

## Notes

The speaker notes for this deck ran off the slide in several places. This PDF includes additional blank slides that were not in the talk to allow you to read the notes.

I talked for over an hour so the notes in these slides cover the basics. I've also included links to numerous blog posts which expand on the topic covered in each slide.

I gave this talk in May and I'm writing the notes in September. This is what I might have said. If you were there and remember otherwise then you're probably right and these notes are based on what I know now, not on what I knew then.

**OK... So here goes!**

Ade Miller is the Development Manager at patterns & practices

p&p is a small group within Microsoft Developer Division who ship guidance on application development to Enterprise customers. Things like...

- Enterprise Library
- Software Factories
- Books

We run several small (6-10 people) agile teams. These teams are often distributed and use a Scrum/XP based development approach.

This is why p&p delivers software the way it does *not* how the whole of Microsoft does it!



Lots of people tell us that they like the things that patterns & practices produces and want to know how we approach developing software.

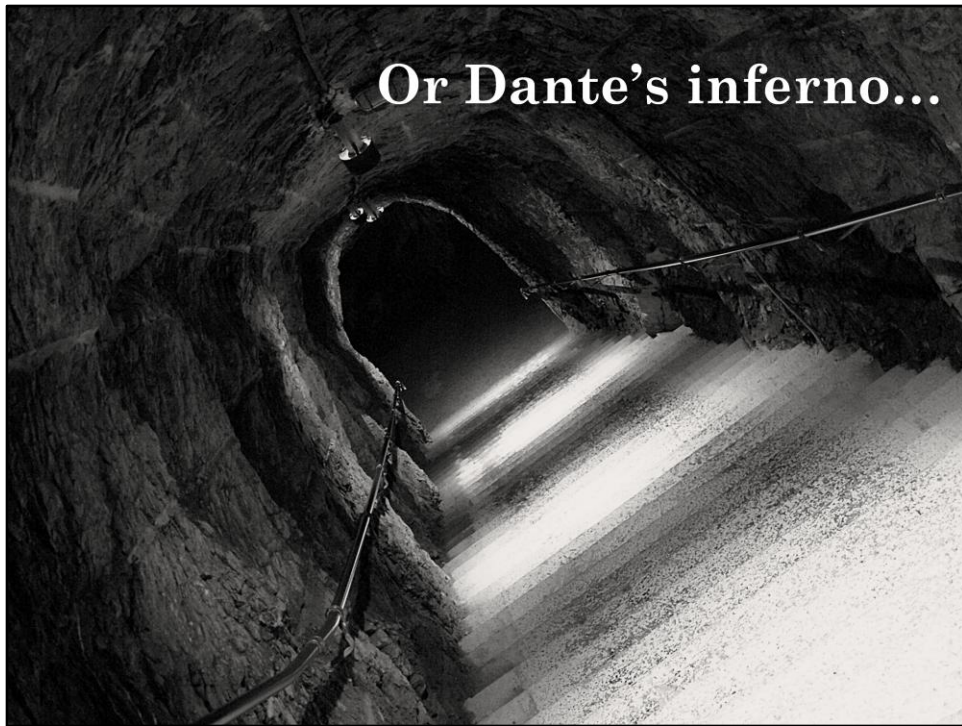
This talk is really about how we think about software development. Hopefully you'll come away with some cool ideas about how to think about your teams and helping them improve the way they develop.

Note... I didn't say you'll do exactly what we do. That's not such a good idea and we'll see why later.

---

Stock photo:

<http://www.sxc.hu/photo/920589>



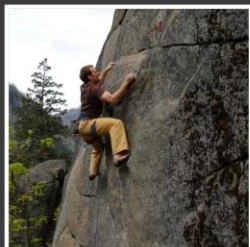
(Joke) Maybe we do a horrible job... but I don't think so.

If I'm way off base with that then now's the time to start throwing things and we can scratch this talk and have a whole different conversation.

---

Stock photo:

<http://www.sxc.hu/photo/876467>



## PRACTICES...

Practice, practice, practice

Lots of people think about development in terms of practices. Be it using source code control or pair programming the easiest way to think about how you're delivering software is to check the practices checkboxes.

