

A decorative graphic on the right side of the slide. It features three sets of concentric circles in shades of blue. Two sets are in the upper right, and one is in the lower right. Thin blue lines extend from the top left towards the circles, and another line extends from the top right towards the bottom right circle.

# Five Why's Root Cause Analysis with Ishikawa's Fishbone Diagram

Making the Hurt Go Away

Root Cause Analysis basics for technical managers – without fluff, mumbo-jumbo, or vigorous self-love.

**David M. Russell**  
**12/14/2008**

Things go wrong... that's just a fact of life. To paraphrase the infamous bumper sticker, "Stuff" happens. In the world of software projects, that stuff consists of bugs (software defects), bad/unclear requirements, missed delivery dates, inappropriate expectations and so forth. And when those things go wrong, all too frequently, the symptoms end up getting treated, only to return in the future when the real problem manifests itself again.

To make matters worse, hands-on IT managers (like me) can end up getting so mired in the details of code reviews, project plans, and production-related issues that it's very easy to miss out on easy fixes to the big problems hiding in your shop while distracted by minutiae. Our training in logic makes us dig, dig, dig to get to the solutions – we look at the subnet, then the router, then the individual computer; we analyze the error message, then the class, then the line of code.

But there's good news: Any application of Root Cause Analysis (RCA, as we insiders call it – you're one too, now) techniques can help tremendously in making these big problems go away – even if you're a novice at the craft.

So, how does someone with a penchant for perfectionism pull back far enough to see the big picture? Here's the thought process I use to determine the most appropriate parts of the process to optimize through Lean Process Improvement techniques:

If you're utilizing statistical process control in your software development environment, you're running a shop in the top percentile or two and probably have this material down cold. This paper is for those who are running shops that haven't quite made it to that point. Maybe your company hasn't invested in the analysis necessary for proper metrics. Perhaps your entire management team consists of ex-

developers and other technical staff who have never had formal exposure to these concepts. Maybe everyone's on board and wants to do it, but the fear of high up-front costs keep everyone paralyzed.

Well, without having to dig into control charts, histograms, and variance measured in standard deviations from the mean,

here are a few quick ways to think about the problems in your world and to determine whether something is worthy of Root Cause Analysis.

***Good news: Any application of Root Cause Analysis techniques can help tremendously in making big problems go away – even if you're a novice.***

### 3 Easy Ways to Tell When to Use Root Cause Analysis:

1. An event keeps happening again and again (eg: long test cycles, high volume of defects for each release)
2. Related events appear to happen regularly (eg: learning of new requirements while the product is under test, missed project delivery dates)

3. Certain infrequent events are very impactful (eg: people with a lot of knowledge quit, source control needs to be restored from last month's tape)

See, you don't have to implement fancy systems, train your staff, or collect metrics for a year (hoping people don't skew the numbers by gaming the system) in order to find opportunities to save a bundle. I'm sure a few minutes of moderate concentration will generate quite a few ideas of what might be worthy of investigation.

Armed with those ideas, here are a couple of points to consider when determining what's ripe for Root Cause Analysis.

### When is Root Cause Analysis the **WRONG** answer?

- **Isolated conditions:** Some problems are so isolated, even though expensive, that the best way to handle the event is preparation rather than resolution. It makes sense to determine this as quickly as possible, rather than waste a great deal of time digging in. For example, a hurricane knocks out your data center. Does it make sense to dig into the root cause of the hurricane, or to just do something to minimize your losses?
- **Unrelated events:** Correlation does not imply causation. This fallacy is so timeless it has been recorded in Latin by ancient scholars: "*Cum Hoc Ergo Propter Hoc.*" Sometimes, managers can get a little

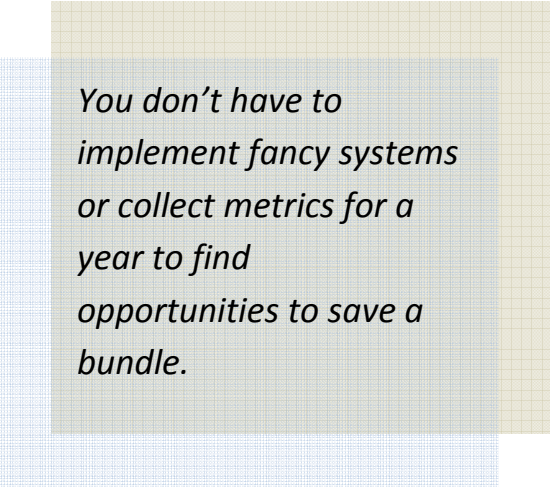
ambitious when learning a new skill.

[Maslow's Golden Hammer](#), "*When all you have is a hammer, every problem begins to look like a nail,*" can result in unnecessary investigation into causality for things which have very little in common other than timing.

