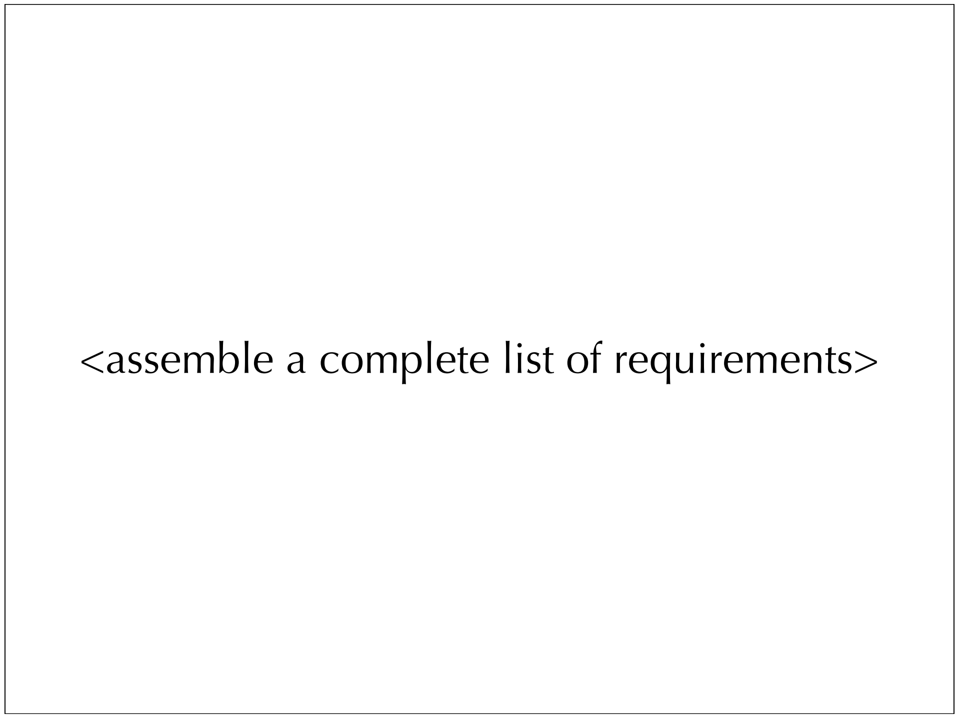
The university is planning to stop printing schedule books every semester.

Work in groups of 2 or 3 to create a list of features that you would like for the new on-line system to have.

Write each feature in one sentence.

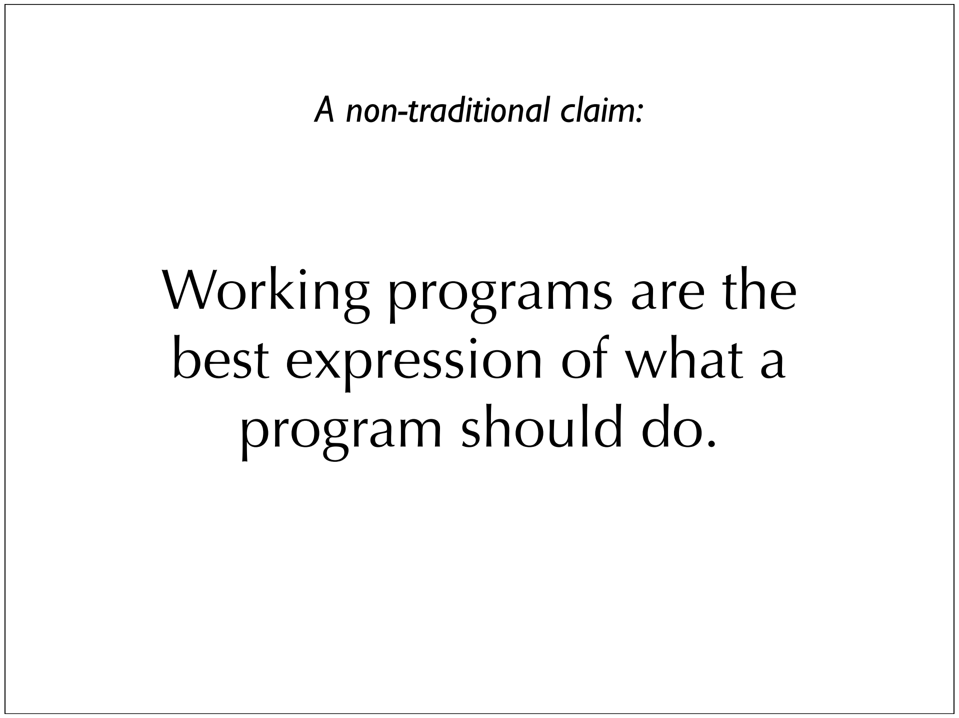
Example:
When a department changes the room for a course, it is immediately updated in the system.

