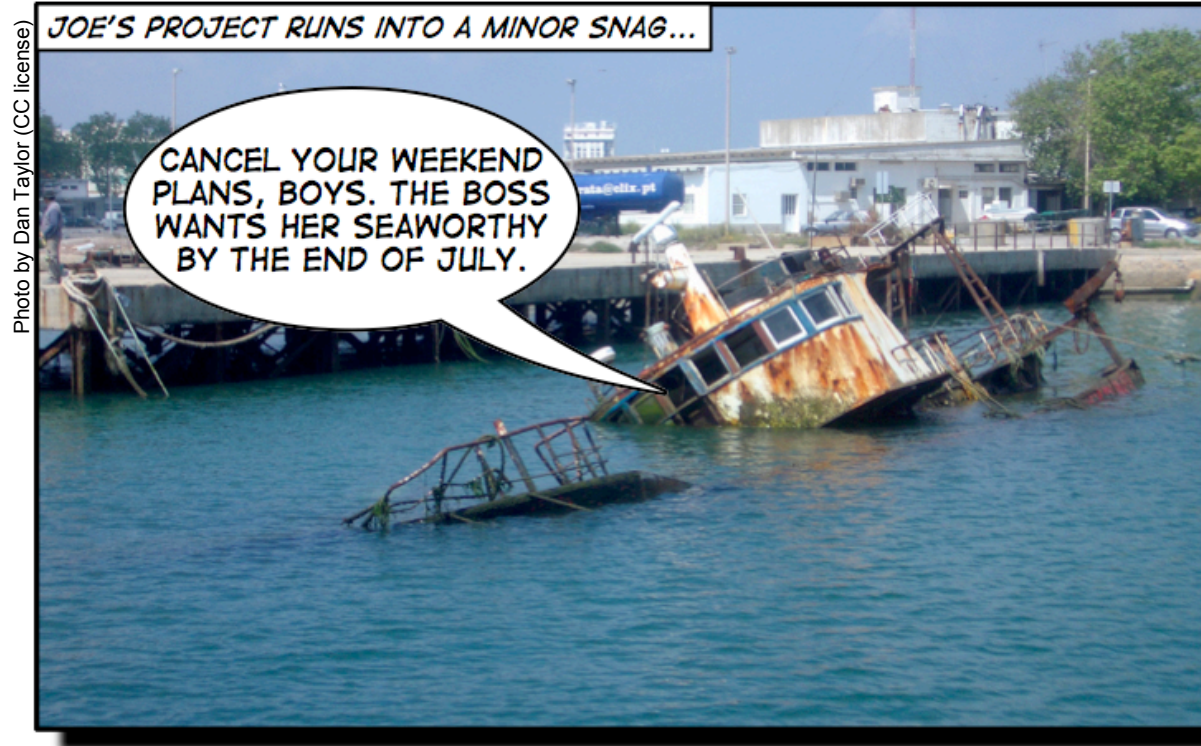


Why Projects Fail



...and what you can do about it.

A presentation by Jenny and Andrew

Who we are...

Andrew started programming in the 80s, and lost count of how many languages he's worked with.

He's led teams of programmers, requirements analysts and process engineers.

Jenny's spent the last 15 years or so working in software quality

She's currently running a large distributed development team for a global media company



Jenny and Andrew truly believe that with better development practices and good programming habits, we can all build better software.

Before we forget... are there any Microsoft C# people in the audience?

Our current project is **Head First C#**, and we're looking for another technical reviewer. If you're a hardcore C# guru and want to help make a Head First book better, please talk to us after we're done.

introducing objects

Mike gets an idea

The interview went great! But the traffic jam this morning got Mike thinking about how he could improve his navigator.

It'd be great if I could compare a few routes and figure out which is fastest...

He could create three different Navigator classes...

Mike could copy the Navigator class code and paste it into two more classes. Then his program could store three routes at once.

```
Navigator
SetDestination()
ModifyRouteToAvoid()
ModifyRouteToInclude()
GetRoute()
GetTimeToDestination()
TotalDistance()

Navigator2
SetDestination()
ModifyRouteToAvoid()
ModifyRouteToInclude()
GetRoute()
GetTimeToDestination()
TotalDistance()

Navigator3
SetDestination()
ModifyRouteToAvoid()
ModifyRouteToInclude()
GetRoute()
GetTimeToDestination()
TotalDistance()
```

Whoa, that can't be right! What if I want to change a method? Then I need to go back and fix it in three places.

