Agile Development

Write the code right, and you can actually be agile

Peter Provost
Sr. Program Manager
Visual Studio

For developers, agile is about being on the defensive

Offense vs Defense

Offense

Product Owner

Business Leader

Stakeholder

Customer

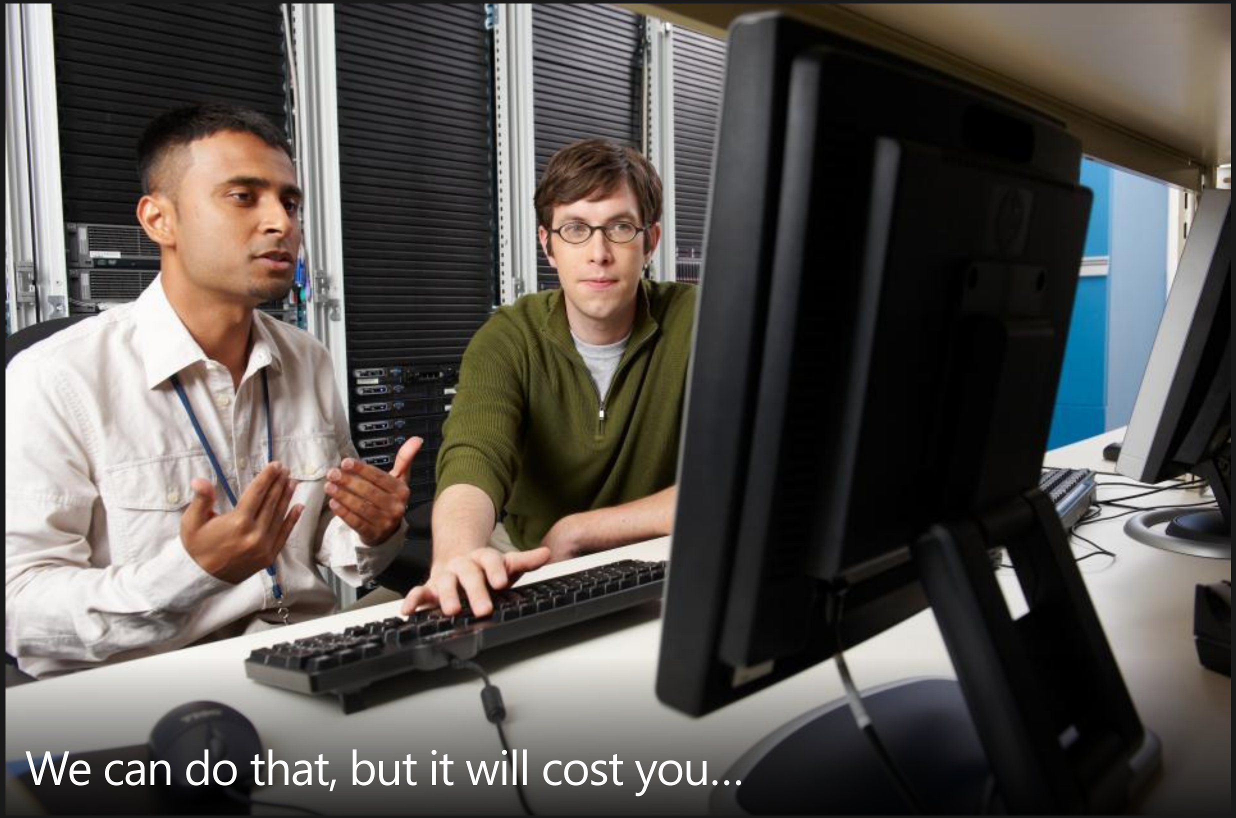
Defense

Developers

Testers

Designers

Architects


We can do that, but it will cost you…

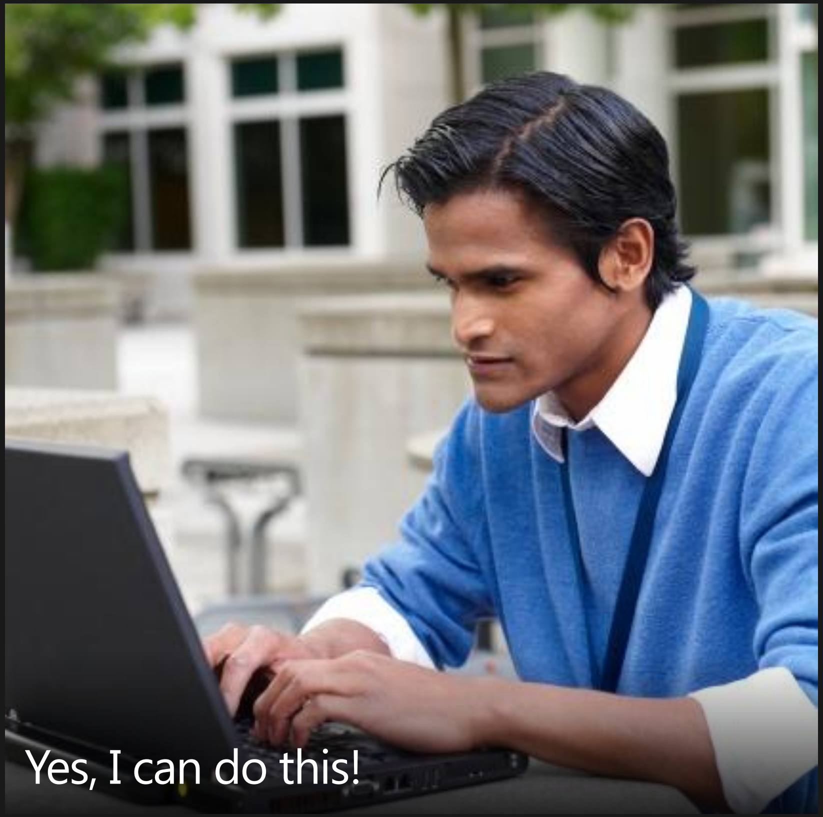
Try To Be In A Position To Say Yes

You own the design

If you ever find yourself wanting to say No to your Product Owner request due to technical reasons, the fault is in the design