.



<assemble a complete list of requirements>

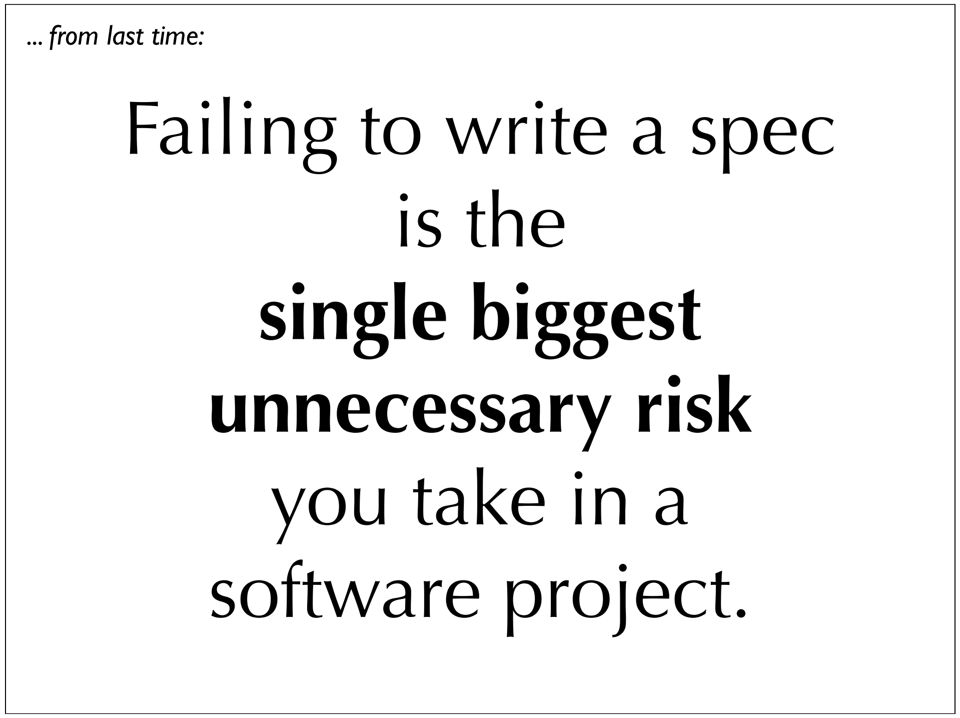
How many of these are ordinary cases? Special cases?
Are we missing any requirements in either category?
How can we know?



A non-traditional claim:

Working programs are the best expression of what a program should do.

Programming is **writing**, precise and concise(?).
Programs are **executable.**
The best documentation of what we know is a program.

