Troubleshooting Methodologies

Using Structured Methods To Solve Real Problems

Jeff Whiteside
University of Alaska
Sr. Network Engineer

“Being an expert is more than understanding how a system is supposed to work. Expertise is gained by investigating why a system doesn't work.”

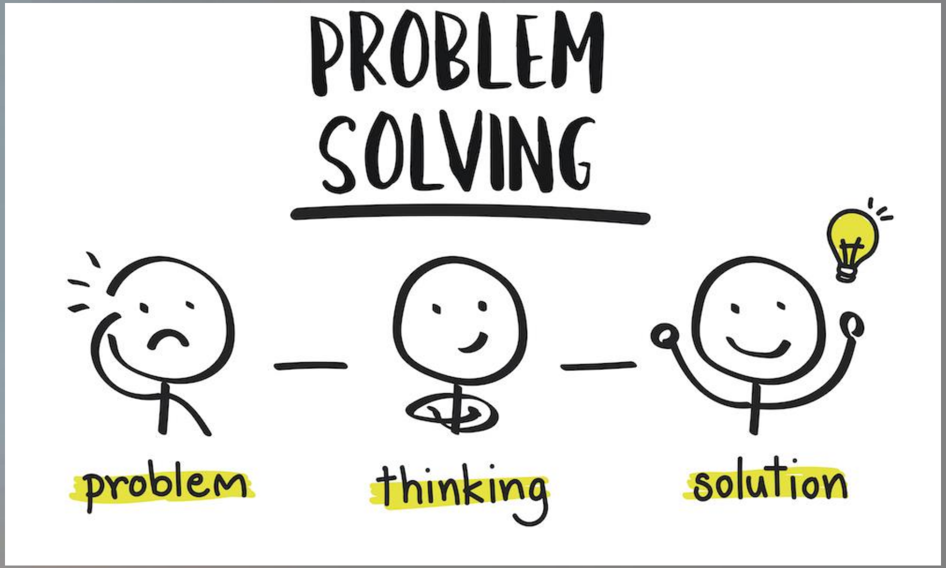
What Do You Think Troubleshooting Is?

- Is it guessing about what a problem might be?
- Is it trying to prove a problem is what you think it is?
- Is it blindly following internet tutorials until you fix it?
- Is it trying everything until you run out of things to try?

Troubleshooting is NONE of these things!

What Troubleshooting "Ackshually" Is

- Troubleshooting is a form of problem solving, usually applied to machines or systems that have failed
- It is the use of one or more structured methodologies to identify root (or most likely) cause
- Often combines a process of elimination or a diagnostic flow to isolate a specific cause
- It is a skill that can be learned, improved upon and applied to one's expertise (or lack thereof)
- It is critical thinking. Not guessing or magical thinking.



Putting The Cart Before The Horse

- Effective troubleshooting requires two things. Knowing how to troubleshoot and general knowledge of how a system works
- Investigating a broken system is an excellent opportunity to learn how something works
- More experience is gained by investigating broken systems than working ones

5 Immediate Steps Before Troubleshooting

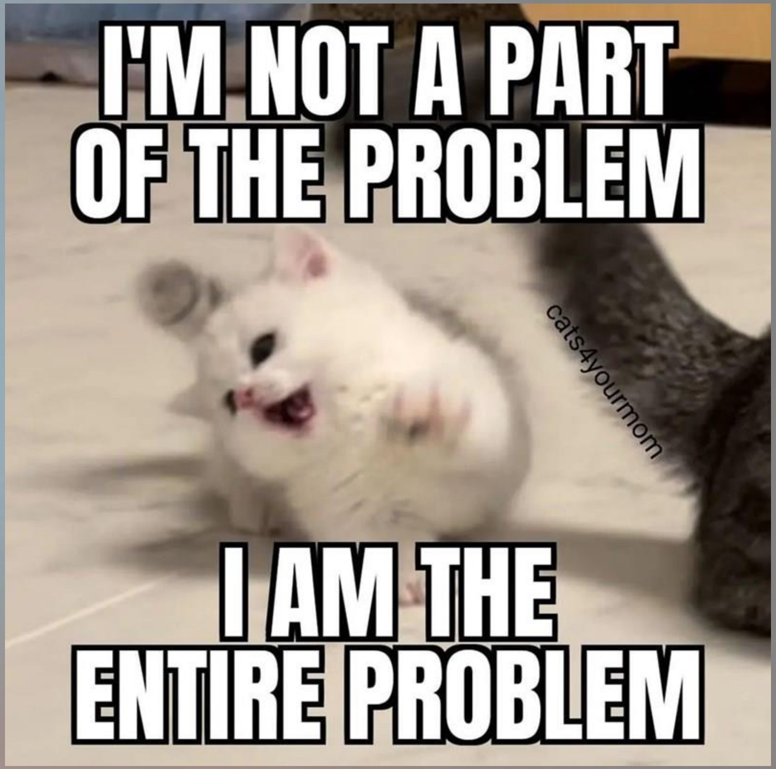
- Step 1: Clearly identify and state the nature of the problem
- Step 2: Identify the end goal or “how it should work” (What is working state?)
- Step 3: Confirm the problem exists (reproduce!)
- Step 4*: Identify the simplest form in which the problem can be reproduced
- Step 5: Assess which troubleshooting methodologies might be best for the given issue