- **Not a costly problem:** When fixing the problem costs more than the problem itself, why try to fix it? This is not meant to serve as an excuse to ignore huge groups of related small things, but rather to ensure that the RCA process doesn't fall flat on its

face because it's too costly to kick off or maintain. Putting 5 people in a meeting for a few days to brainstorm a solution to a \$1,000 problem doesn't make sense or cents and will only merit negative attention to your otherwise prudent effort.

- **Upper Management doesn't want the truth:** The unfortunate reality is that sometimes the people at the top **are** the root cause. Their mandates, direction, or petty foibles could very well be the reason why a number of things keep happening again and again. High turnover, counterproductive politics, and bad vendor selection have a tendency to come from upper management who simply doesn't care what you, the peons, think about their actions. If your analysis leads you to the conclusion that upper management is to blame, you should reconsider



*You don't have to implement fancy systems or collect metrics for a year to find opportunities to save a bundle.*

communicating this and/or your position at the firm in question.

I have, in the pursuit of pure knowledge and the spirit of total quality, mistakenly shared findings that upper management didn't want to hear - on more than one occasion. Unfortunately, even when you're told "think out of the box" or "you're not going to hurt my feelings", they may not have expected the results your research provides. What I have deduced from my experiences and those related to me is that what upper management frequently wants is for "someone else" (us) to **fix** the problem within a narrowed scope of control - not for us to tell **them** that **they** are the problem. They want someone from the outside to come to the same conclusions they already have and use statistics and research to put a ribbon and bow on their presentation to the Board of Directors or whatnot. It is wise to avoid this landmine.

### The Fishbone Diagram

The Fishbone Diagram is a neat little visual tool. It helps with the brainstorming process of determining causes and root causes. It also provides a quick visual representation of cause density. It's called the fishbone diagram, because when you're starting out, it has a sideways tree look to it which resembles a fish skeleton. Sometimes it's called a cause and effect diagram or Ishikawa diagram because of what it depicts and who came up with the thing, respectively.

To draw a fishbone diagram, draw a line pointing off to the right with the problem statement, such as "Too many defects in the

last release". This is the "effect" part of the cause and effect diagram.



From that main line, create branches for various cause categories. These can either be placed first to fuel brainstorming, or deduced as you go through the process and cleaned up along the way.

Here are some commonly used cause branches when performing Fishbone Analysis:

#### The 8 P's

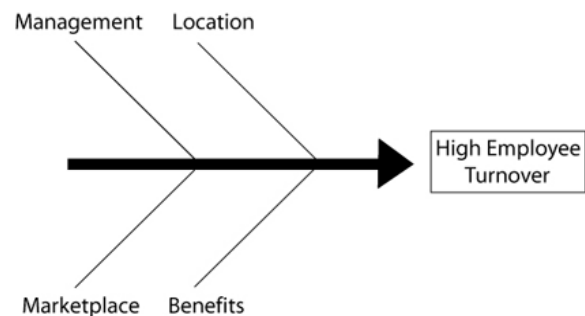
Procedures, People, Price, Promotion, Processes, Plant, Product, and Policies

#### The 6 M's

Machinery, Materials, Maintenance, Methods, Mother Nature, and Man.

#### The 4 S's

Skills, Surroundings, Systems, and Suppliers.



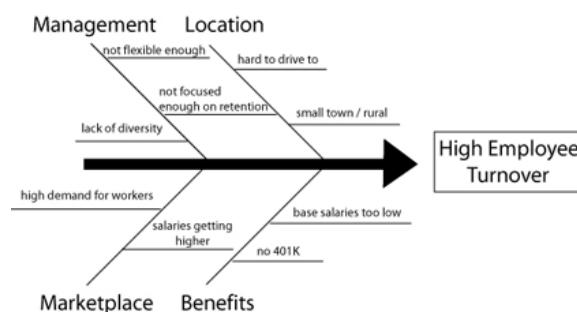
This is when the cause and effect diagram starts to look like a fishbone diagram. After this point, the diagram stops resembling a fishbone pretty quickly. I suppose that's when it evolves into a full-fledged Ishikawa diagram.

## The Five Whys Method of Fishbone Analysis

With a child's inquisitiveness, brainstorm ideas of what might have caused each event you're looking into by asking "why", "why", "why?"

The number five in "5 whys" is somewhat arbitrary – along the lines of the lucky number seven or the unlucky number thirteen. Think of five as a "rule of thumb", the point is to keep the eye on the prize – the "root cause". Whether that takes three whys or twenty whys, keep going until you're done.

Working your way down the fishbone, keep asking why, all the while adding more and more lines off of lines explaining the rationale for each event down the chain, creating a chart which looks a bit like this example.



As an example, in the case of a large number of software defects, one might ask why each of the individual events occurred. From each attributed cause, continue to ask how that particular symptom or situation emerged. And so on until you get to what caused that problem in the first place.