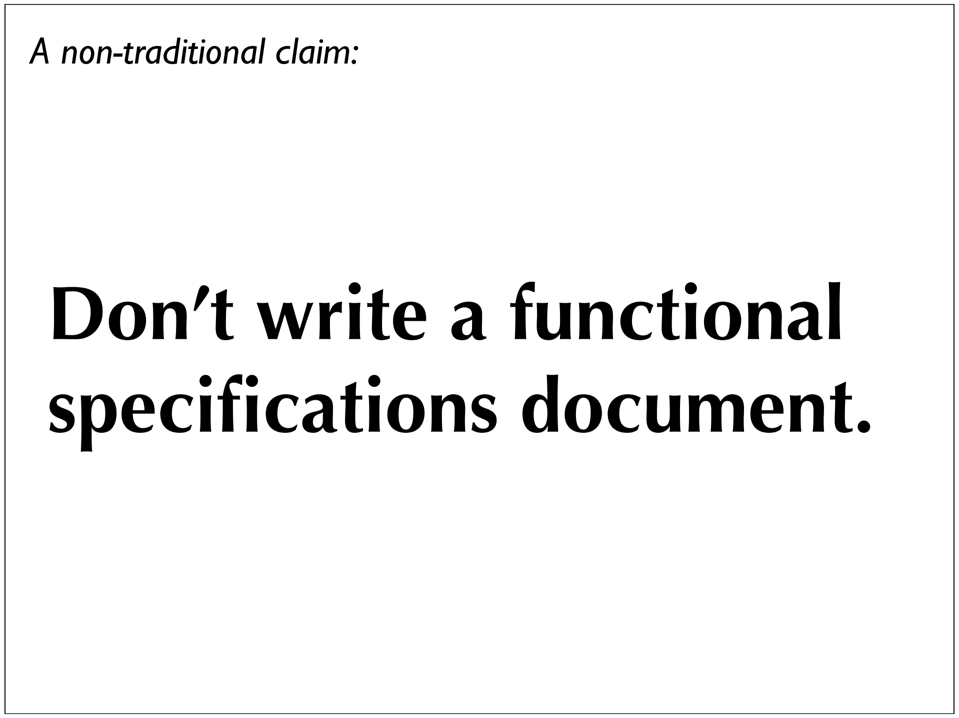


... from last time:

Failing to write a spec
is the
single biggest
unnecessary risk
you take in a
software project.

A quote from Joel Spolsky:
http://www.joelonsoftware.com/articles/fog0000000036.html

What is the consequence of not writing a spec?

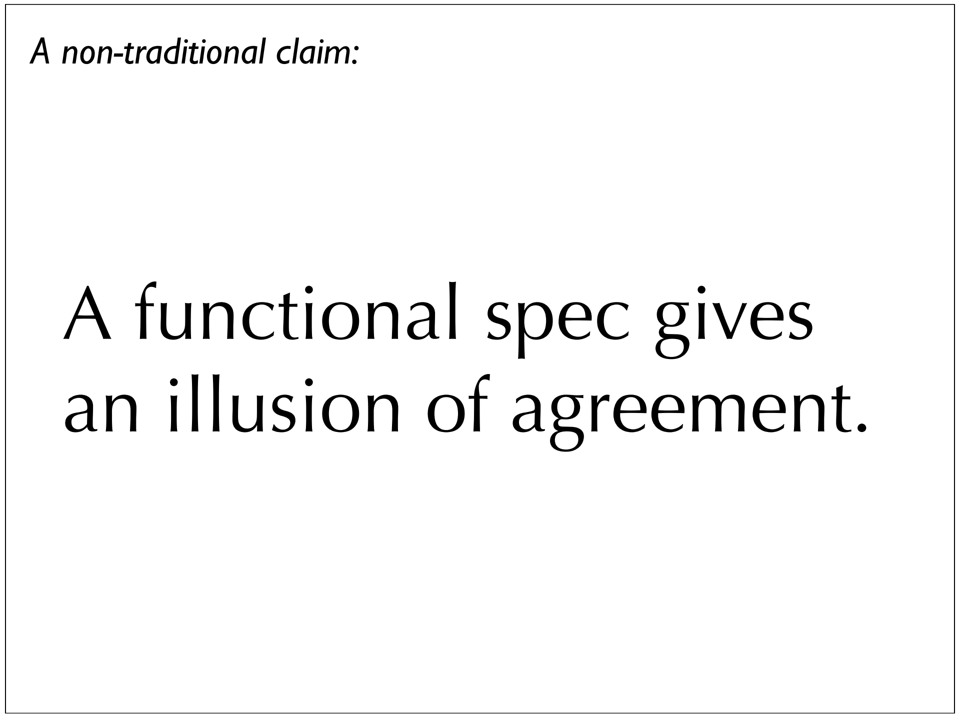


A non-traditional claim:

Don't write a functional specifications document.

