X
World 2011

Help!

Problem Solving and Troubleshooting

Daniel Rodwell
Australian National University

AUC

XWII

Intro

Outline

Today's Session

Two Parts

- Problem Solving
  - Concepts and Theory
  - Methods
  - Group Solve

- Troubleshooting
  - Concepts
  - Methods

Today

What's in it

- Professional Development workshop
- Toolset for you to use
- Lighthearted, not too serious
- Mixture of Skills and Backgrounds
  - hopefully theres something here for everyone

Part 1: Problem Solving

Problem Solving Concepts

Problem Solving

The dictionary says...

problem |ˈpräbləm|

noun

1 a matter or situation regarded as unwelcome or harmful and needing to be dealt with and overcome : *mental health problems* | [as adj.] *city planners consider it a problem district.*

• **a thing that is difficult to achieve or accomplish** : *motivation of staff can also be a problem.*

ORIGIN late Middle English (originally denoting a riddle or a question for academic discussion): from Old French ***probleme***, via Latin from Greek ***problēma***, from ***proballein*** ***'put forth,'*** from ***pro 'before'*** + ***ballein 'to throw.'***

Problem Solving

The thesaurus says...

problem
noun

1 *they ran into a problem*: difficulty, trouble, worry, complication, difficult situation; snag, hitch, drawback, stumbling block, obstacle, hurdle, hiccup, setback, catch; predicament, plight; misfortune, mishap, misadventure; dilemma, quandary; informal headache, nightmare.

2 *I don't want to be a problem*: nuisance, bother, pest, irritant, thorn in one's side/flesh, vexation; informal drag, pain, pain in the neck.

3 *mathematical problems*: puzzle, question, poser, enigma, riddle, conundrum; informal teaser, brainteaser.
adjective
a problem child: troublesome, difficult, unmanageable, unruly, disobedient, uncontrollable, recalcitrant, delinquent.

ANTONYMS well-behaved, manageable.

Problem Solving

The dictionary says...

solve |sälv; sôlv|

verb [trans.]
find an answer to, explanation for, or means of effectively dealing with (a problem or mystery) *: the policy could solve the town's housing crisis* | *a murder investigation that has never been solved.*

Problem Solving

In Context

For System Administrators or System Engineers

- design a new system
- grow an existing system
- transition to another system
- codify a process or activity
- solve an IT need

But...

Problem Solving Skills are reusable!

- Core Skills can be applied generally to solve non-IT problems, anywhere.
  - design a building
  - organise a world-wide roadshow
  - fix something

How do we know?

How do we know we have a problem?

Two ways we typically discover a problem

SENSE

we sense something is different from 'normal'

TOLD

someone tells us we have a problem

At this point

You should be thinking...

ALERT!

SUBJECTIVE INFORMATION SOURCES

AUC

XWII

Subjective

cf. objective

- Perception based
- typically not driven by fact or data
- opinion rather than scientific observation
- *May contain traces of Emotion*

How do we react?

How do we react to a problem?

PANIC !

AARGH!
SCREAM!
ALARM!

EEEEK! TERROR! FLUSTER! HYSTERIA!

GRUMBLE BLAMETHROWER SPAGHETTI-CHUCKER GRIPE

SIGH
MOAN

COMPLAINT :(

DISMISSAL...

WHATEVER...
AGAIN...

SHE'LL BE RIGHT... THERE IS NO PROBLEM... MMMM K...

How do we react?

How do we react to a problem?

Sometimes, but rarely

- Analytically
- Pragmatically

movie clip

Understanding the Problem

Don’t be mislead or confused

Before you do anything:

