

# Spread the Love!

Involving Customers in Agile Projects

# Who - Me?

## Max Muermann

### email

max@synaphy.com

max@muermann.org

### web

<http://whatsnextapp.com>

### web 2.0

[twitter.com/maxm](https://twitter.com/maxm)

# Disclaimer

There will be no explosions in  
this presentation

# The problem is...

- ▶ People want to control what they don't understand
- ▶ Your managers/ customers believe you work better like this:

# The problem is...

- ▶ People want to control what they don't understand
- ▶ Your managers/customers believe you work better like this:



# But...

- ▶ But they really want you to do this:

You need to allow your customers to feel in control. You cannot do that by letting them attempt to specify every parameter of a project up front. Therefore, you must involve them throughout the project!

# But...

- ▶ But they really want you to do this:



You need to allow your customers to feel in control. You cannot do that by letting them attempt to specify every parameter of a project up front. Therefore, you must involve them throughout the project!

# What can we do?

- ▶ Agile development methods deliver better software in a shorter amount of time

# What can we do?

- ▶ Agile development methods deliver better software in a shorter amount of time
- ▶ Most “Agile” project are not agile at all

# What can we do?

- ▶ Agile development methods deliver better software in a shorter amount of time
- ▶ Most “Agile” project are not agile at all
- ▶ What can go wrong - and how can we prevent it?

# The Agile Manifesto

**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

<http://agilemanifesto.org>

We have it better than the Java folks!

Individuals – if you preferred processes and tools, you wouldn't be here!

Working Software:

RUnit, RSpec, Autotest

Capistrano

Cruisecontrol.rb

and shitloads less code!

Responding to Change

most of us work in really small teams

iterations are short

normally an absence of Gantt-chart-driven project managers

# So Ruby and Rails make some of this easy...

- ▶ But ensuring Customer Involvement is still a problem!

# Getting customer commitment can be difficult

▶ “Our customers have day jobs”

1. if they need a new piece of software to *do* their day jobs, then software development *is* part of their day job!
2. then develop a joint vocabulary that enables users and developers to speak the same language!
3. users have been trained through years of mismanaged software projects to demand everything and the kitchen sink up front. Solve the basic business problems first and educate them about how you can accommodate change in the project.
4. how are you going to deliver a great piece of software without any user commitment??!?

# Getting customer commitment can be difficult

- ▶ “Our customers have day jobs”
- ▶ “Our customers can’t understand the technical details”

1. if they need a new piece of software to *do* their day jobs, then software development *is* part of their day job!
2. then develop a joint vocabulary that enables users and developers to speak the same language!
3. users have been trained through years of mismanaged software projects to demand everything and the kitchen sink up front. Solve the basic business problems first and educate them about how you can accommodate change in the project.
4. how are you going to deliver a great piece of software without any user commitment??!?

# Getting customer commitment can be difficult

- ▶ “Our customers have day jobs”
- ▶ “Our customers can’t understand the technical details”
- ▶ “Our developers must be protected from unreasonable customer demands”

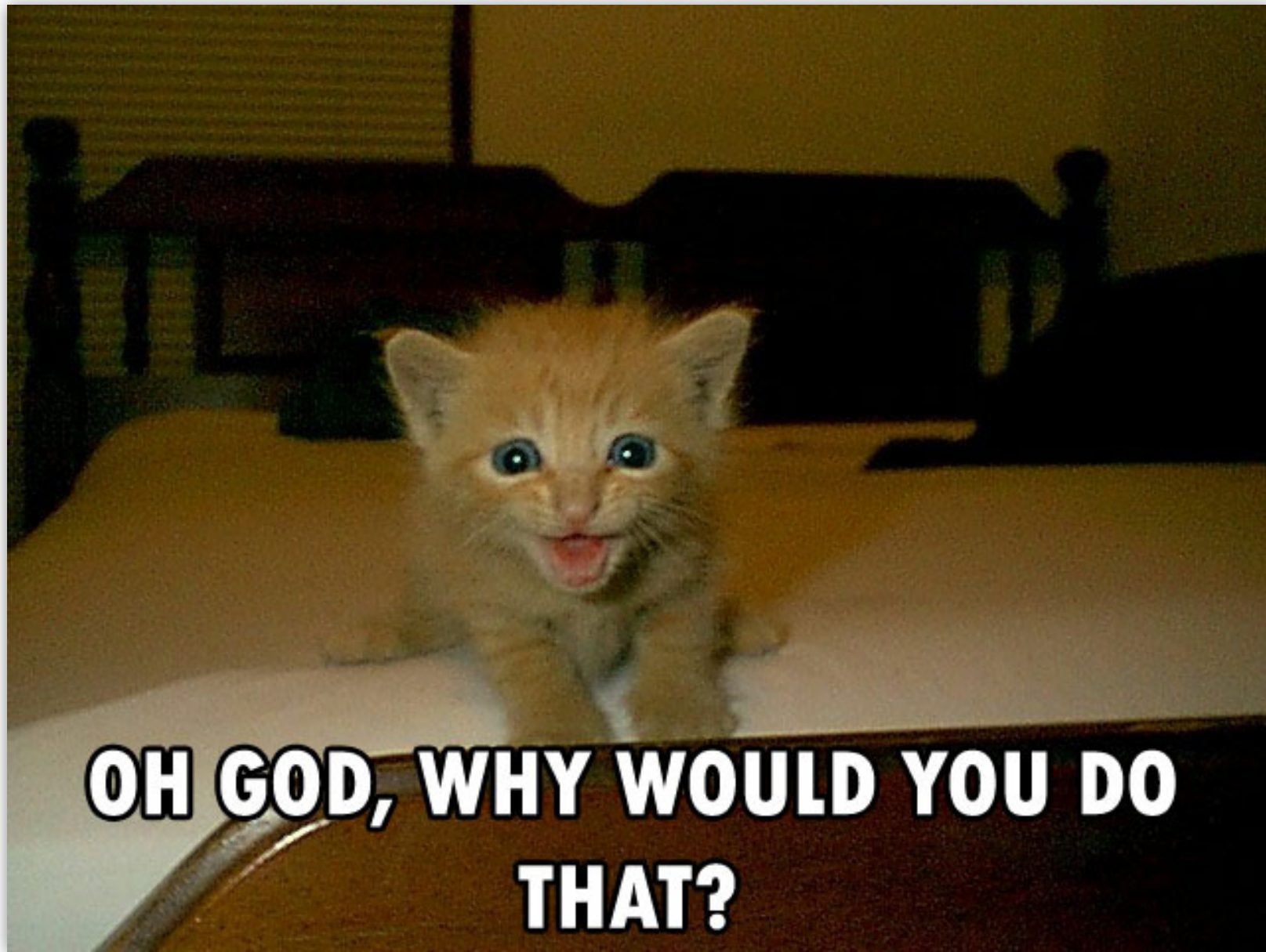
1. if they need a new piece of software to *do* their day jobs, then software development *is* part of their day job!
2. then develop a joint vocabulary that enables users and developers to speak the same language!
3. users have been trained through years of mismanaged software projects to demand everything and the kitchen sink up front. Solve the basic business problems first and educate them about how you can accommodate change in the project.
4. how are you going to deliver a great piece of software without any user commitment??!?

