The Survival Guide for Brand New ScrumMasters

By Dave Rooney

The Survival Guide for Brand New ScrumMasters

Dave Rooney

This book is for sale at
http://leanpub.com/scrummastersurvivalguide

This version was published on 2014-02-07

Tweet This Book!

Please help Dave Rooney by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just bought The Survival Guide for Brand New ScrumMasters!

The suggested hashtag for this book is #scrummastersurvivalguide.

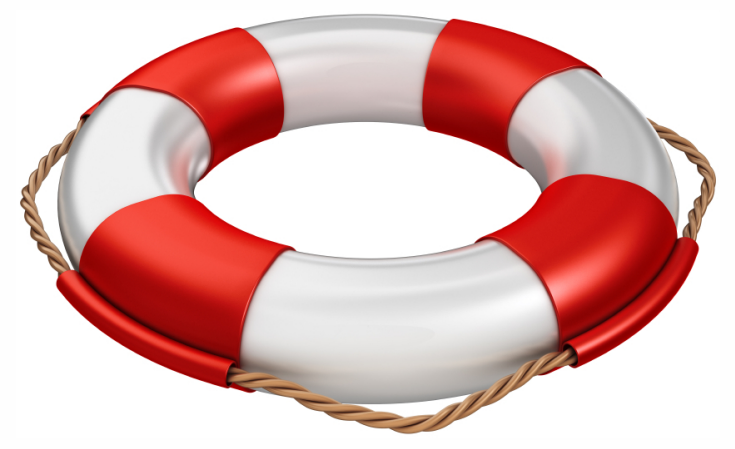
Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#scrummastersurvivalguide

For all ScrumMasters thrown into the deep end of the pool after a few days of training!

Contents

Introduction



Welcome! Perhaps you've been a team manager, project manager, or possibly a business analyst. You may be a tester or even a software developer. Maybe you've been through a good number of projects and have experienced the highs of success and fallen in a few potholes along the way.

Well, your team or organization has decided that they want to "go Agile" and you've been volun-told to be the ScrumMaster. Congratulations! You've had a few days of training, but now what?

There are plenty of books out there about Scrum and Agile Software Development in general, and some on Agile Coaching

in particular. The coaching books are excellent - I've read them - and there are dozens of books to read to educate yourself on Agile/Scrum/Lean in general, or on specific topics such as Estimation and Planning, Organizational Issues, Technical Practices, etc. You could literally spend months doing nothing but reading. However, you don't have months… you don't have weeks… in fact, you're lucky if you have days! So what can you do?

Well, this book is somewhat different from the others. It won't delve into the minutiae of the Agile values & principles or any particular Agile practices unless warranted by the example at hand. What the examples will do, though, is distil specific issues that commonly occur while working with Agile teams into small lessons that you can easily read and applied in a short period of time.

These examples use the metaphor of our progression through the stages of life to provide a backdrop for a team's journey towards Agility, and your journey as a parent… er, I mean ScrumMaster. Anecdotes provide the context for a particular point, and concrete examples give you strategies for working through those situations, and even entire stages in a team's "life".

Life Stages

A team's progression during a transition to an Agile Software Development approach can be roughly equated to the following stages of a person's life:

- Infant
- Toddler

- Preschooler
- Elementary School
- Adolescent
- Teenager
- Young Adult
- Mature Adult

It's not a perfect model, but that's the beauty of models - none of them are perfect. That's why they're only models! We're going to examine each of these stages through anecdotes and advice as a team progresses from their nascent beginnings to full maturity.

So, again, welcome to your journey into the world of a Scrum-Master!

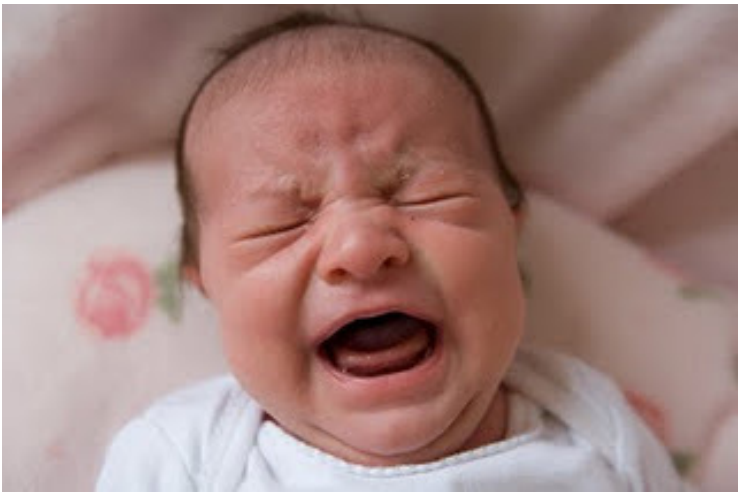
How to Read This Book

Each anecdote tells the story about an issue, and follows with the "Coaching Point" about how that issue is associated with what you and the teams you work with face. Each anecdote also has "Key Points" summarizing what was learned.

