Software Rescue Projects

By Eric Davis | www.LittleStreamSoftware.com

TABLE OF CONTENTS

INTRODUCTION

INTRODUCTION

So you've heard that there are Software Rescue projects. Perhaps you're wondering if you have one, but you're not sure.

A Software Rescue project is a software project that is going bad. It's not a failure yet but somewhere, somehow it went off the rails and now it's heading down the wrong path.

There are hundreds of reasons that lead to a Software Rescue project, but some of the most common are:

> unclear requirements
> inexperienced development team
> inexperienced management team
> changes too rapidly
> unsustainable development pace
> missing development practices and discipline

INTRODUCTION

The cause could be one, all, or some other combination of factors. Generally it’s not caused by only one factor, but a mix.

Still, there is hope.

The difference between a rescue project and a failed project is that the rescue project can still be saved – rescued.

HOW TO RESCUE A PROJECT

Fixing a rescue project is possible, though it will be hard work. Depending on the factors that caused the problems, it might not even be practical to fix it. This is especially true if the cause is outside of your control and you can't change it (e.g. industry changes, powerful stakeholder).

You can take a systematic approach to try and diagnose the problem(s) and fix them though.

STOP DIGGING

STOP DIGGING

The first thing you need to do is to prevent things from getting worse.

When you're digging a hole and discover that you shouldn't be, the first thing you do is to stop digging. Not change the angle of your shovel. Not slow down. Not buy a new shovel with some fancy technology. Just stop digging.

Stopping things from getting worse will take different tactics based on what the cause is. For example if the list of new features are growing too fast, freezing features and stopping new feature development would stop it. If the problem is that new developers are joining the project, then stopping new hires would help.

You don't necessarily have to stop all development either. If you know the problem is with new features, you can stop them but still let the team work on bugs or other unaffected areas.

STOP DIGGING

Keep in mind that when it comes to really desperate projects, stopping all development can be the prescription that’s needed. Even if you “lose” time and resources, if you were heading in the wrong direction in the first place then stopping would still put you ahead.

TEAM PROBLEMS

TEAM PROBLEMS

The biggest factor in a software project is the team. There are usually multiple teams involved, even informally.

There is a development team who has developers, designers, testers, and maybe even a writer. These people are involved in the actual production of the software.

Then there is the management team: the project manager, product manager, software architect, and business stakeholders.

People can cross between these teams. A developer might be involved in the management team by filling the role of the lead architect, or maybe the project manager is also doing testing.

Due to the dynamic nature of people, problems with the team and team members are frequently a cause of a rescue project.

As Gerald Weinberg said:

TEAM PROBLEMS

No matter how it looks at first, it's always a people problem.

Problems in the team will be the most difficult to solve, but they will also have the highest payoff.

The best place to start is to talk to the people and see what they think is wrong. Do it 1-on-1 and take your time so they have a chance to fully express themselves. You might not get the exact answers you need, but you can get an idea of the root cause for the team's problems.

SOFTWARE PROBLEMS

SOFTWARE PROBLEMS

Another type of problem for rescue projects comes from the software itself.

Maybe the technical challenges are too big, or maybe the technology choices made for the project are too risky and cutting edge.

Technology problems can be a red herring though. They can easily cover up the real problem because it's always easier to blame technology instead of a person or your business. But sometimes, there actually publishing.gemspec a technology problem.

To solve a technology problem you're going to have to do 1-on-1s with your team as well as group reviews. Look for clues about specific components, processes, or tools that come up again and again. Chances are you'll find two or three that are causing the majority of your problems.

SOFTWARE PROBLEMS

Sometimes, swapping out a technology for another will solve the problem, but don’t jump to that conclusion until you’ve looked at everything else first, otherwise the problem could reappear.

EXPERIENCE PROBLEMS

EXPERIENCE PROBLEMS

The third major type of problem that causes rescue projects is experience.

This is the intersection where people and software meet. It’s also a ripe environment where problems can incubate and grow.

With the right people, even poor technology choices can be successful.

Even with the wrong people, great technology choices can be successful.

But if the wrong people are matched up with the wrong technology, you have a big problem.

This works because the right people have the experience and knowledge to work around poor technology. They’ll be slower, more annoyed and perform worse than usual, but they will overcome the technology.

EXPERIENCE PROBLEMS

On the other hand, when the wrong people are given great technology they can usually make it work. It won't be as fast, it will be more expensive and harder to maintain than if the right people were used. But it will work.

This problem comes up when the wrong people are using the wrong technology. This happens because those people are out of their experience (desktop developer doing web development), their experience is incorrect, or they have hidden their inexperience.

This is a hard type of problem to solve. And you'll never be quite sure if you solved it completely or not.

HAVE HOPE

HAVE HOPE

At the end of the day, there is one thing that you should remember.

With every software project comes the hope that it can be improved.

It might be expensive. It might be difficult. It might require outside help and perspective. It might even require some significant changes to your organization. But that potential is there.

Good luck.

Eric Davis runs Little Stream Software, a one-man Ruby on Rails and web development consultancy. Focusing on software and marketing companies, it has developed (and rescued) various software projects over the years. For more information and advice about how your project can be rescued, contact edavis@littlestreamsoftware.com.