Understand Your Problem!

- One of the most important things is to truly understand your problem. Vague definition results in vague conclusions.
- Examples of important things to know?
  - What is the problem?
  - Where does the problem occur?
  - When did the problem start?
  - When does the problem occur?
  - Who does the problem affect?
  - What are the symptoms of the problem?

Common Examples Of Problem Causes

- Design Issues (i.e. bad design)
- Human factors
- Component failure
- Input failure
- Faulty state (i.e. turn it off and on again)
- Misconfiguration
- Incorrect assumption or understanding
- No trouble found



Troubleshooting Structures

There's More Than One Way To Approach A Problem

The Guessing Game Method

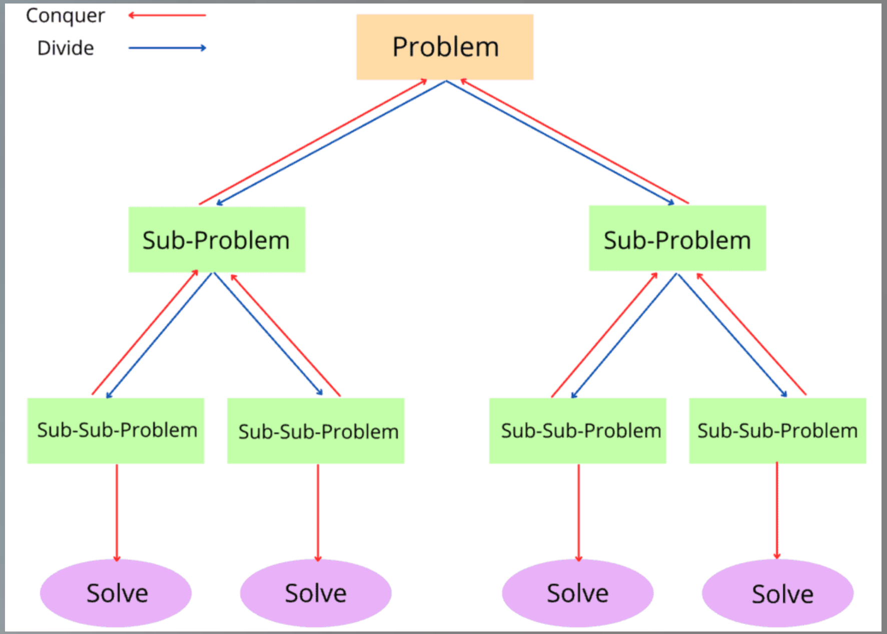
- This method usually involves inventing (guessing) various possible problems and then trying to prove your problem fits into them
- Often used by beginner troubleshooters, tends to be a “natural” way of approaching problems
- Very poor methodology, low accuracy and often time consuming. Depends a lot on luck.
- Many problems may have possible causes that aren’t even considered with this method
- Can be used by those very experienced with a system as the guesses are likely quite accurate

Serial (Linear) Troubleshooting Method

- Usually involves serially (or linearly) checking/replacing each component in a system (e.g. replacing parts, loosening, tightening, turning on/off, etc.)
- Often doesn’t feature structure or a hypothesis that will help guide diagnosis
- Tends to be very inefficient and slow, but will often uncover the nature of the problem
- May be a good technique for “simple systems” with few options

