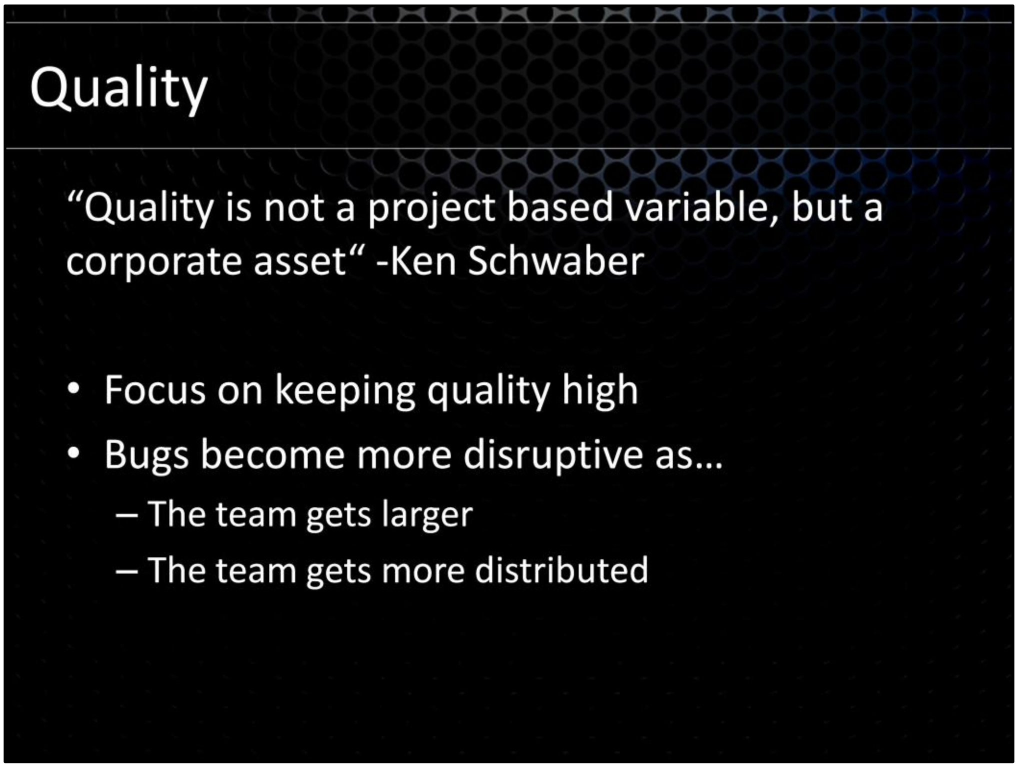
Principles not practices

- Understand the whys of agile
- Think about the principles
- Adapt the practices to the situation



Communication

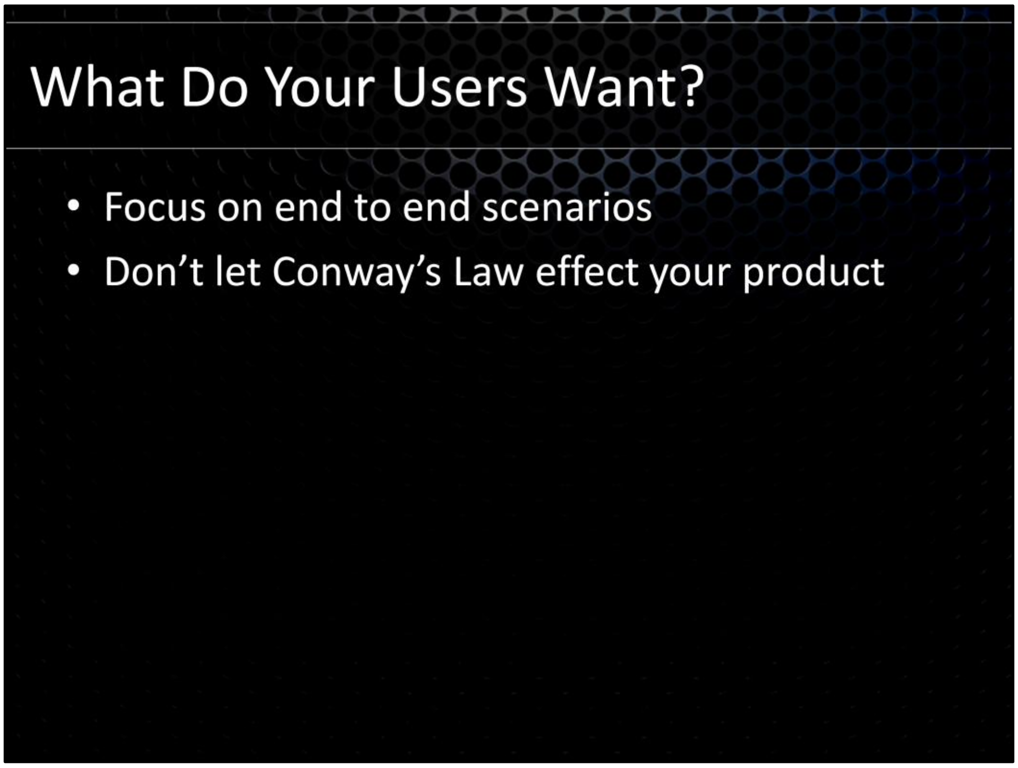
- New communication challenges when the whole team cannot be in the room
- Requires new approaches
  - Feature Crews
  - Communication and reporting tools



