8 Common Types of Rails Rescue Projects

By Eric Davis | www.LittleStreamSoftware.com

TABLE OF CONTENTS

TABLE OF CONTENTS

INTRODUCTION

INTRODUCTION

Just like Rails projects take many forms, a Rails Rescue project does too.

A Rails Rescue project is basically a Rails project that has gotten into trouble. This might be temporary, permanent, caused by external factors, internal factors and so on.

STOP FURTHER DAMAGE

STOP FURTHER DAMAGE

While identifying the root cause of the problem may prove useful, I've found that focusing on the present is more productive. Knowing the cause might help you make better decisions now to counter the cause, but not many rescues have the luxury of looking back and analyzing.

The priority to stop the damage and start repairing. This requires focus on the now and problems that affect the project today.

Fortunately there are several common types of Rails rescue projects. This makes it easier to diagnose the problem start repairing..

One thing to remember: no project can be strictly defined by one type. Just like there is no "normal person", software projects are diverse and varied. This means your project will exhibit signs of multiple rescue types.

STOP FURTHER DAMAGE

This fluidity of software also means that your project may shift from one type to another as it goes. Some of the most interesting rescue projects start with one dominant type and as they are fixed, another dominant type emerges.

ONE RULE

ONE RULE

Whatever the type, there is one over-arching rule I use to fix Rails projects in trouble

> Fix the most important thing first, then the next most important thing.

Follow this advice and your project will lean to the right, then to the left, then forward, and finally stand on its own.

Here are the different types:

1. PERFORMANCE PROBLEMS

1. PERFORMANCE PROBLEMS

The performance type is straightforward.

What used to work great in development and under basic testing, starts to fail with real traffic levels. Maybe your application becaome more popular than you thought (congratulations!) or maybe it's just hitting some performance stress points that you can't solve.

Other times the performance has been decreasing little-by-little over the past months or years. You can't pinpoint one thing that slowed everything down, but it's visible to your team and your users too.

Solution: Start with a complete performance audit and look for the areas with the largest negative impact on performance. Start an optimization process there. Repeat the complete audit and continue. Don't forget to audit the environment your ap-

1. PERFORMANCE PROBLEMS

plication runs in too (e.g. Ruby version, server setup, browsers, etc.).

2. DEVELOPMENT TEAM TOO SMALL

2. DEVELOPMENT TEAM TOO SMALL

Great applications can be built with small development teams, even by just a single developer. But if there is a single problem with small teams is that they can lose perspective easily. This makes innovation difficult as well as amplifies any problem outside of your small team's experience.

There's a common saying in software:

> when all you have is a hammer, then everything looks like a nail

Solution: Perspective is the key to fixing this type of rescue. The long-term solution is to expose your team to more ideas and experiences about Rails development. However,that might not be possible in the short run, so you'll need to hire someone with different experiences to guide and advise you.

3. STABILITY PROBLEMS

3. STABILITY PROBLEMS

Rails applications aren't supposed to wake you up every night at 3am to reboot the servers. They were meant to run and run and run.

Instability in your application or infrastructure prevents you from helping your users. Over time, this will frustrate them, and it will surely affect your business. Remember: unhappy users today become ex-users tomorrow.

This rescue is closely related to the Performance type. Many times an area with poor performance causes the application to become backed up, which cycles until it crashes.

Solution: Performing a system-wide stability and code audit will let you find which areas are contributing to the instability. Once found, a variety of techniques can be used to stabilize that area, like in-depth testing, defense code, re-architecture and proactive notifications.

4. COMMUNICATION PROBLEMS

4. COMMUNICATION PROBLEMS

One of the most painful situations occurs when the communication within a team breaks down. The code will still get written and work will still get done. Or at least for a little while…

Eventually, the application development will come to a grinding halt.

Every communication problem with a team is like an infected cut. Treat it quickly and you will never know it happened. But let it fester and you could lose a limb, or worse.

The worst about communication problems? They sneak up on you. You can't see them unless you actively look for them.

Communication problems can also be the root cause of many of other rescue types.

Solution: This type is hardest to fix. You need to have frank, honest conversations with the entire team in order to find out

what's really going on. The solution will vary depending on what's causing the communication problem.

5. THE OUTSOURCED PROJECT

5. THE OUTSOURCED PROJECT

Working with people worldwide is great, especially when you can get your application built quickly and inexpensively.

But six months later that inexpensive application might start showing its warts. Features that were supposed to be working don't work anymore. Bugs that were squashed have come back with a grudge. All of the recent development has been slower than before.

The Outsourced Project combines other rescue types but with the additional problem of not understanding the application and development team themselves.

Solution: With outsourced projects that you've taken back, you can start fixing the problems. For ones that are still outsourced, you'll have to work with the outsource company to get them to correct the problems. One option you have is to hire an impartial, third-party to evaluate and steer the project for you.

6. THE BIG PORT

6. THE BIG PORT

"Remember, no one is to touch that old Windows NT server! The one with The Application that the business relies on. That was produced by a company that has since folded."

Your team has tried to rewrite and port it to Rails, but every time you try to something gets in the way.

Rescues from Big Port are much more common than people realize. Many applications are rewritten and updated by using Rails. This problem occurs when the new application doesn't work "the same way" and when new changes are made during its development. What started as just copying the old application, turns into developing something new but also the same.

Many of the problems stem from communication problems between the development team and the stakeholders (e.g. users, executives, champions).

6. THE BIG PORT

Solution: Depending on the source of the problem, a stronger development process can improve the communication about the porting. This can (re)set realistic expectations about the project. Other times the size of the project was underestimated and now progress has slowed down.

7. PROTOTYPE IN PRODUCTION

7. PROTOTYPE IN PRODUCTION

The first prototype version of your application was great. Your users loved it, it worked, and it let your business get off the ground. But when it was created, it wasn't ever supposed to last this long. Temporary fixes became permanent. Band-aids and scaffolding were never replaced.

Now is the time to go back and start correcting these mistakes.

Very common in move-fast startups but also with established companies too, the prototype in production problem is caused by moving faster than the development process can handle. Instead of working the way everyone agrees on, the fastest option is chosen.

This can be useful when speed is important in the short-term, but it comes with a cost later on.

Solution: Either the development process needs to be changed so the team can continue to move fast but without problems,

7. PROTOTYPE IN PRODUCTION

or the team needs to slow down and follow the process they created. In either case, time to clean up the mess from the prototype will need to be scheduled.

8. THE STARTUP

8. THE STARTUP

You weren't sure if your idea would work. Even with some early validation, you were always wondering if this application would build your business.

You were right, but now it's time for the next version. It's time to clean away the cruft and failed experiments so you can grow your business to the next level.

The Startup is similar to The Prototype in Production but in this case, the entire business is moving too fast.

At early stages this might be okay, but the development and business costs will have to be paid back in the future.

Solution: Once again, either the development process needs to be changed so the team can continue to move fast with the business, or the team needs to slow down and follow the process they created. This might mean the business has left its high growth phase and is now starting to stabilize.

IDENTIFY, STOP, REPAIR

IDENTIFY, STOP, REPAIR

These eight rescue project types cover the most common cases. There are still other types out there, and each one of them can mix with others to create a fragrant bouquet of problems for your project.

Acknowledging that your project **might** have a problem is the first step. Even if the problem is small or imagined, thinking critically about how the project is running is a sign of a mature leader.

Getting a second opinion on your project is a great next step. You'd want the opinion of someone who has seen many rescue projects and is impartial enough to give you the critical advice you need. When you're ready for that advice, the Rails Rescue Review is the perfect place to start.

Eric Davis

Little Stream Software