Divide & Conquer (My Favorite)

- Involves dividing the problem "roughly" in half (or into logical sections) to isolate where the problem is. The system is further divided until few possibilities remain
- Helpful in isolating large swaths of possible problems
- Great to use early in troubleshooting to focus other, more appropriate troubleshooting methods later
- May not allow full isolation of the problem due to inability to divide further or with enough granularity
- Can also be gamed (i.e. play 20 questions)



The "What Changed" Method

- This method is usually invoked when it's known (or highly suspected) the failure related to a prior change (e.g. Joe changed spark plug, engine doesn't start)
- The proposed change should very clearly have a plausible reason to be suspected as the source of the problem (no magical thinking!)
- Can slow down or delay diagnosis if the change assumption is incorrect
- Usually involves reverting the prior change(s) as a test, or working around the prior change in some way
- Older changes are sometimes overlooked, if a problem isn't uncovered soon enough

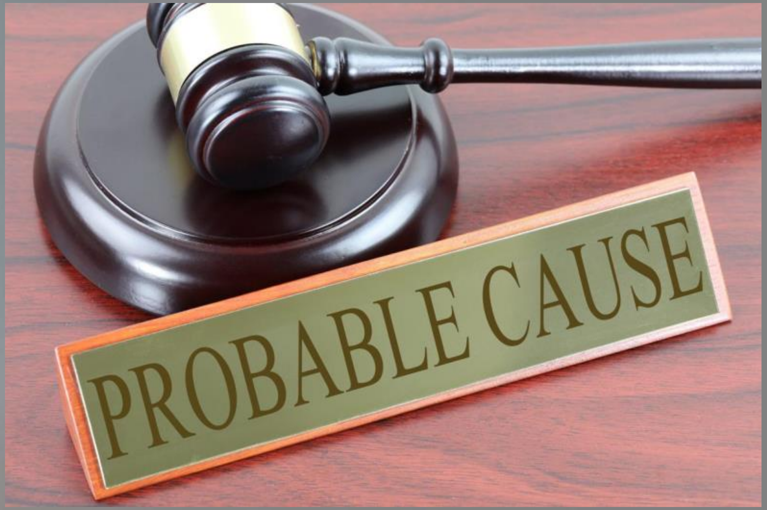
The Process Of Elimination Method

- Typically involves systematically removing, disabling/enabling or changing components of the system to see if the issue persists
- Often used to identify the simplest form in which the problem can be reproduced (simplify troubleshooting)
- Order of operations is logical, from most likely to least likely
- Good for complex systems with many interdependencies, can also feed into other troubleshooting methods (e.g. divide & conquer, serial, etc.)
- Often doesn’t result in root cause, but can strongly indicate where further diagnostics should be pointed

I appreciate your valuable contribution over the years, but as you can see, I can now operate the can opener, so your position is being eliminated.

The Probable Cause Method

- Structure usually involves identifying the problem, researching the issue, establishing one or more probable causes followed by one or more tests to prove the cause or gain more information.
- Best to order the probable causes in order of most to least likely. Least to most intrusive is also valid in many cases.
- Often good to employ in the final phases of troubleshooting complex systems to truly identify root cause
- Can often produce “red herrings” and depends highly on the quality of the initial research



The Five Why's Method

- Involves asking "why" 4 to 7 times, with each "why" aiming to answer the previous "why"
- Invented by Toyota and presumes to get to root cause within 4-7 why's
- Example: Why is the UPS not running? Batteries are dead. Why are the batteries dead? They are bad. Why are they bad? They weren't changed recently. Why weren't they changed recently? They weren't on the maintenance schedule. Why weren't they on the schedule? Because it wasn't known the UPS was under their care. (i.e. a human problem)
- Method is great for many kinds of problems, but it suffers under highly complex or intertwined systems
- Sometimes misses the real root cause (e.g. a missing process, procedure, etc)

The Top-Down or Bottom-Up Method

- This method involves mapping out the system, then going through each successive step to identify where the failure in the system is.
- Directionality is usually determined by where the problem is “initially thought” to be
- Often combined with divide & conquer to limit the scope of troubleshooting efforts
- Main weakness is multiple problems and intermittent problems, where the root issue can be missed