For a quick reference of issues faced by teams and the anecdotes associated with them, you can jump to Appendix A.

1 Sample Stories

Patience, Persistence… and Ear Plugs

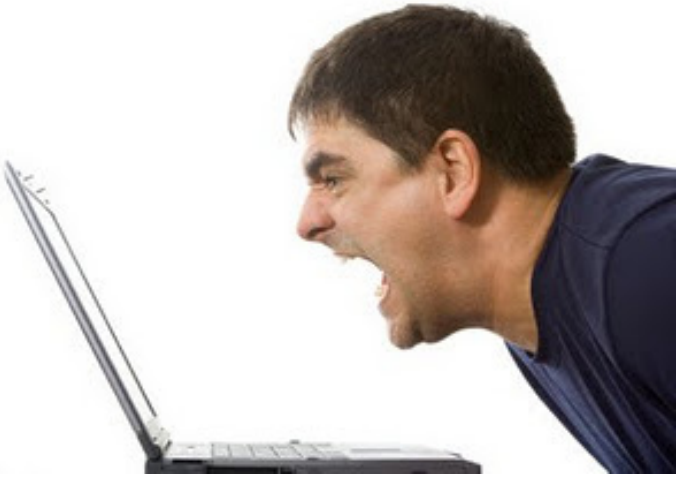


For the first couple of months after my son was born, we had a heck of a time getting him to sleep in the evening. After a couple of months I realized that I just had to let him scream until he fell asleep. Once he did he slept like, well, a baby!

Getting to that realization was tough. He was literally screaming as loud as he could into my ear (I've been tested, and the hearing in that ear is diminished!), and he would wriggle away. I figure that he was not impressed with having to go to bed! Regardless,

the screaming became a routine, and once that routine was established it wasn't as stressful. After a few months the patience and persistence paid off. He screamed less and less, eventually going down to sleep without a fight.

Coaching Point



When a team is first starting their transition to Agile, you will hear a lot of screaming… mostly in the figurative sense, but sometimes literally.

For most of the organizations with whom I've worked, Agile represents a fundamental change to how they think about their organization and their work. Any change that significant will create fear and stress. People will think that they may lose their job. There will be those who lose what they believed to be a

prestigious title. Sacred cows may be slain in the name of moving to Agile.

Be prepared to put the team on your shoulder (figuratively, of course) and let them cry it out. Give them time to learn how to work in short cycles. Give them the support and time needed to learn test automation. Give them the support and tools needed to effectively inspect and adapt. When the team has gripes & complaints and wants to give up, listen patiently and with compassion.

If that support is available to them, over time their need to "cry it out" will diminish and eventually disappear.

Key Points

- Scrum and Agile can introduce significant disruption to the status quo.
- Have some compassion for the people on the teams being affected by these changes.
- Remind everyone that work ***will*** get better as they learn!

Playing in the Mud

Many parents, at times this one included, become obsessed with ensuring that your kids are clean. We do whatever we can to prevent them from getting dirty - cover our kids with clothes

that prevent anything dirty from touching their skin, keep them out of mud and dirt, have a constant supply of hand sanitizer nearby, hose them off if they do manage to get something on them… and they still manage to find ways to get dirt on them!!

When they get food on their faces while eating we chastise them and wipe it away. When a child encounters a mud puddle, their first reaction is to jump into it… a parent's is to react in horror and haul the kid away to be cleaned.

We worry about our children playing in public sandboxes because we don't know who or what has been playing in there before them!

> ***IT'S A DIRTY WORLD, AND WE HAVE TO PROTECT OUR CHILDREN FROM IT!!***

Or do we?

Coaching Point

Put simply, humans need to be dirty. We literally need to have bacteria on our skin in order to be healthy[1]. All of the handwashing and sanitizer we foist upon our children is meant well, but in reality we're actually doing as much long-term harm as we are short-term good[2]. The fact is that we as a species evolved in a dirty environment - we didn't have soap or hand sanitizer 50,000 years ago!

[1] http://ucsdnews.ucsd.edu/newsrel/health/11-09Gallo.asp

[2] http://www.dispatch.com/live/content/local_news/stories/2010/01/17/tooclean.ART_ART_01-17-10_A1_0UGATN1.html

The same type of side effects occur when a software development team doesn't work in an environment where they're allowed to *get dirty*. They aren't allowed to experiment. They aren't allowed to fail in small ways that allow the team to learn as they move along. Everything has to be perfect the first time, or it's considered a complete failure.