Quality

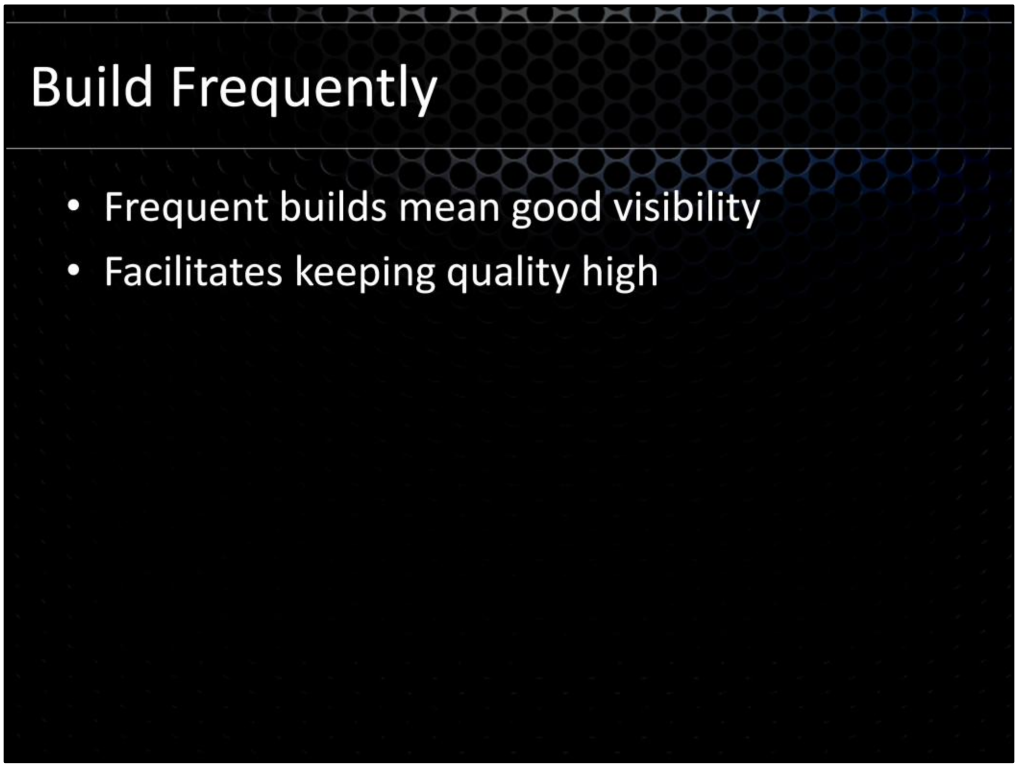
"Quality is not a project based variable, but a corporate asset" -Ken Schwaber

- Focus on keeping quality high
- Bugs become more disruptive as…
  - The team gets larger
  - The team gets more distributed



What Do Your Users Want?