The Detective Method

- Involves using logs, historical data or past incidents to establish common patterns. May involve reporting from multiple systems for correlation.
- A good method to employ with intermittent issues that are difficult to reproduce.
- May involve establishing "enhanced debugging" or diagnostic methods over time to establish information flows
- Important to review multiple incidents for similar or identical behaviors that lead up to the problem

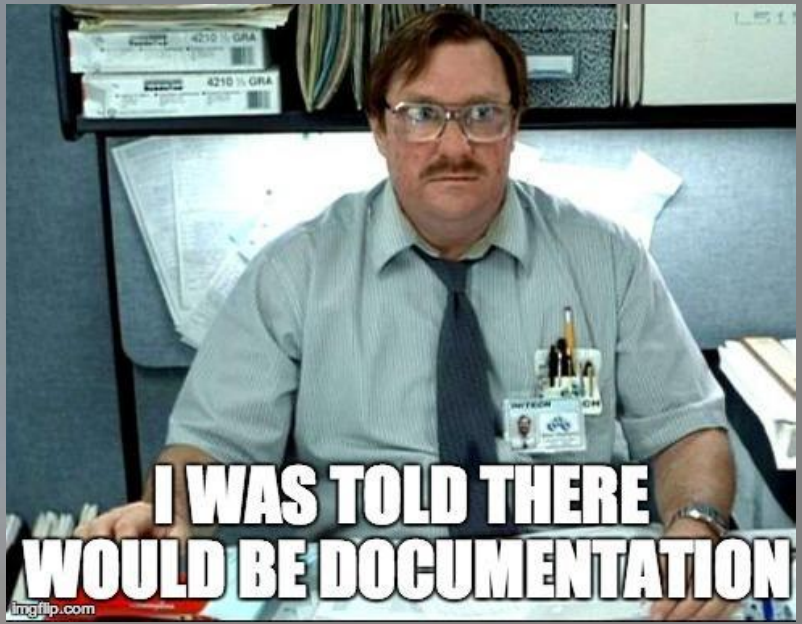
Other Troubleshooting Methodologies

- There are <u>many</u> other troubleshooting methodologies that can be utilized.
- Examples:
  - Scientific method
  - Known good state
  - Brainstorming
  - Fishbone (Ishikawa) diagrams
  - Means-ends analysis
  - 8D method (Ford)
  - OODA Loop (Observe, Orient, Decide, Act)
  - Specialized discipline-centric techniques

Helpful Practices In Troubleshooting

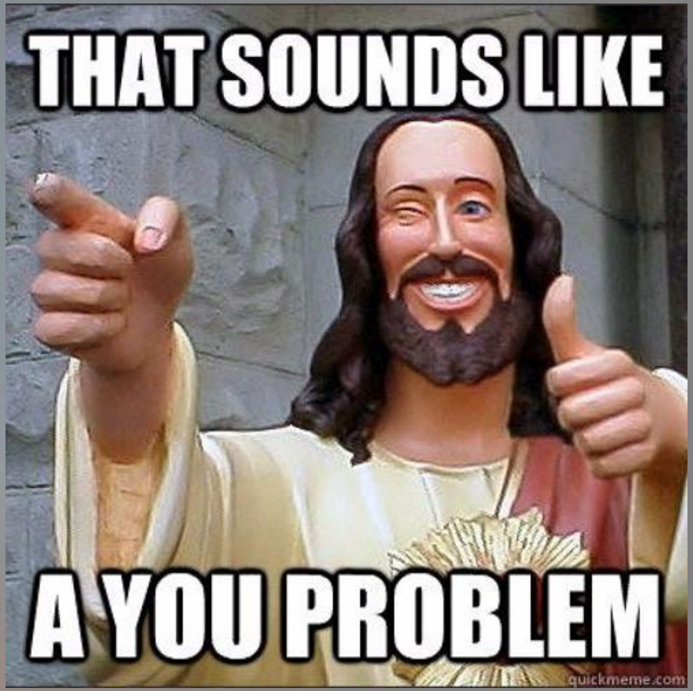
Establish Or Update Documentation

- Most systems are documented to some degree. If not, start that process!
- Documentation persists. Memory is faulty.
- Major troubleshooting efforts for the system should absolutely be documented.
- Examples of documentation methods? Binder, wiki, electronic documents, etc.
- Good information to keep? System manuals, project documentation, service bulletins, prior troubleshooting notes, change history, etc.