Right! Maintaining three copies of the same code is really messy. A lot of problems you need to solve need a way to represent one thing a bunch of different times. In this case, it's a bunch of routes. But it could be a bunch of turbines, or dogs, or music files, or anything routes. But all of those programs have one thing in common: they always need to treat the same kind of thing in the same way, no matter how many of the thing they're dealing with.

Mike can use objects to solve his problem

Objects are C#'s tool that you use to work with a bunch of similar things. Mike can use objects to program his Navigator class just once, but use it as many times as he wants in a program.

This is the Navigator class in Mike's program

```
Navigator
SetDestination()
ModifyRouteToAvoid()
ModifyRouteToInclude()
GetRoute()
GetTimeToDestination()
TotalDistance()
```

navigator1 3.5 miles
Navigator object

navigator2 3.8 miles
Navigator object

navigator3 4.2 miles
Navigator object

Mike needed to compare three different routes at once, so he used three Navigator objects at the same time.

All you need to create an object is the new keyword and the name of a class.

```
Navigator navigator1 = new Navigator();
navigator1.SetDestination("Fifth Ave & Penn Ave");
string route;
route = navigator1.GetRoute();
```

Now you can use the object! When you create an object from a class, that object has all of the methods from that class.

you are here > 7

So why do projects fail?

Good question.



If you can recognize a failing project ***before*** it crashes and burns, you can usually save it.

“This time it’s different...”

There’s an old saying about how there are a million ways to fail, but only one way to be right. When it comes to projects, nothing’s further from the truth. Projects fail the same few ways over and over again.

Don’t go in the basement!

Software projects are a lot like cheesy horror movies. After you’ve seen a few of them, you know that the first guy to leave the group is going to get an axe in his head. Projects are the same way. People keep making the same mistakes over and over, and it keeps getting their projects killed.

You know you're on a failed project when...

A judge in 1964 said, "I don't know how to define pornography, but I know it when I see it." And the same goes for failing projects - we all know when we're on one that's sinking.

What does a failing project look like?

You know your project failed if it got aborted and everyone was laid off. But there are other, less obvious kinds of failure:

- ★ The project costs a lot more than it should.
- ★ It takes a lot longer than anyone expected.
- ★ The product doesn't do what it was supposed to.
- ★ Nobody is happy about it.

People hate the word “failure”.

Nobody sets out to fail. Most projects start with a good idea and talented people. (Not all, but most.) But talking about failure makes people uncomfortable, because nobody wants to give or take that kind of criticism.

A show of hands, please...

We’ve never met a single professional software engineer with more than a few years of experience who hasn’t been on at least one failed project.

Are there any here?

Four basic ways projects can fail

There are plenty of ways that you can categorize failed projects. We like to think of them like this:

★ **Things the boss does**

Ways your management can screw up your project for you

★ **Things the software does (or doesn't do)**

How your project doesn't quite meet the needs of the people you built it for

★ **Things the team should've done**

Yes, sometimes we do mess it up too

★ **Things that could have been caught**

...but weren't until it was way too late.

Things the boss does

Let's face it... a lot of project problems start at the top.

- ★ Tunnel vision
- ★ Over-reliance on gut instincts
- ★ Repeated false starts
- ★ Mid-course corrections
- ★ An artificial “wall” that the business puts up to disconnect from the engineering team



Things the software does (or doesn't do)

It seems pretty obvious that you should know what the software's supposed to do *before* you start building it... not that that stops us.

- ★ We only find serious problems after we've built them into the software
- ★ We have big, useless meetings that fail to figure out what the software's supposed to do
- ★ Scope creep
- ★ 90% done, with 90% left to go.



Things the team should've done

The team could have done the work more efficiently, if only we'd taken the time to think it through.

- ★ Padded estimates compensate for unknowns.
- ★ Project teams will just pick a deadline and stick to it, no matter what basic reason and common sense tell them.
- ★ Somehow non-programming tasks always seem to get cut when the deadline gets closer.
- ★ Misunderstood predecessors lead to cascading delays.



Things that could have been caught

Which would you choose: a well-built program that doesn't do what you need or a crappy one that's irritating to use and does?

- ★ Getting a few tech support people to “bang on the software” is **not** testing.
- ★ Maybe we could've caught that design problem *before* the code was built.
- ★ Maybe we could've caught that code problem *before* we went to test.
- ★ “Beta” does not mean “use at your own risk.”