So, how well does that approach work? What are the side effects of sanitizing the development environment to remove all the dirt? Innovation is stifled. Code becomes riddled with Technical Debt because there is fear of making the changes required to keep it clean. Eventually, it costs an increasing amount of time and thus money to maintain and extend the code and ultimately a complete rewrite is required.

Wouldn't it be less expensive and much more effective to allow teams to play in the dirt and find ways to improve the software in smaller ways over time? Yes, there will be occasions where the small experiments won't work. Yes, there is a risk that some of the small changes will break other code. But, like the long term effect of allowing a child's immune system to be exposed to staph bacteria early in life, the rewards of keeping the code fresh and clean enough far outweigh these risks.

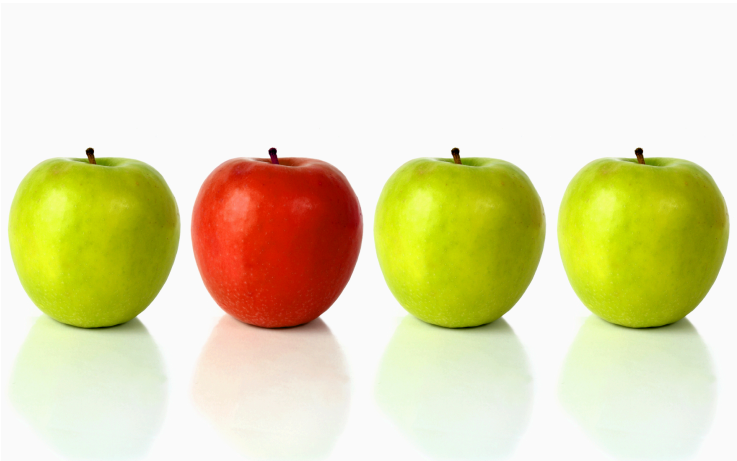
So, paradoxically, if people are empowered to get dirty, cleaner code, a better product and a more productive team will result.

Key Points

- Small failures and mistakes are where true learning takes place.
- Create a safe to fail environment, rather than a fail-

safe one.
- Encourage small experiments that keep the team fresh and engaged in their work.

You're Not That Special!



I'm the 4th of 5 kids, and thus had 3 siblings precede me through teenagehood. I was actually pretty good to my parents during those years, although I did have one party when I was in my first year of high school. My parents went away overnight, and my younger sister stayed at my Grandparents' home. I had the place to myself, so naturally I wanted to have a party! So, I did.

Sorry to disappoint you, but this isn't a case where 300 people

showed up, totally destroyed the home and I made the local, regional and even national news. There were about 15-20 people who came, and if there was any drinking it was done outside. Nothing was broken, no police were called, and as I recall we all had fun.

I cleaned up the house the next day, thinking that I had completely eradicated all evidence of the party. When Mom & Dad came home late in the afternoon, Mom went through the living room, into the kitchen and then asked me how the party went. I was shocked! How could she have known?!

Well, I'm the 4th of 5 kids and thus had 3 siblings precede me through teenagehood! She did compliment me on the cleaning job, though.

Coaching Point

Every organization I've worked with in my career, even before this whole Agile thing, believed they were special. They were different from everyone else. In a sense that's quite true - each organization has different people, therefore they are different. However, from a software delivery process perspective they really aren't.

I'm willing to concede that there are two fundamental types of software organization - software product companies, and internal IT or software development groups. The former is in the business of developing software products, or their software is part of a larger system. In the latter, the focus of the organization is not the delivery of software - the software supports the core functions of the business. Beyond that, though, I have witnessed

striking similarities in organizations in both the public and private sectors, small and large.

So, when you're told that a team can't possibly do Continuous Integration because of their company's policy for committing to source control, ask why they think that their company is different. If they tell you that they have to create a high-level design that must be submitted to and approved by an architecture group, ask why they think their company is different. If they tell you that they can't possibly ship a meaningful increment of software every few months, ask why they think their company is different.

You will hear a myriad of reasons. Almost none of them are valid. Organizations of similar size and function have been doing these things since long before the term Agile was coined. Even more are doing it now.

The people are special. How those people deliver software is not.

Key Points

- Every organization thinks that they're different (hint: really, they aren't!)
- Dig deep for the reasons behind excuses for not following Scrum or Agile practices!
- There are plenty of examples of groups and organizations from almost all domains using Agile approaches to deliver software - Google is your friend!