A quote from Jason Fried:
http://37signals.com/svn/archives/001050.php

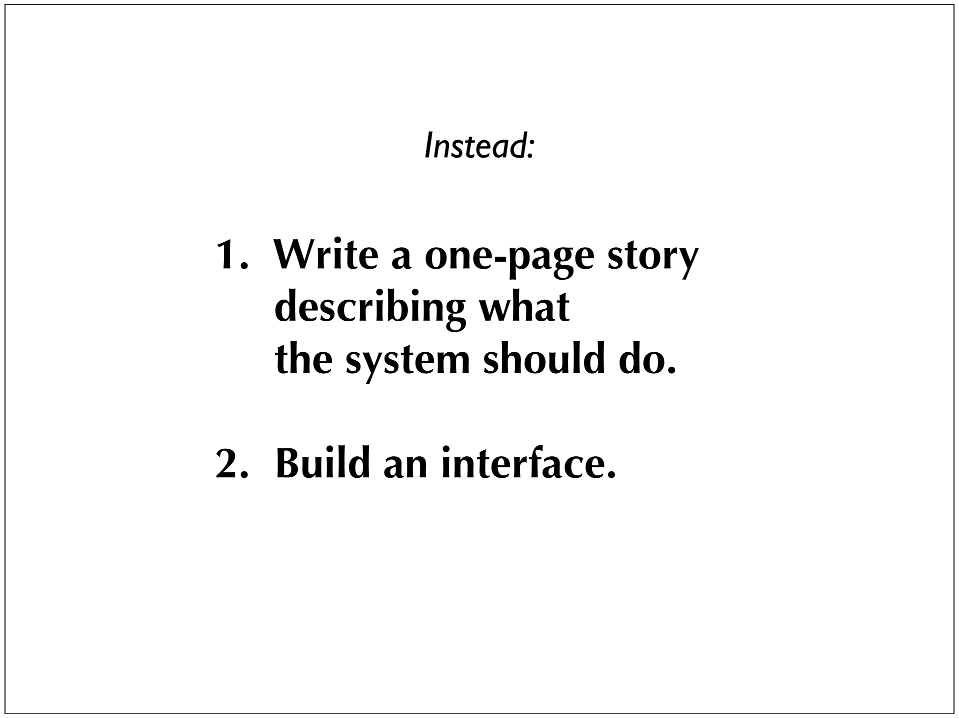
"Why? Well, there's nothing functional about a functional specifications document."


A non-traditional claim:

A functional spec gives an illusion of agreement.

A quote from Jason Fried:
http://37signals.com/svn/archives/001050.php

“Functional specs are about making decisions before you have enough information to decide.
They are about predicting the future and we all know how accurate that is.”



Instead:

1. **Write a one-page story describing what the system should do.**

2. **Build an interface.**

The story expresses the basic idea of the program.

The interface is a **functional spec**.