What you can do about it

Some easy ways to make sure your project doesn't fail:

- ★ Tell the truth all the time
- ★ Trust your team
- ★ Review everything, test everything
- ★ All developers are created equal
- ★ The fastest way through the project is the right way through the project

The talent is there... the engineering's not.

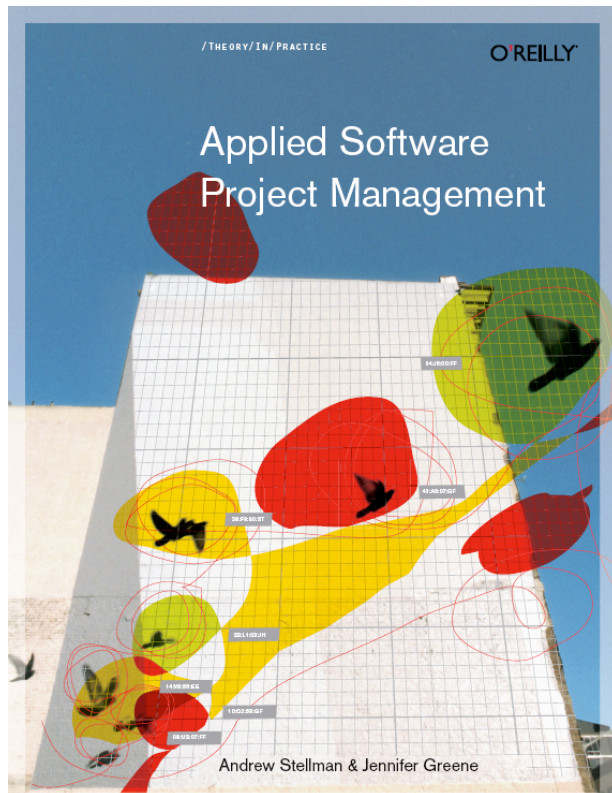
Hoover Dam was finished two years early, and under budget. Software's not so different that we can't engineer it just as well.

- ★ Our problems have, for the most part, been solved.
- ★ Over and over and over again. Seriously.
- ★ We just have to stop ignoring the solutions.
- ★ Also, hire awesome consultants who know what they're doing and have solved these problems before.
- ★ (us.)

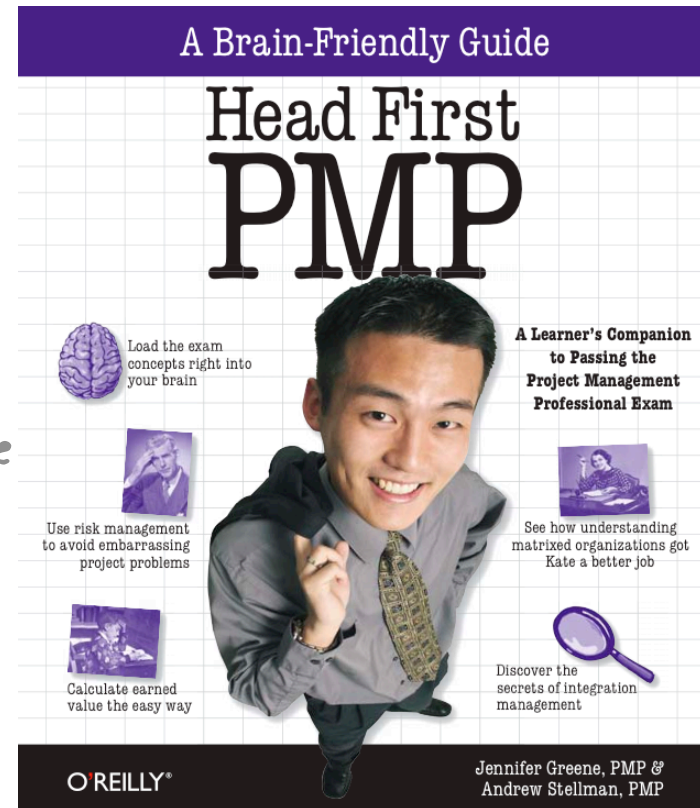
Do you think your project is more complicated than this one?



One last quick note from the O'Reilly marketing department



Buy these books



And check out our blog, "Building Better Software"

<http://www.stellman-greene.com/>