THIS IS NOT THE BEST WAY TO THINK ABOUT THIS!

Practices are important and I someone says to me "We're not doing X", I want to ask why. This talk is more about the stuff *behind* the practices. We did some thinking about the values behind the practices and that's what I want to talk about today...

(So this talk isn't a detailed description of exactly *how* we deliver software, it's more about the thinking behind it)

---

Stock photo:

<http://www.ademiller.com/climbing/>

<http://www.sxc.hu/photo/851553>

<http://www.sxc.hu/photo/535933>



We've been doing a mixture of Scrum and XP for some time but this talk really comes out of some thinking we did over the 2007/2008 holiday season. We locked several people in a room and asked them to come up with a slightly more formal definition of what it is that we do to create software.

Imaginatively we called this activity "Project X".

I actually wrote a series of blog posts while we were doing Project X. You can read them in conjunction with this deck.

<http://www.ademiller.com/blogs/tech/2008/05/choosing-an-agile-process-summary/>

---

Stock photo:

<http://www.sxc.hu/photo/475457>



**Values** - Abstract, distinct ideals. For example [XP's values](#) are; courage, communication, feedback, respect and simplicity ([Extreme Programming Explained, edition 2](#) added respect).

**Principles** - Are values as applied to your problem domain or industry.

**Practices** - Practices are the principles applied to your team.

For example: Value -> Principle -> Practice(s)

Communication -> Whole Team -> Daily standup / shared workspace / collective ownership

Lots of people work backwards from the practices and evaluate their team based on practice adoption. While many of the agile practices are simply good engineering they need to align to your team's values.

I wrote more on value alignment here:

<http://www.ademiller.com/blogs/tech/2008/01/choosing-an-agile-process-part-3-evaluation/>

---

Stock photo:

<http://www.sxc.hu/photo/1001265>



Values are very core to people.

Unfortunately it often turns out that getting people to agree to a set of values is easy. This is largely because values are vague and lofty. For instance who's going to say they don't value communication (raise your hands)? That's because "communication" is so vague that it's easy to say yes and disagree on the definition later. Just as importantly communication sounds like a laudable goal and you'd feel like a bit of an idiot saying you don't value it flat out!

Where things tend to break down is aligning those values to principles and practices. For example everyone can agree on communication but half the team refuse to turn up to standup meetings and want to report status by email once a week!

I wrote more on value alignment and what happens when your team's values don't align:

<http://www.ademiller.com/blogs/tech/2008/01/choosing-an-agile-process-part-3-evaluation/>

---

Stock photo:

<http://www.sxc.hu/photo/504217>

## Buy vs. Build



We spent the early days in the Project X room looking at books on development methodologies and contemplating our own navels. In fact this whole project almost disappeared down a rabbit hole until we realized that choosing a methodology is just another buy vs. build decision. It turns out that none of the downsides that you usually associate with buying something really apply to a methodology. For example if you buy something usually it's harder to modify it if it's not quite what you wanted. Agile methodologies expect you to modify them.

You can read more about this phase of the project in a couple of blog posts I write at the time, they explain the buy vs. build tradeoff in more detail...

<http://www.ademiller.com/blogs/tech/2008/01/designing-your-own-agile-process-part-1/>

<http://www.ademiller.com/blogs/tech/2008/01/designing-your-own-agile-process-part-2/>