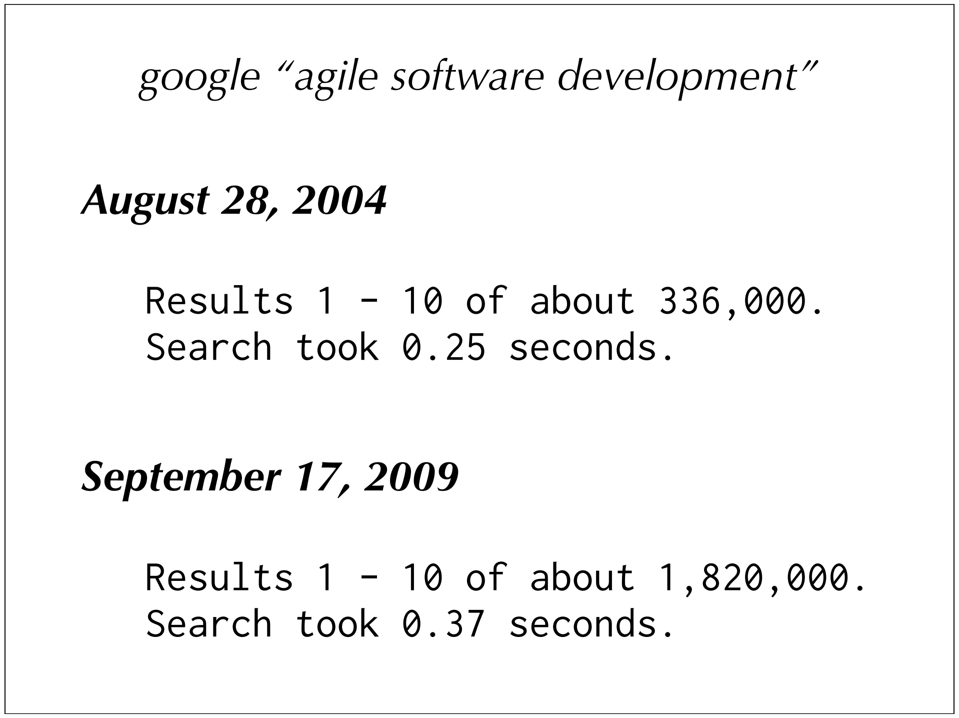
Can this possibly work?

Can it work for small systems? Big systems?
Can it work for small teams? Big teams?
Can it work for simple tasks? Complex tasks?
Can it work for web-based systems? For embedded systems?

agile software development

BaseCamp’s approach is one we can characterize as “agile”,
in contrast to the “sturdier” approach we have been discussing.

google “agile software development”...



google "agile software development"

August 28, 2004

```
Results 1 - 10 of about 336,000.
Search took 0.25 seconds.
```

September 17, 2009

```
Results 1 - 10 of about 1,820,000.
Search took 0.37 seconds.
```

google "agile software development"

August 28, 2004

```
     Results 1 - 10 of about 336,000.
     Search took 0.25 seconds.
```

September 17, 2009

```
     Results 1 - 10 of about 1,820,000.
     Search took 0.37 seconds.
```

September 17, 2009

Individuals and interactions
over processes and tools

Working software
over comprehensive documentation

Customer collaboration
over contract negotiation

Responding to change
over following a plan

These are the values expressed in the Agile Manifesto http://agilemanifesto.org/

There **is** value in the bottom items. Agile developers tend to value the items on top **more.**

“Our highest priority is to
satisfy the customer
through early and
continuous delivery
of valuable software.”

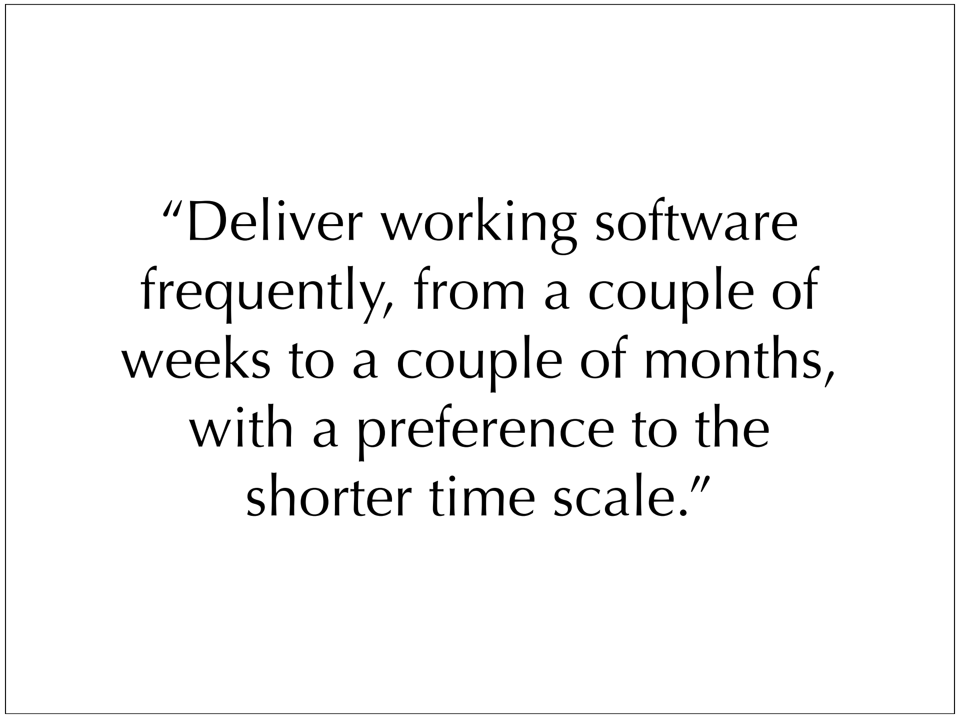
a principle of agile software development