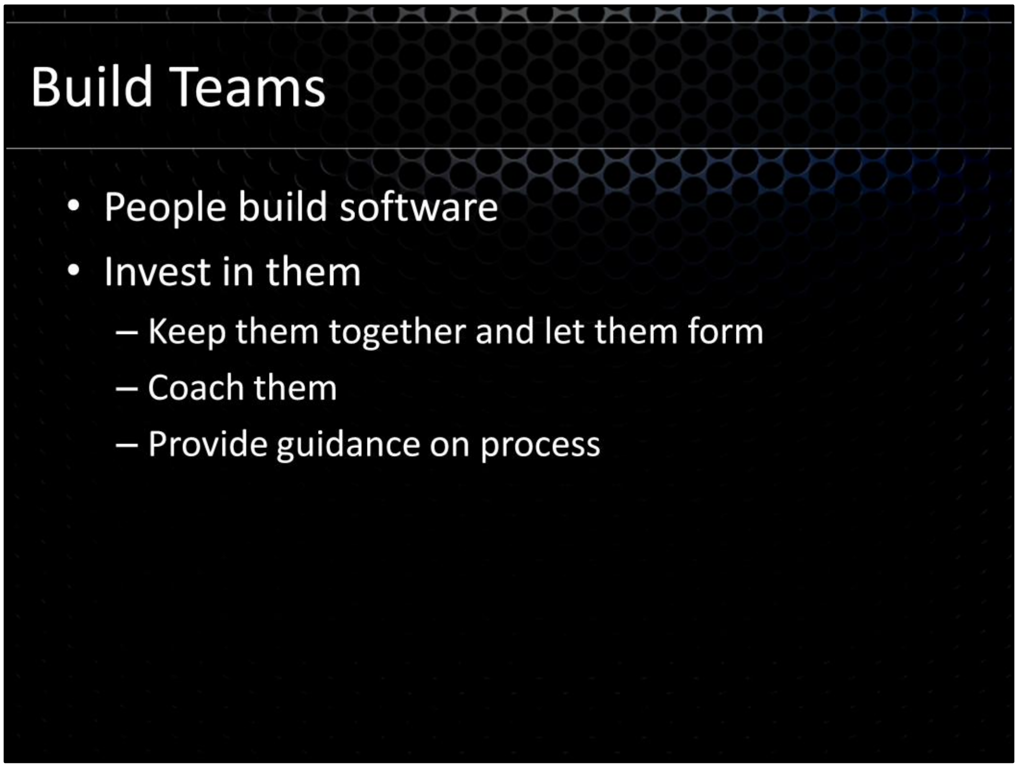
- Focus on end to end scenarios
- Don’t let Conway’s Law effect your product



Build Frequently

- Frequent builds mean good visibility
- Facilitates keeping quality high

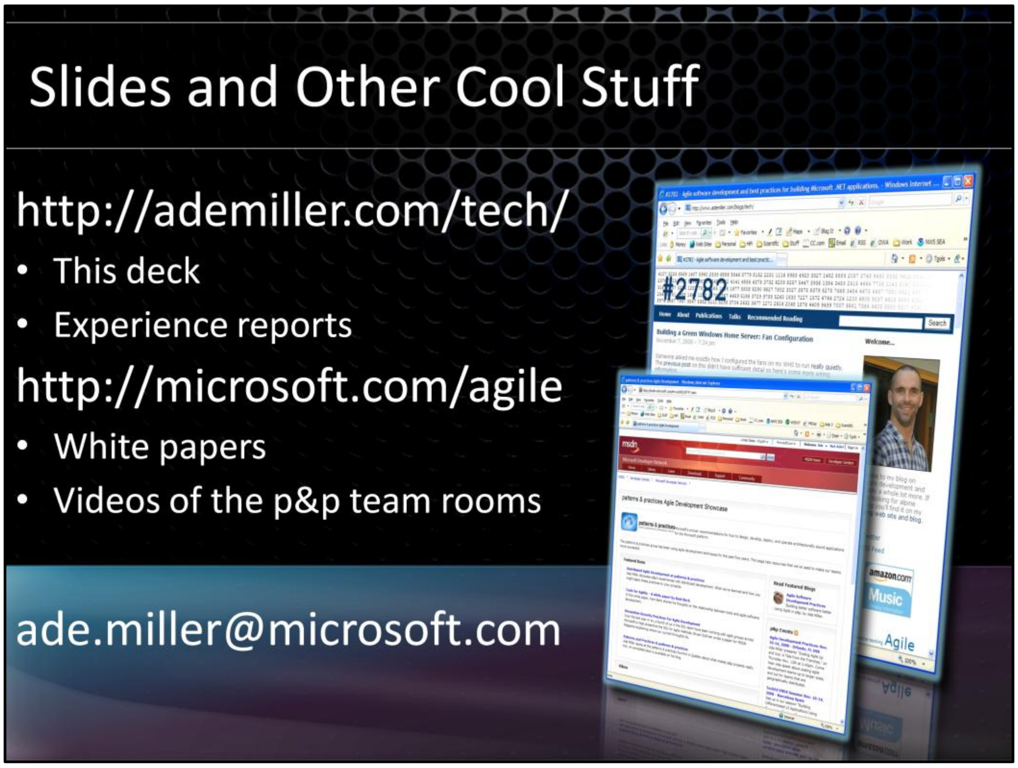
I didn’t mention CI when we talked about distributed teams but we used it heavily



Build Teams

- People build software
- Invest in them
  - Keep them together and let them form
  - Coach them
  - Provide guidance on process

Process is a dirty word but for large or distributed teams you need some



Slides and Other Cool Stuff

http://ademiller.com/tech/

- This deck
- Experience reports

http://microsoft.com/agile

- White papers
- Videos of the p&p team rooms

ade.miller@microsoft.com