# Getting customer commitment can be difficult

- ▶ “Our customers have day jobs”
- ▶ “Our customers can’t understand the technical details”
- ▶ “Our developers must be protected from unreasonable customer demands”
- ▶ “We cannot talk to the customers because they are not 100% behind the project - it was a management decision”

1. if they need a new piece of software to *\*do\** their day jobs, then software development *\*is\** part of their day job!
2. then develop a joint vocabulary that enables users and developers to speak the same language!
3. users have been trained through years of mismanaged software projects to demand everything and the kitchen sink up front. Solve the basic business problems first and educate them about how you can accommodate change in the project.
4. how are you going to deliver a great piece of software without any user commitment??!?

# Getting customer commitment can be difficult



1. if they need a new piece of software to \*do\* their day jobs, then software development \*is\* part of their day job!
2. then develop a joint vocabulary that enables users and developers to speak the same language!
3. users have been trained through years of mismanaged software projects to demand everything and the kitchen sink up front. Solve the basic business problems first and educate them about how you can accommodate change in the project.
4. how are you going to deliver a great piece of software without any user commitment??!?

# The Waterfall Method

- ▶ Also called “linear sequential”
- ▶ Relies on flawless execution of each phase
- ▶ Discourages communication between teams
- ▶ Seems to be the most widely used development method
- ▶ I don't find this very appealing...

# The Waterfall Method

- ▶ Also called “linear sequential”
- ▶ Relies on flawless execution of each phase
- ▶ Discourages communication between teams
- ▶ Seems to be the most widely used development method
- ▶ I don't find this very appealing...



**In so many words...**

We want to give customers  
what they need right now, not  
what they managed to write  
down 6 months ago!

And the only way to find out what they want is to talk to them! Is that so hard to understand?

# There are some prerequisites

- ▶ Short iterations
- ▶ A working continuous integration system
- ▶ Working automated deployments

# Techniques to encourage Customer Involvement

- ▶ Involve customers and developers from the start of the project
- ▶ Co-locate if possible
- ▶ Model the Domain together
- ▶ Deliver frequently and incrementally
- ▶ Show-and-tell sessions
- ▶ Pair programmers with Customers

# Involve Developers and Customers From the Start

- ▶ Builds familiarity that lowers communication barriers
- ▶ Increases sense of project ownership in all involved parties

# Co-locate if Possible

- ▶ Face-to-Face is the most effective form of communication
- ▶ If co-location is not an option, consider:
  - ▶ Video conferencing
  - ▶ Internet Telephony
  - ▶ IM

Presence information is important – lowers the barriers to making a call/sending a message

# Joint Modeling Sessions

- ▶ Build familiarity between participants
- ▶ Develop a common vocabulary
- ▶ Develop a joint understanding of the business domain
- ▶ Develop a common understanding of the technical limitations

# Deliver Frequently and Incrementally

- ▶ Nightly integration builds
- ▶ Weekly test builds

Ensure that system urls are frequently (re)communicated to all involved users

Now – how do we make sure the users actually play with the software?

# Show-and-tell Sessions

- ▶ At least once every two weeks
- ▶ Secure firm time commitments!

# How to run Show-and-Tells

- ▶ Each developer gets up and walks through their work since the last presentation
- ▶ users are allowed to ask clarification questions, but *not* to discuss the implementation or underlying business requirements
- ▶ park all those issues and discuss in a separate forum

# Pair Programmers and Users

- ▶ Get users to sit with programmers for an hour every week
- ▶ Most changes affect the user interface
- ▶ Developers make small changes (<5min) on the fly
- ▶ Larger changes are recorded and parked
- ▶ Builds Users' understanding of complexities involved
- ▶ Fosters sense of ownership of the end product

Users get the feeling they have been involved “hands-on” in the development process

# First Principles

- ▶ Get people to know each other
- ▶ Get people to talk to each other
- ▶ Force people into situations where they *have* to collaborate
- ▶ Foster a real sense of ownership in everybody involved in a project