... delivering real value for time (= money) spent.

“Welcome changing requirements, even late in development.”

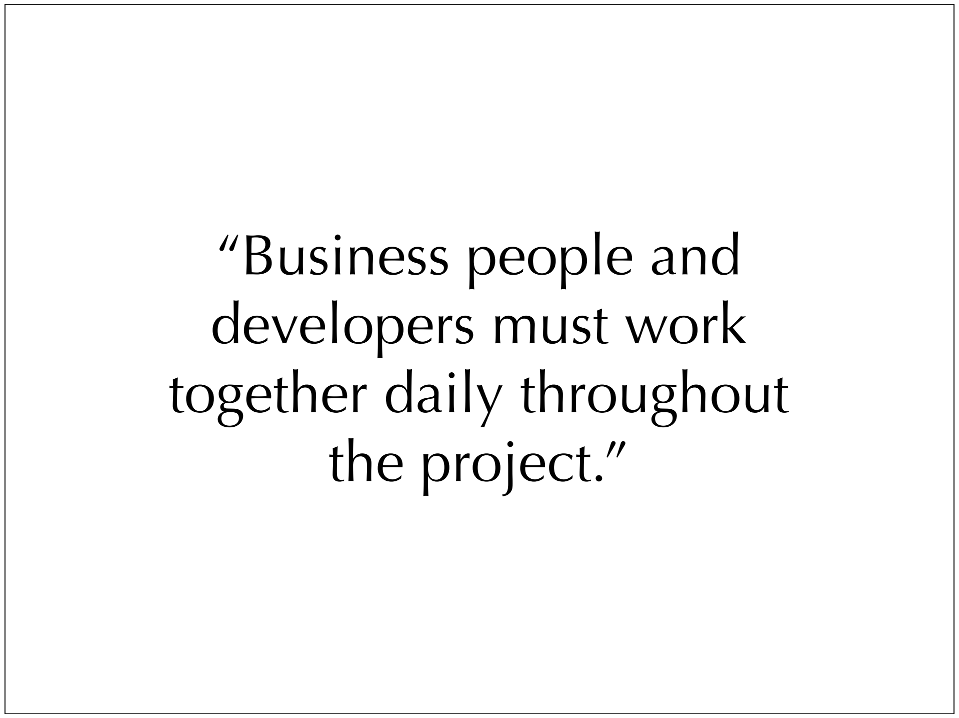
... harness change to the customer's advantage.

