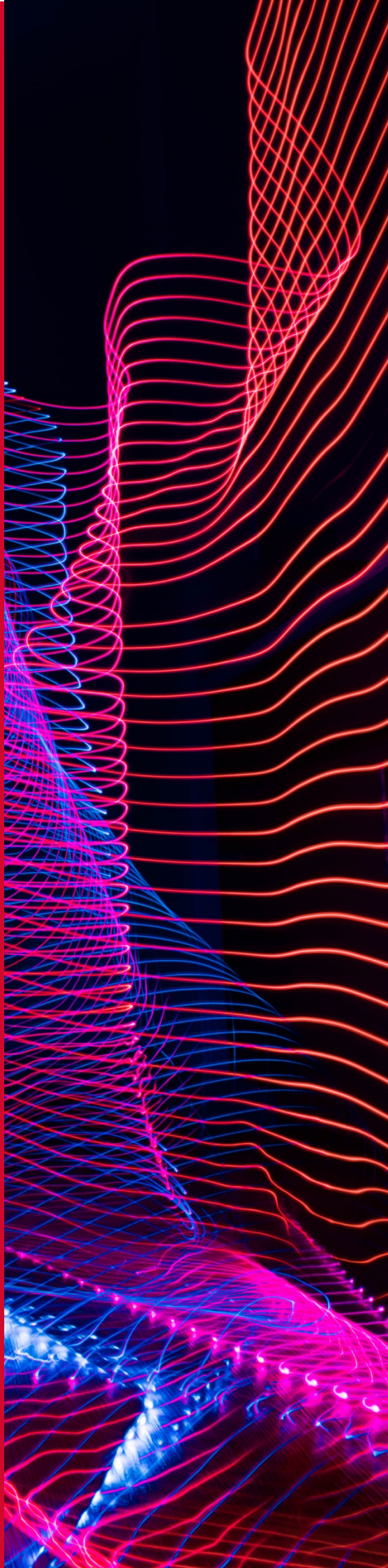
responsiv

simple · effective · distinctive

POINT OF VIEW

Solve Problems Like an Expert in 10 Steps



More Method Than Madness When You Solve Problems Like an Expert

Some people solve problems with very little information. Some make endless demands for more information before making a determination. While others can jump to the wrong answer with relative ease.

What is the secret of minimal information and right answers?

Minimal Information

There are some (extremely annoying) people in this world who seem to be able to walk into a room and solve problems without appearing to know anything about the underlying technology.

At first this looks like "lucky guesses", but these people are **incredibly lucky** with their guesses. When we observe these "experts" in operation closely **a pattern** becomes apparent.

You can learn the method and be just as successful at solving problems. Here are ten things to get you started...

Outsider Advantage

When technical teams are under pressure to solve problems in a critical situation, they often don't take a step back and look at the problem.

It takes an outsider to step into the frame and guide the team.

The outsider is unaffected by previous assumptions and may not be as "new to the problem" as they appear. They have seen similar problems, may understand the technology or similar technology, and may have a method.

Solve Problems like an Expert

Jumping [1]

Never jump to a conclusion until you have the full picture, you will be wrong and look less expert in the long run. "Knee jerk reactions" usually result in bruised knees.

Underestimating [2]

Don't underestimate the problem and try to use some quick hacking to see if you can fix it, a small problem will be solved quicker using the techniques we will show you.

Many problems are prolonged by an erroneous first attempt to fix that obscures the original symptoms and confuses the situation or worse, makes the problem far more complex to solve.

Asking questions [3]

Ask the silly and "obvious" questions to fully understand the problem. Make sure you are clear about the terminology and what is actually being said. For example

"John will be late" - how do you know that? Because John said he "had jobs to do before leaving". Ok, so he may be late but may be early. In fact, until John is late, we don't know anything.

Often the technique of "I don't know about this product, please explain" helps to reveal how much others actually know, and their fluidity in answering will illustrate their confidence. It will also make them take a mental step back to allow you to get the answer you need.

It is likely that you will be told things you already know. This is actually very useful, because it allows you to check their knowledge, and assures that they did not assume common knowledge - which is often anything but common.