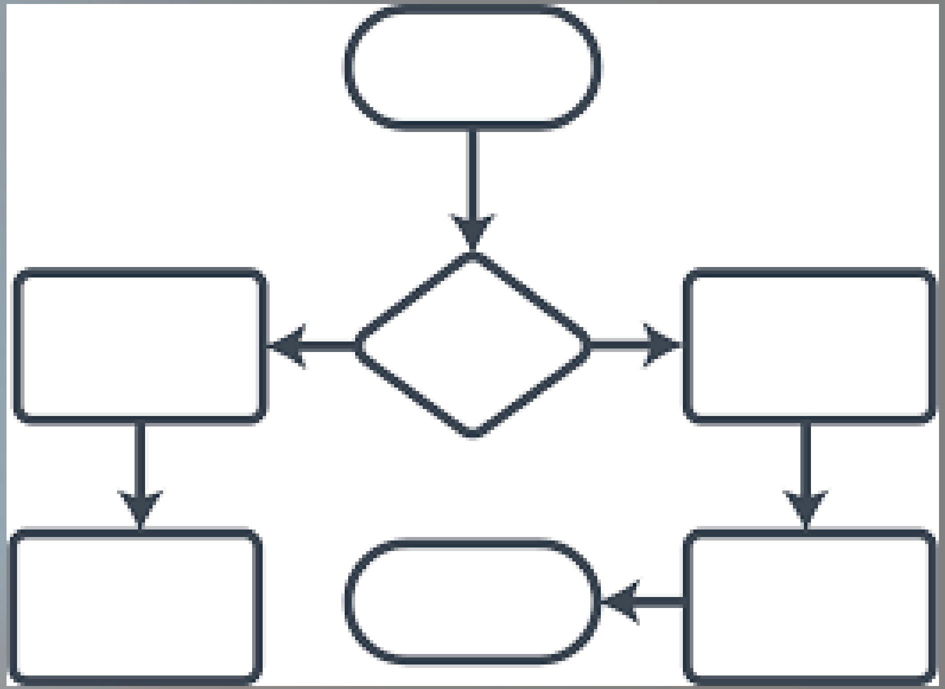
Ask For Help / Talk About Your Problem

- When you're deeply into an issue, it's often very helpful to talk about it with your peers
- Describing the issue in detail often leads you to consider possibilities you hadn't previously considered
- Probing questions asked by others also leads to considering other possibilities
- An outside perspective can approach the issue without any bias, sometimes leading to clearer thinking