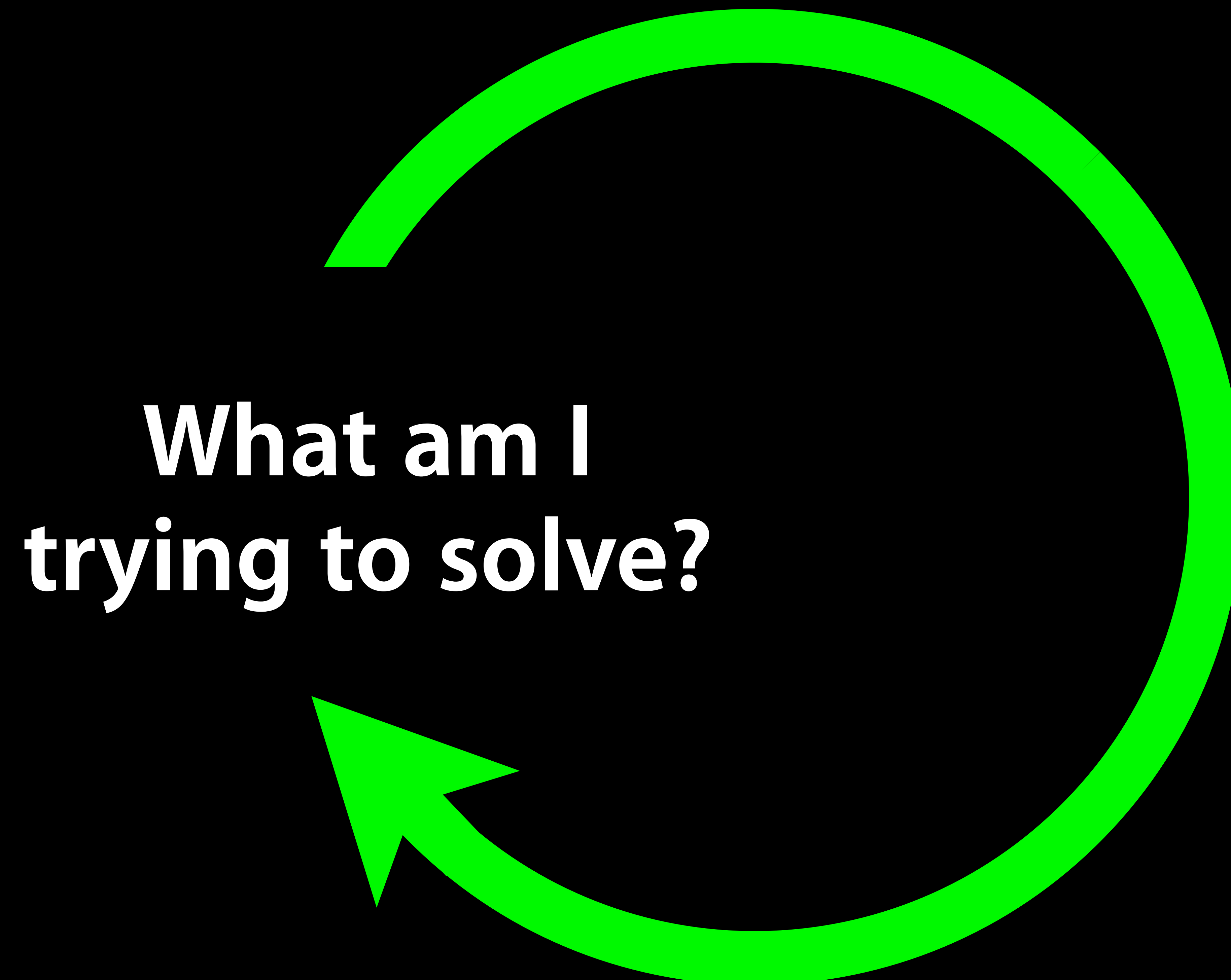
1. **Determine if there is an actual problem**
2. **clearly define what the problem is**
3. **and what you are trying to solve**

(the act of solving is sometimes the easy part).

Why?

We want to make the situation better, not worse.

(how many times have you seen the opposite happen?? DIY anyone?)



Constant Re-evaluation

AUC

XWII

OBVIOUSNESS ALERT!

This all seems like common sense.

But... its easy to get lured into a big mess.

Often you don’t know you have a big problem, until you have a really big problem.

How do we get in this mess?

Understanding the precursors

1. **Pressure** (Management, time, resourcing)

· Rationale and the ability to reason often disappear under pressure.

· Your focus is set on “fix” rather than “solution”.

· There may be few incentives to step back, and think before doing.

How do we get in this mess?

Understanding the precursors

2. **Limited Familiarity**

- The technology is unknown to you or you have only basic knowledge
- You've inherited a system and it's broken
- You're new to a role or organisation

How do we get in this mess?

Understanding the precursors

3. **Overconfident**

- Massive underestimation of the problem
- “how hard can it be?”

How do we get in this mess?

Understanding the precursors

4. **Quick Fix Temptation**

- It's tempting
- It's delicious
- You'll regret it later.

Quick Fix Now = probably a really big problem later.

Problem Solving Methods

Stage 1 - Problem Definition

1. Determine if there is actually a problem

- Gather information
- Understand the situation
- Establish a baseline where the problem is a ‘variation on normal’ - ie capacity & performance problem.
- Verify the problem exists

Stage 1 - Problem Definition

2. clearly define what the problem is

– Scope

– Impact

– Nature

Stage 1 - Problem Definition

3. what are you trying to solve

- Outcomes
- Deliverables
- Solution

- ie. What you want to see at the end of it.

Simple Example

We have No Milk!

1. Determine if there is actually a problem
   - Look in the fridge. Yes, there's no milk.

2. Clearly define what the problem is.
   - We need milk for breakfast in the morning, and we don't have any.... and I need a a coffee before leaving the house.

3. What are you trying to solve.
   - Get enough milk for breakfast, nothing more, nothing less.

What am I trying to solve?

How many systems or projects have you seen that don't solve the original problem?

Remember this ?

Stage 1: Problem Definition

- Stage 1 is your foundation - weak problem definition will lead to weak solutions.
- Your problem definition doesn't need to be pages and pages of blurb. A concise, accurate problem description is better
- Stage 1 is knowledge and familiarity building.