Keep it flexible

The code should always be modular and clean so you can change things without rewriting huge chunks



Always think about design

Simplicity is key

Skeleton Architecture

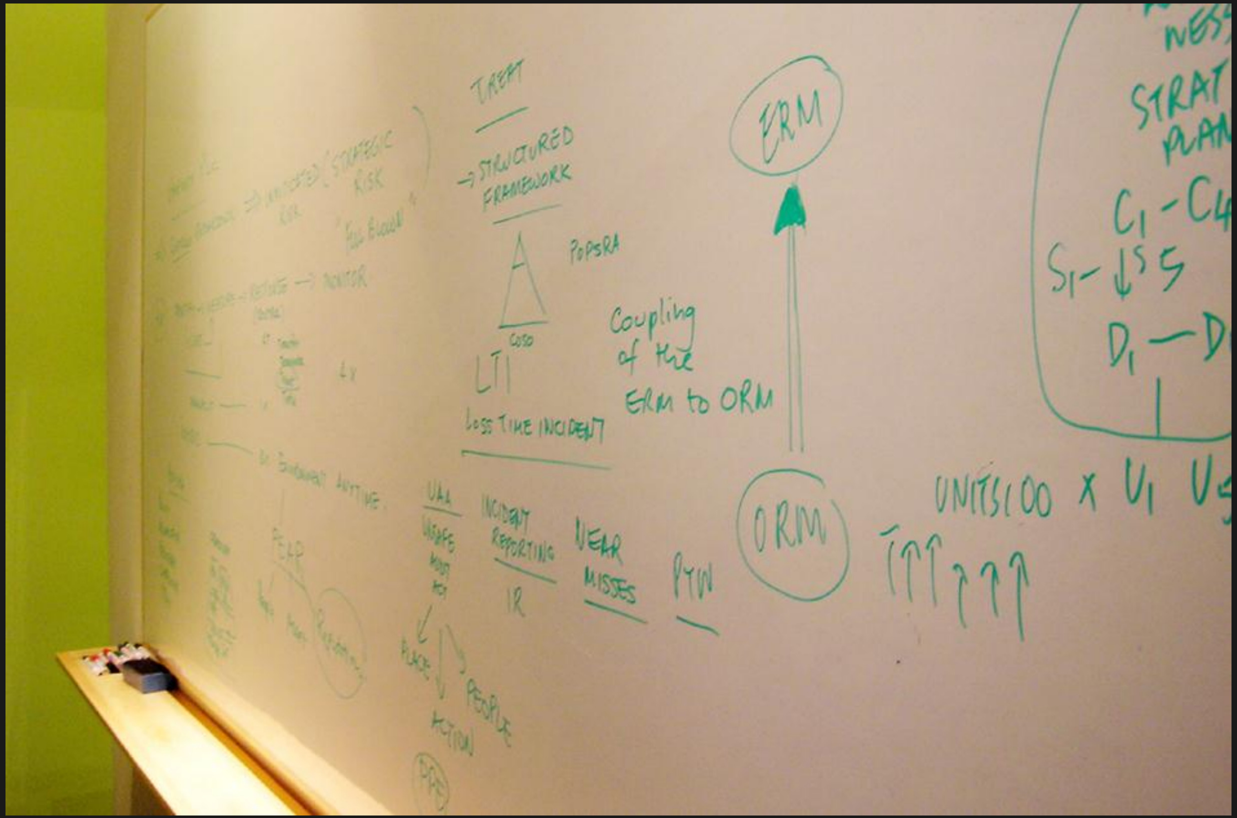
Know Where You Are

Have a shared, clear and consistent picture of your overall architecture

Embrace change

Even in your architecture!, things can and will change

Have open, inclusive conversations when you need to adjust the big picture



Agree on What Done Means

Done means done

The team must agree what it means to be done

Include everything you care about

But it will vary

Task, Story, Experience, Release, etc. all have their own definitions

Make it your own

My done is not your done

I am done-done!

Like everything else, your definition of done will change

Don't over-complicate things

The Scientific Method for Code

Unit test into goodness

Test the smallest testable unit of code

Hard to test? That is telling you something

Control and confirm

Control the experiment

Validate the results match what you expected

Learn from failure

Passing tests don't prove anything

Failing tests tell you when something needs attention

Don't write more code than is required to make the experiment succeed

Predict and verify, then reuse it so it doesn't change

Test Everything, All The Time

The best bang-for-the-buck

The single best thing you can do to protect yourself from changing requirements

The only way to make a change with confidence

Care about your tests

Unit tests require care and feeding

Run them all the time, with every checkin

If you can't run them all quickly, figure out why

Refactor Aggressively

Refactoring is not a backlog item

Your Product Owner doesn't want it

Remove duplication, or you will get bit later

Make room for new functionality

Care about the design

Would you sign your name on the bottom?

Would you put it in your resume?

Would you show it off in an interview?

Refactoring requires unit tests or you might break something

Be confident while you work, and be proud of what you create

Pair Programming

Everyone does it when they are in trouble

You’re not as bad as you think you are

You’re not as good as you think you are either

Come on, you know you want to

Baby Steps

Do one thing at a time

Throughput goes down exponentially when you do more than one task as a time

Do it well

Do just enough, but make sure what you do is great

Then continue on

Pick up the next task and do it all again

Thank you!

Peter Provost

Email: peter.provost@microsoft.com

Twitter: @pprovost