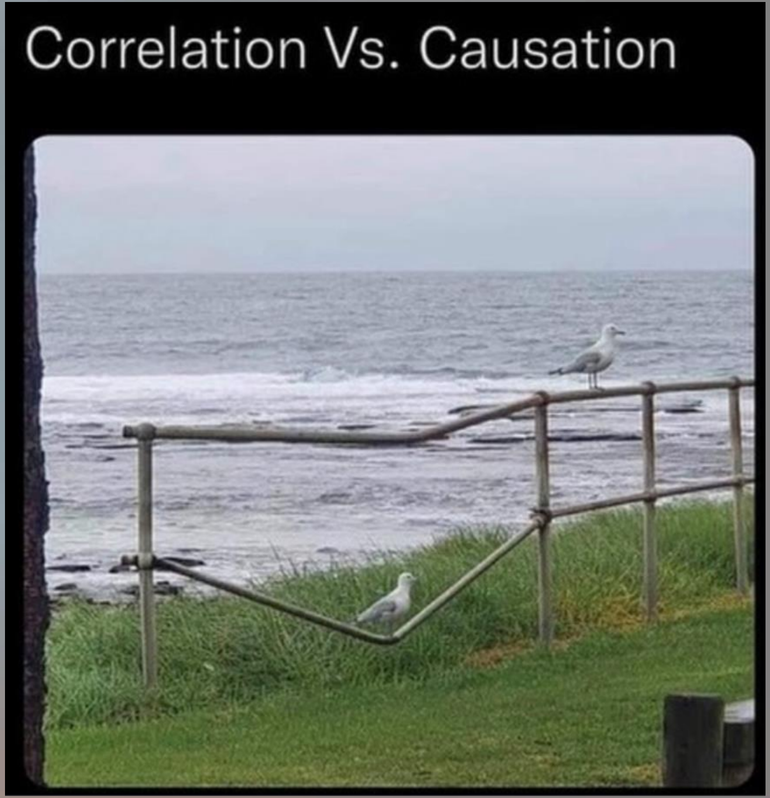
Establish A Flow Diagram

- When understanding complex systems, a flow diagram is often a helpful reference
- Documents, at a high level, how the given system works, along with various inputs or outputs that are necessary for system operation
- This flow diagram can be used with nearly all diagnostic methods
- Very useful for “consults” when you ask for assistance and may need to bring others up to speed



Deeply Understand Practicality & Causality

- All problems have causes
- Not all problems need solutions
- All failures are not equally probable. When you hear hoofbeats, think horses not zebras.
- Correlation is not causation. Just because something does happen, doesn't mean it's related to your problem.
- Coincidence is a very real thing and it happens all the time!

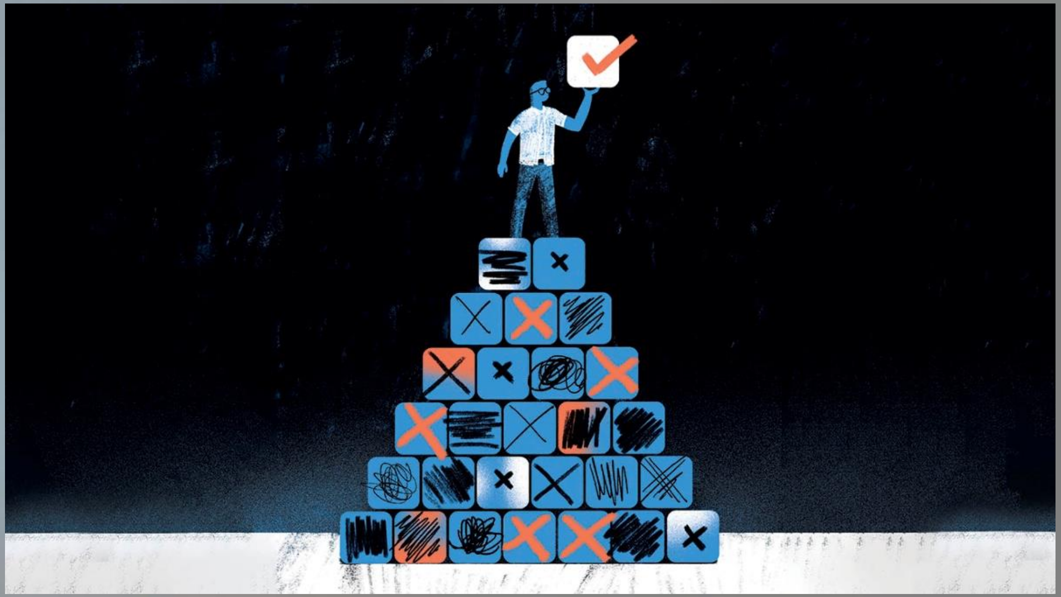


Beware Of Multiple & Intermittent Problems

- Both of these scenarios make the troubleshooting process inherently more difficult
- An intermittent issue is defined as one that does not have a known reproduceable cause
- Multiple issues can often interfere with troubleshooting as singular resolutions aren't effective
- Try to establish the detective method, when possible!
- Best of luck!