Finding the Edges [4]

Gather as much information about the whole solution, especially what is around the edges.

Understand the inputs and outputs of the system, this includes expected loads and actual loads when the problems occurred.

Consider the Time Dimension [5]

Temporal questions are often revealing. What went on before the problem occurred? Were there any recent or relevant changes to the solution? What upgrades, patches, or hardware changes have happened?

Does the problem occur at specific times or dates? Build up a timeline of the problem.

Check the Facts [6]

Temporal questions are often revealing. What went on before the problem occurred? Were there any recent or relevant changes to the solution? What upgrades, patches, or hardware changes have happened?

Does the problem occur at specific times or dates? Build up a timeline of the problem.

Check the Work Date [7]

Temporal questions are often revealing. What went on before the problem occurred? Were there any recent or relevant changes to the solution? What upgrades, patches, or hardware changes have happened?

Does the problem occur at specific times or dates? Build up a timeline of the problem.

Don't Fixate [8]

Try to avoid anchoring on a single solution, reassess the facts as new information becomes available.

Change One Thing at a Time [9]

Avoid changing more than one thing at a time, you will have no way to identify what fixed the problem if you make multiple changes.

Note that sometimes multiple things need to be changed together - in these cases the set of changes should be restricted to one set of expected outcomes.

Whenever you make a change document the expected outcome. Compare your expectations with the reality - if you are wrong, or the result was unexpected, then ask why.

Write Everything Down [10]

Document all the questions and answers ready for your report, or ticket resolution.

An Example

The following example is based on a "simple" problem that was difficult to fix.

One of our web servers was crashing at random times. Each time the memory was increased to fix the problem along with a script that rebooted the server every once in a while.

I do wonder how many servers out there are being rebooted every day rather than fixing the problem at root cause?

What we know	How we know it
Website crashed at 13:27 on 01/01/021	Zabbix reported "off line"
Website ran out of memory	Reported in /var/logfile at 13:26
Website rebooted with twice the memory	System admin confirmed (knew jerk fix – you know who you are)
Software versions, centos7, WPress 5.x httpd 2.4.x, java, php 5.3	Verified using yum list installed

What we don't know	How we can know it
What was the load on the website at the crash	See Zabbix or examine /var/httpd/logfiles
Is the memory sensible value	See monitoring
Is software up to date	Check suppliers website
Any known problems of this version	Check using yum
Has the increased memory fixed the problem	Monitor closely using linux tools & Zabbix

These two tables are invaluable when working to trouble shoot, they stop people going down "rabbit holes" and retesting the same thing more than once.

From the two tables above we can now make hypothesis and a plan.

What we are going to do	How we do it and who does it	Answer
Get load statistics for website and collate	Download log files and sort by time in excel, plot graphs	Load normal
Check recommended memory for expected load	Read docs	Memory now too much
Confirm software versions	Sysadmin to run yum commands	Back version running
Check software bug list for version	Sysadmin to read docs	Some memory related bugs not specific to this problem? php
Monitor memory usage and load	Add Zabbix & scripts to gather data for the next month	Memory usage still growing

Update the What we know.

What we know	How we know it
Website crashed at 13:27 on 01/01/021	Zabbix reported "off line"

Website ran out of memory	Reported in /var/logfile at 13:26
Website rebooted with twice the memory	System admin confirmed (knew jerk fix – you know who you are)
Software versions, centos7, WPress 5.x httpd 2.4.x, java, php 5.3	Verified using yum list installed
Memory is increasing under normal laod	Load graphs and resource graphs

New hypothesis and a new plan based on new information.

Looks like the software has some known bugs at this version so we need to eliminate these from the equation.

What we are going to do	How we do it & who does it
Update WPress & php to latest	Snapshot server & follow upgrade docs
Monitor memory usage vs load	Add Zabbix & scripts to gather data for the next month
Return memory to recommended size if OK	VMWare change

What Now?

Now you know how to solve problems like an expert. If you want to learn more about problem solving, or want to get some advice, get in touch today!