a principle of agile software development

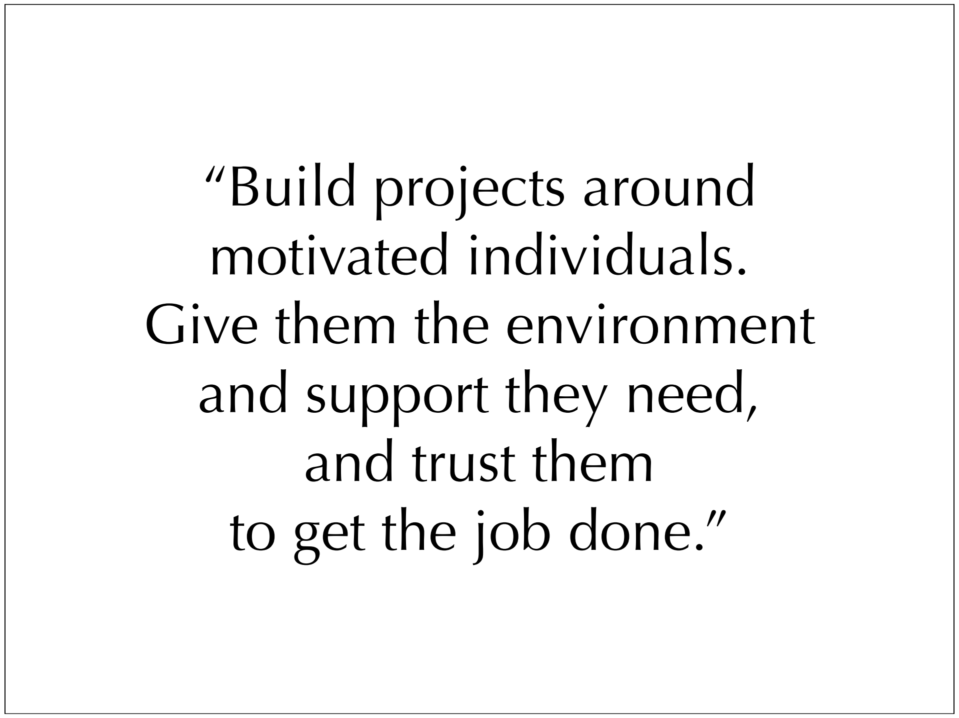


“Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.”

a principle of agile software development


“Business people and developers must work together daily throughout the project.”

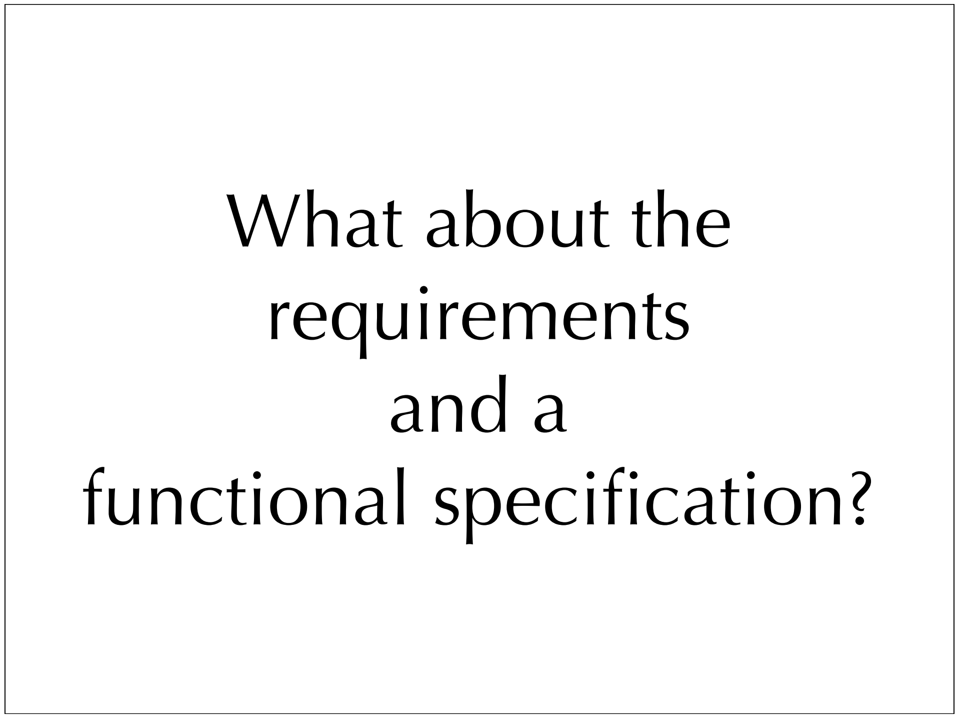
a principle of agile software development



“Build projects around
motivated individuals.
Give them the environment
and support they need,
and trust them
to get the job done.”