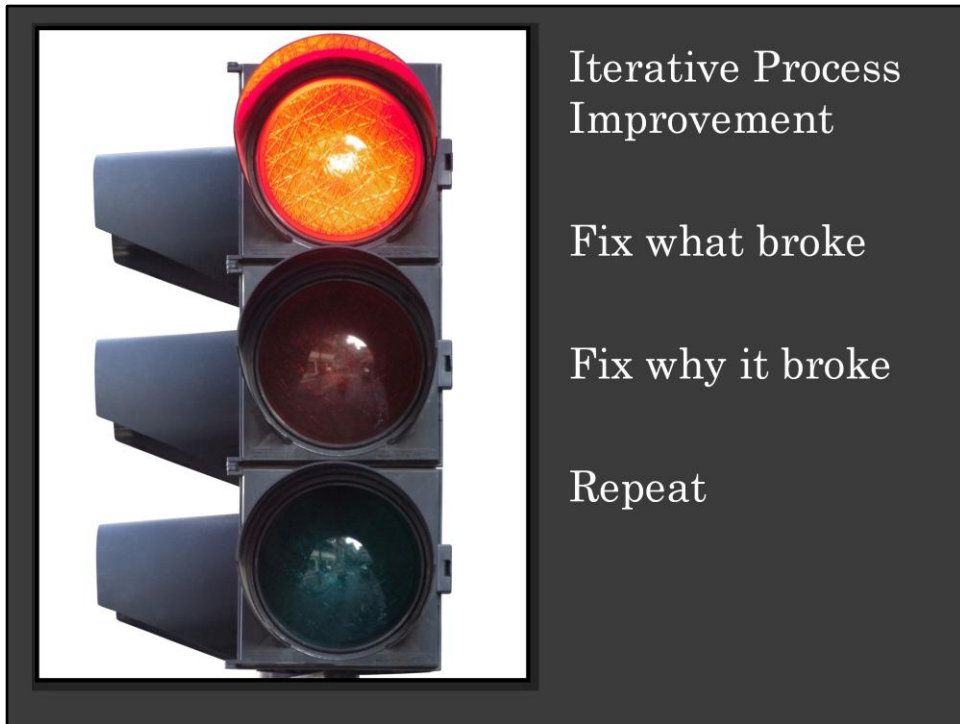
The process we decided upon is pretty much that outlined in "Scrum and XP from the Trenches" by Henrik Kniberg. You can download it here:

<http://www.infoq.com/minibooks/scrum-xp-from-the-trenches>

---

Stock photo:

<http://www.sxc.hu/photo/208866>



Project X produced something I like to refer to as a “baseline”. In other words we came up with a starting point for the teams and *actively encouraged* them to adapt to their needs.

This is common to all agile processes – they are adaptive.

Again, if you don’t have some values and principles behind your practices you’ll find this a lot harder to do.

For example... Your team is very distributed so pairing turns out to be much harder. Sure you can pair remotely using SharedView and the like. Sometimes even this isn’t possible because of time zone differences. If you don’t understand the principles and values (communication) behind pairing then it’s hard to figure out what to do next. Maybe you just drop the pairing practice? At p&p we substitute both individual and team code reviews for pairing.

---

Stock photo:

<http://www.sxc.hu/photo/433203>



## PATTERNS and anti-patterns

In the process of implementing project X on a couple of our teams we came up with some patterns and anti-patterns to watch out for.

We wouldn't be patterns & practices otherwise now would we?

---

Stock photo:

<http://www.sxc.hu/>



Hire the best people you can! I'm lucky in that my whole team are star developers *who work well with others on a team*. This is different from the solo hero or rock star who can do great things *on their own* and isn't interested in working with other people.

<http://www.ademiller.com/blogs/tech/2008/01/choosing-an-agile-process-part-5-heroes-and-villains/>

During the talk someone asked me if agile will work with more junior people. The answer is yes if you have some senior people who are prepared to help them grow and value the team over their individual goals.

Martin Fowler has a blog post about the false economy of hiring lower quality people which is also worth a read...

<http://www.ademiller.com/blogs/tech/2008/03/the-high-cost-of-talent/>

---

Stock photo:

<http://www.sxc.hu/photo/930008>



The team is more important than the people on it! France winning the Soccer World Cup in 1998 with a team of team players not a team of superstars. On several occasions I've seen the superstar developer leave a team and everyone predict it would fall apart and the project would fail. Only to have the team exceed expectations!