Negative Results Are A Positive

- When a test or change reveals an inaccurate diagnosis, this can tell you a lot!
- Knowing what a problem is not is valuable, conclusive proof
- Negative results can often inform future diagnostics and testing methodologies, establishing minimum levels of failure
- Illustrates helpful results for troubleshooting techniques like divide & conquer

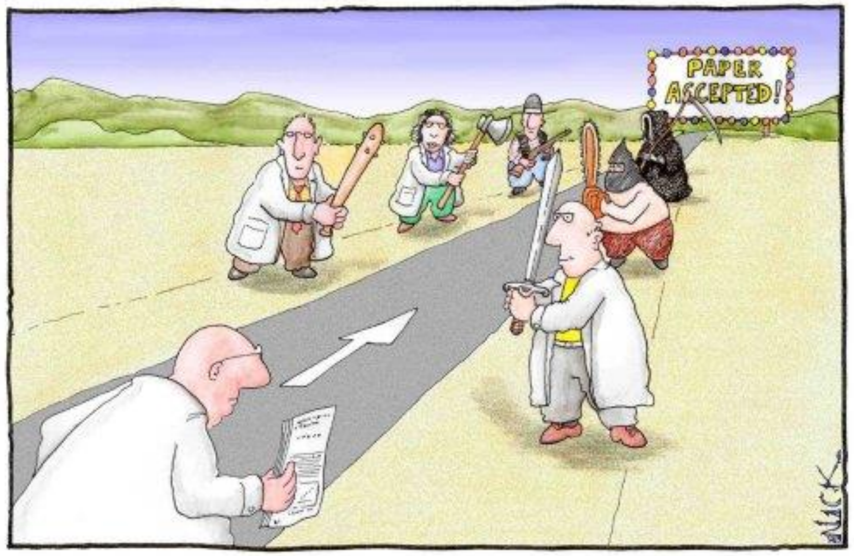


Go Deep – Logging vs. Debugging

- Many systems feature logging capabilities. Some systems also feature debugging capabilities. These are not the same.
- Logging is “high level” and in theory, what you need to know
- Debugging is a deep dive into the system’s operations, often the deepest possible output the system can create
- Debugging often requires expertise and knowledge built over time to interpret the information properly
- The tool of choice is often what separates the casual troubleshooter from the expert troubleshooter.
- Some problems are better served by one or the other

Post Mortem Reviews

- Often seems “managerial” in nature, but it serves the purpose of formalizing the troubleshooting process and problem resolution
- Usually results in a document that can then be put into documentation or referenced otherwise
- Helpful as a teaching method for others, builds broad awareness of the system



Most scientists regarded the new streamlined peer-review process as ‘quite an improvement.’

Keep A Long Term Troubleshooting Journal

- Incredibly helpful to establish a personal "journal" of various troubleshooting experiences. Can act as a reference in future troubleshooting.
- Important captures? Problem description, notes about the diagnostic process, root cause discovery, resolution method.
- Personal in nature, so it can be referenced wherever you are at

DEAR DIARY, MY FOOD DISH IS HALF-FULL
IT IS OBVIOUS THAT I WILL SOON STARVE TO DEATH, THIS MAY BE MY LAST ENTRY.

Common Pitfalls In Troubleshooting

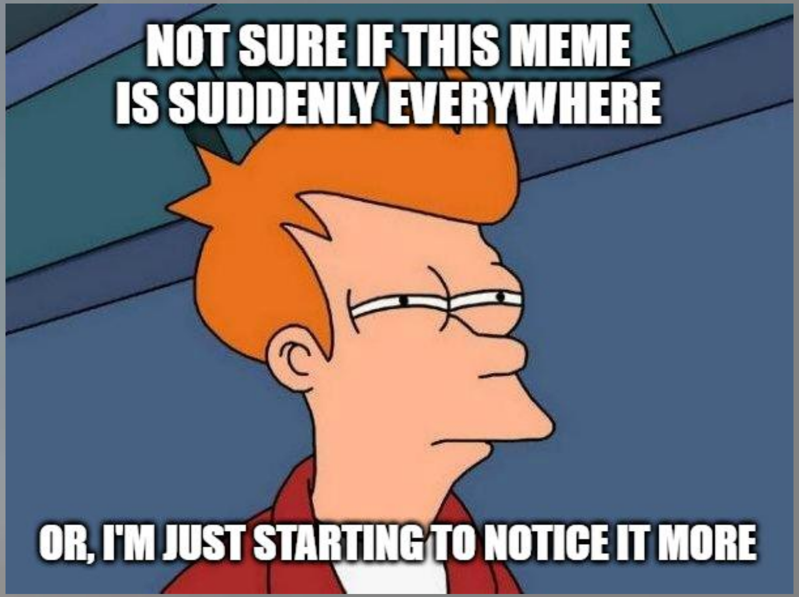
Recency Bias

- Example: Yesterday, there was a network issue and that's why my printer doesn't print.
- Avoid the temptation to "blame" (instead of consider) recent problems as the cause of new or unassociated problems
- Recency bias often leans into "magical thinking" that everything is mysteriously intertwined
- Most often seen from "end users" that underestimate complexity and assume everything is always the same