## The Root Cause

Now, when you've worked through all of the individual events and all of the causes suspected – go back through the brainstormed list and figure out which of those, at the

deepest level, is associated with the largest volume of events.

Sometimes, it's obvious from all of the results where the problem lies, and sometimes you'll have to prune out the silly or distracting ones.

Bear in mind that there can be more than one "root cause". Sometimes, several related causes act in concert, creating a perfect storm. Catastrophic failures, for example, seldom occur because one person failed to perform their job adequately. Usually, it's because several people or processes at very key points didn't work as expected. So, don't beat yourself up looking for the fabled "root cause", forsaking all others. You might have a couple of contenders that are all worthy of consideration.

You might even find that an entire category is the "root" cause, as in lack of sufficient management training of a particular department, or lack of accountability in a certain group, and so forth. When you see (ratio exaggerated for effect) 4,000 items in one category and only fifteen in all the others, it should become rather apparent where problem-solving talents should be focused.

This process of looking for a "big group of problems in one place" concept is called "Pareto Analysis". You may already know of the Pareto Principle by its more common name, the 80/20 rule. The major premise behind Pareto Analysis is that you should spend your time trying to figure out that one thing (or small set of things) which will have the greatest impact if you fix **just** that.

If you don't apply the Pareto Principle to your analysis, you run the risk of "Analysis Paralysis" – that's where you spend so much time thinking

about the problem that you never end up doing anything about it – watch out for this, it's a really big time waster. As Voltaire said, "[perfection is the enemy of good enough](#)". You need just enough information to do something about the problem.

### Brainstorming Solutions

As Peter Drucker, father of "modern management", has said, "Plans are only good intentions unless they immediately degenerate into hard work." Once you come up with the root cause(s), it's imperative to work on a plan to address those causes. Try to come up with easy things to do (or stop doing) that will knock out a majority of your root causes or the singular root cause most of the time.

Be careful that your answer isn't "more process". Sadly, you can't test quality into a process. As a result, it's most effective to find ways to improve the work in the first place rather than "check to make sure it got done."

### Planning the Work

Once you have one or two fairly inexpensive ways to address your root cause, work out an incremental plan for putting those changes in place. Do it fast, but do it small, especially at first. Don't be afraid to start small. It's better to do the right thing slowly than to do the wrong thing with a deafening thunderclap.

**WARNING:** Resist ALL temptation to create more software to solve the problem. Again, beware Maslow's Golden Hammer. Chances

are that your problem has to do with people and processes rather than how quickly those people execute processes. Software should be used to speed up things that are already going well, rather than to impose an intended change.

### Working the Plan

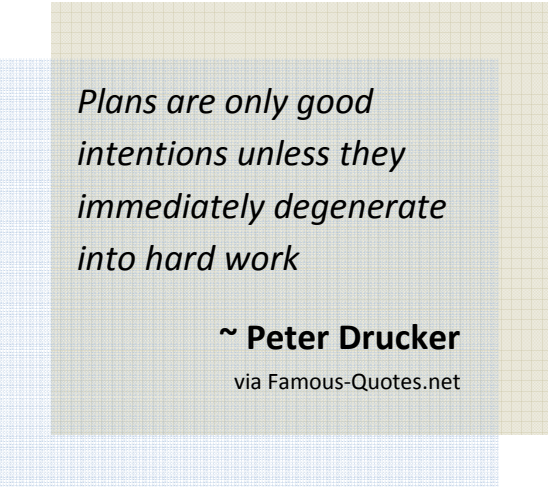
Make it happen. Try the changes with one group on paper. Pick a group who are really dedicated to making things better, rather than the group who need it most. The group in need of the most help may need that help for far more reasons than those addressed by your new plan.

Refine your changes as you learn more about your successes and setbacks with your test group.

You may find widespread adoption easier with a little automation to reduce the time it takes for people to come up to speed on the new direction.

Once you've proven that your plan works in a smaller

environment, you've ironed out all of the wrinkles, and you have a way for people to execute the new plan with minimal discomfort - you're ready to roll it out for the big show and really have an impact. Broadcast your root cause analysis, your findings, your solution, and your initial results and others will beg for your assistance to solve the problem in their part of the organization.



*Plans are only good intentions unless they immediately degenerate into hard work*

~ Peter Drucker

via Famous-Quotes.net

### Further reading:

- [Root Cause Analysis](#)
- [Fishbone Diagrams](#)
- [The Pareto Principle](#)
- [Quotes by Kaoru Ishikawa](#)
- [Maslow's Golden Hammer](#)
- [Analysis Paralysis](#)

### About the Author:

David Russell is a Management Consultant in Central Florida specializing in Software Development organizations. David helps companies build and improve software teams while delivering meaningful products that satisfy all stakeholders on time and on budget. Follow him on Twitter at [Twitter.com/DaivRawks](https://twitter.com/DaivRawks).