<http://www.ademiller.com/blogs/tech/2007/05/does-agile-need-extremely-competent-people-in-order-to-work/>

<http://www.ademiller.com/blogs/tech/2006/05/software-development-is-a-team-sport/>

---

Stock photo:

<http://www.sxc.hu/photo/825498>

## Deliver frequently... Or become irrelevant



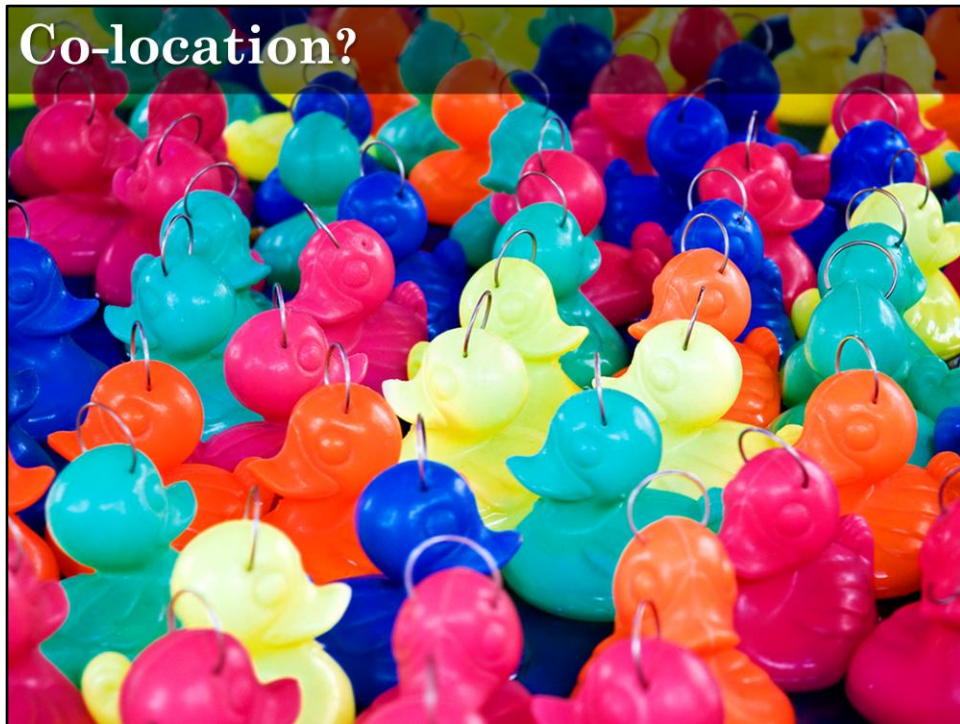
Teams at patterns & practices all deliver frequently! Most teams deliver *working software* to customers (you) every couple of weeks. This is the single biggest advantage that agile development gives to us... the ability to deliver frequent and *adapt our thinking* based on feedback on what we just delivered!

If you look at the most basic agile methodology – Crystal Clear – at it's heart is stick a bunch of smart people in a room and let them have at it. But... *make them show you working software frequently*. Working software is what keeps everyone honest.

---

Stock photo:

<http://www.sxc.hu/photo/918252>



I've written so much about this subject it's not even funny. You can find a whole series of blog posts on this at:

<http://www.ademiller.com/blogs/tech/tag/distributed-teams/>

The next slide has notes with a list of *just some* of the practices to consider. Distributed development is something we struggle with a lot but it's a fact of life for most of our teams.

In the 2008 State of Agile Development survey 57% of respondents stated that their teams were distributed. Furthermore 41% of respondents said that they were currently using or plan to combine agile with outsourced development.

<http://www.ademiller.com/blogs/tech/2008/09/who-moved-the-cheese-team-rooms-and-agile/>

---

Stock photo:

<http://www.sxc.hu/photo/967027>



Co-location

Dislocation

Dissociation

Dysfunction

I compiled this list after reading a lot and thinking about some of the things we were doing. Think of these more as suggestions rather than proven practices used today at p&p:

**Co-locate as much as possible:** Whenever possible, co-locate your entire team in one location. If this is not possible then plan for team members to spend periods traveling and working at the main or offshore site, especially at the start of a project - the first few iterations - when key architectural/design decisions will be made.

**Align the team locations:** The more offshore locations your team has the higher the communication barriers become. When using offshore team members group them into sub-teams aligned by feature (see below).