... a principle of agile software development

There are half dozen more as part of the manifesto.

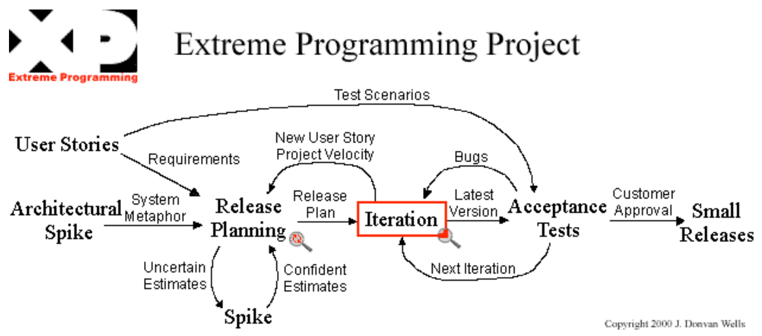


What about the requirements and a functional specification?

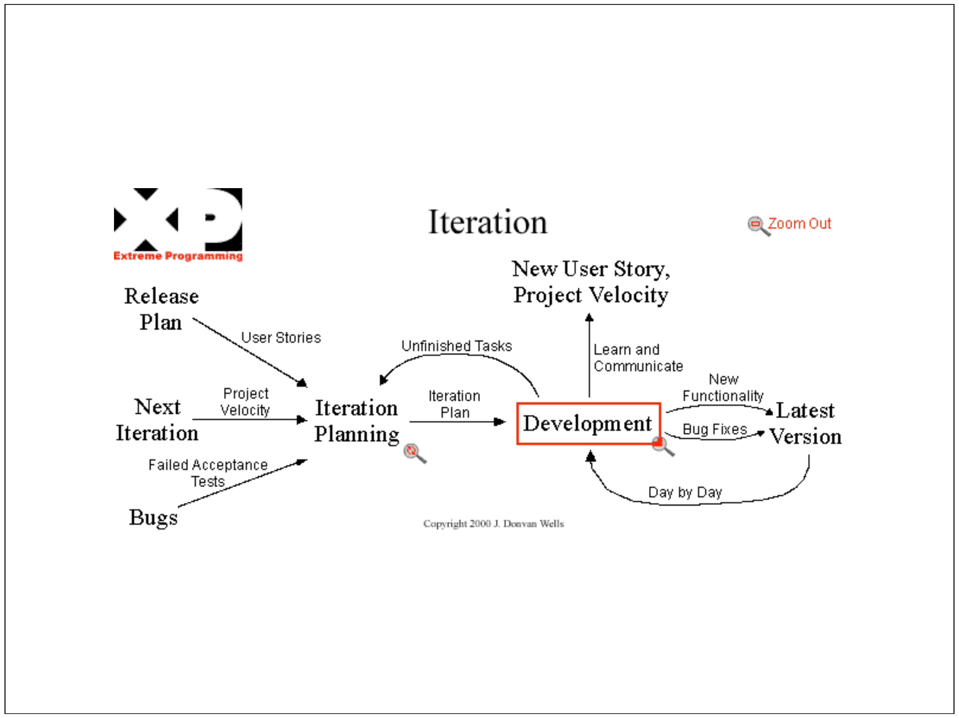
List of requirements, broken into small “stories”.
Each story can be implemented quickly — a day, a few days, a week, ...
Prioritize based on value and cost (time).
Implement features. Deliver working system. Get feedback. Repeat.

short iterations

... implementation and delivery on very short cycles.



Courtesy of http://www.extremeprogramming.org/



Courtesy of http://www.extremeprogramming.org/

On to the project...

Team assignments.
Talk to the two uncertain teams.