EVERYTHING I FIND THAT AGREES WITH WHAT I THINK
REMINDS ME HOW RIGHT I AM
imgflip.com

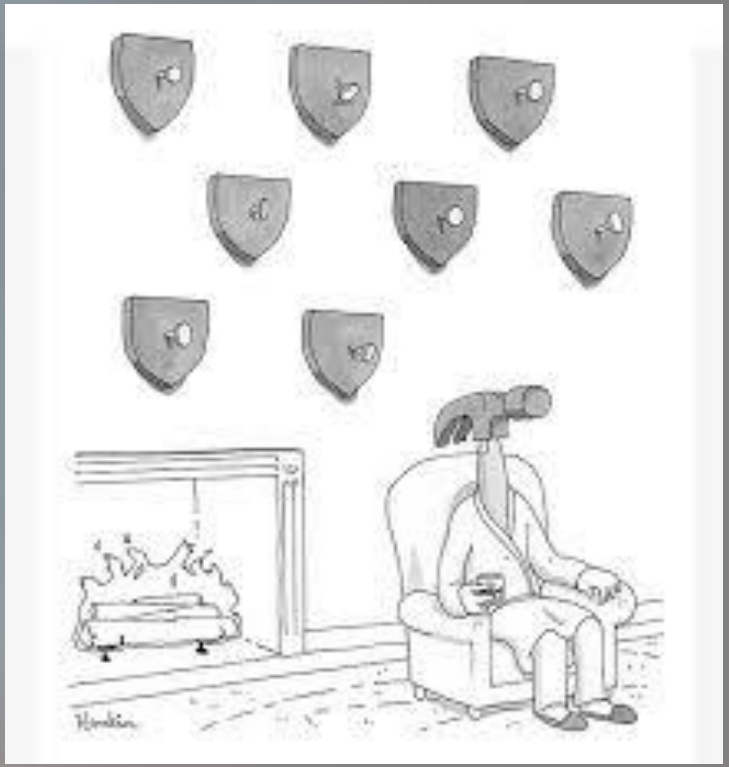
Frequency Bias

- Example: Light bulbs frequently burn out, therefore a light that doesn’t light is always a burnt-out bulb.
- Just because something sometimes (or even often) happens, doesn’t mean it’s always the cause of a given problem
- Seen even from technical professionals, well versed in troubleshooting
- Often results in wasted time and effort, proving things aren’t a factor. Using troubleshooting methodologies would lead to faster resolution


NOT SURE IF THIS MEME IS SUDDENLY EVERYWHERE
OR, I'M JUST STARTING TO NOTICE IT MORE

The Hammer's Bias

- Example: I just saw this problem yesterday. It looks like I need to do the same fix again today.
- Often affects those who are gaining expertise in a system, before every possible cause has been enumerated
- People well versed in a discipline will be inclined to limit diagnostics within their discipline
- Often can indicate that root cause has not actually been determined
- Important to consider these incidents are likely symptoms and not causes



The Professional’s Bias

- Vast experience with a system often allows for rapid diagnosis and troubleshooting of those systems (e.g. inference)
- Trust your instinct & allow those rapid diagnostics to occur
- Be aware that this skips many critical analysis steps, which can easily mask problem cause
- It may be important to “step back” from the issue and default to troubleshooting methodologies if “expert swipes” don’t result in resolution

Troubleshooter Induced Problems

- Change is often a necessary part of troubleshooting. It's important to understand that change can affect the system and cause more problems.
- It's often a good best practice to back-out any induced changes, if proven to not work
- Some changes may fundamentally change the way the system works!
- Some issues mandate change to meet "best practice" and may not be desirable to back-out. (e.g. calibration, tolerance, "correctness," etc.)
- Be sure to think through the consequence of these changes and be prepared to defend their innocence (or consider guiltiness)

Let’s Discuss

Thank you for coming!

Enjoy The Rest Of RVTEC 2025!