**Time zones add further tax:** One of the hardest aspects of distributed collaboration is working across time zones. Try to minimize the distance between the team locations as much as possible and try to establish 'core working hours' that all members can adhere to.

---

Stock photo:

<http://www.sxc.hu/photo/980600>



**Pay the tax associated with distribution:** Co-location of your team allows them to communicate rapidly with minimal formal processes. If you have distributed team members be prepared to pay the tax associated with this. For example, your specifications will need to be much more complete to offset communication that would have occurred in a team room - you'll be writing much more complete story cards for example. More pre-work will also be required for meetings, like getting user stories in shape prior to iteration planning so that the team can review and ask clarifying questions via email rather than during the meeting.

**Improve communication where possible:**

- Get good communication going from the outset of the project.
- Make sure everyone has access to conference phones and reliable Internet connectivity.
- Consider having an open instant messenger, IRC or conference phone during the core working hours.
- Provide hands free headsets so that team members can pair remotely.
- Have a camera on hand for taking whiteboard pictures.
- Involve the whole team in meetings not just a single local representative.

**Provide appropriate tools:**

- Make sure that your SCCS and other team tools will function effectively across each remote site. For example consider adding a TFS proxy server at remote locations.

**Focus on team consistency:** It takes time to build up good working relationships, especially on distributed teams. Try and minimize churn on teams so that this is not lost.

**Onshore representative for offshore team:** Have someone in the team room how is responsible for being the offshore team's person in the room. This isn't to get the offshore people off the hook for attending standup or using the [other practices](#) I discussed to maximize communication. It's designed to help the remote team members stay sync'ed up with key conversations they may have missed.

## Beware of shiny objects



In 1986 Fred Brooks said that there was no silver bullet. That's still true today. So don't think that any one practice, technology or tool is going to make all your problems go away. Especially when it comes to the actual *activity* of software development it's more about the people than anything else. Don't expect a tool to change the way people behave.

BTW: Fred Brooks also wrote the Mythical Man Month. This is one of the few books in software engineering that has truly stood the test of time (over thirty years). If you've not read it then you should.

[http://en.wikipedia.org/wiki/The\\_Mythical\\_Man-Month](http://en.wikipedia.org/wiki/The_Mythical_Man-Month)

If we knew in 1985 that adding more people to a project doesn't speed up delivery then why are we still doing it over thirty years later?

---

Stock photo:

<http://www.sxc.hu/photo/914361>



(Joke) And neither is hockey... unless you're Canadian

Seriously though... It's very easy to become an "agile zealot".

Someone asked me yesterday "How do I convince people to do Agile?"

My answer is "Don't". By that I mean I don't think you should adopt Agile for Agile's sake alone, that's just "The means justify the ends". It has to be solving some issues in your organization. Organizational change requires effort so you need to build a case for it.

---

Stock photo:

<http://www.sxc.hu/photo/427948>



What you should be asking is what are your biggest challenges?

If the answer is that everything is just fine but my developers don't like the soda we provide then Agile isn't going to solve the soda problem. If the response is more along the lines of shipping poor quality software, now that's a problem agile can help with. Figure out their pain points, figure out where they want to be and then start applying some agile principles to help them get there.

Taken from the following blog post:

<http://www.ademiller.com/blogs/tech/2008/02/choosing-an-agile-process-part-7-making-the-case/>



QUESTIONS?

I'm done!

---

Stock photo:

<http://www.sxc.hu/photo/728931>

<http://www.sxc.hu/photo/418215>



## REFERENCES

[Scrum and XP from the Trenches](#), Henrik Kniberg

[Extreme Programming Explained, edition 2](#), Kent Beck

Microsoft patterns & practices

<http://msdn.microsoft.com/practices>

Patterns & practices Lab video

<http://channel9.msdn.com/posts/Charles/Tour-Patterns-and-Practices-Lab/>

Service Factory: ME

<http://msdn.microsoft.com/servicefactory>

Ade Miller's blog

<http://www.ademiller.com/tech/>

Stock photos in this presentation

<http://www.sxc.hu/>

Some other useful links.

---

Stock photo:

<http://www.sxc.hu/photo/971068>