Knowledge + Familiarity = less stress

Stage 2: Research

Understanding:

- What has been done so far
- The factors that have lead to this situation

Research:

- You might not be the first to encounter this problem.
- Your research may lead you back to Stage 1 again

Stage 3: Peer Check

Possibly the most powerful resource

Describe the problem to a peer or colleague

- Clearly articulate what the problem is
- What you're trying to solve
- any difficulties you see

Why?

- gaps or gotchas will be exposed
- it might sound good in your head, but verbalising it exposes the weaknesses

Stage 3: Peer Check

Possibly the most powerful resource

What if I'm working alone?

- Write it down.
- Blog it.
- Tweet it.

- Even if no one reads it, you have a record of your thoughts.
  - Gives you a point of return if you get lost

- Talk to your manager (!)

Stage 4: Nature of Problem

The nature of the problem will guide you toward a methodology.

Loosely Defined Problem

- Broad, non-specific goals
- Ideal-based
- Experimental / Trial / Future Projects

Stage 4: Nature of Problem

The nature of the problem will guide you toward a methodology.

Tightly Defined Problem

- Specific goals
- Target-based
- Production ready, workflow style systems

Problem understood

Now how to solve it

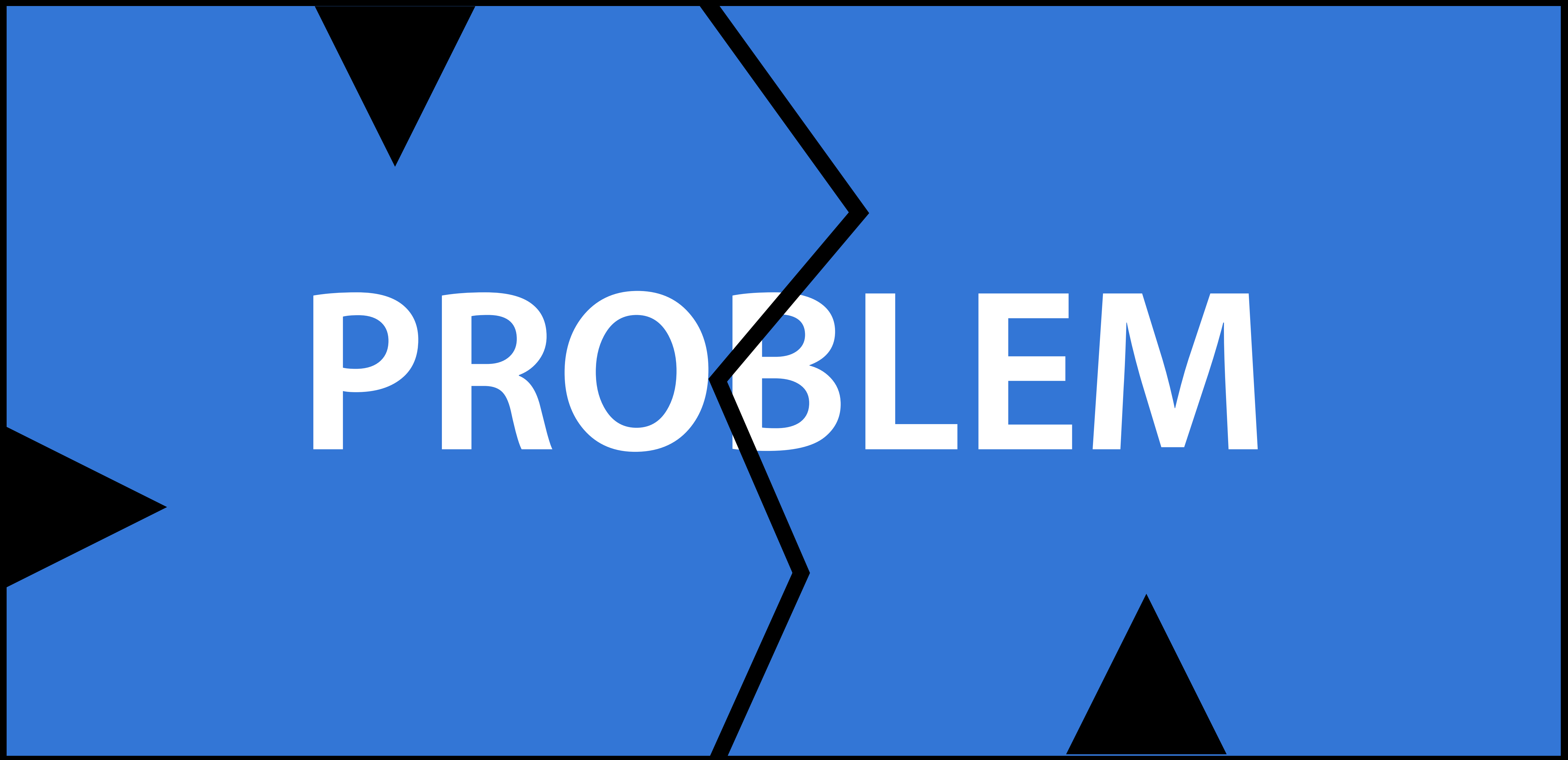
We have a big lump of a problem

PROBLEM

Problem understood

Now how to solve it

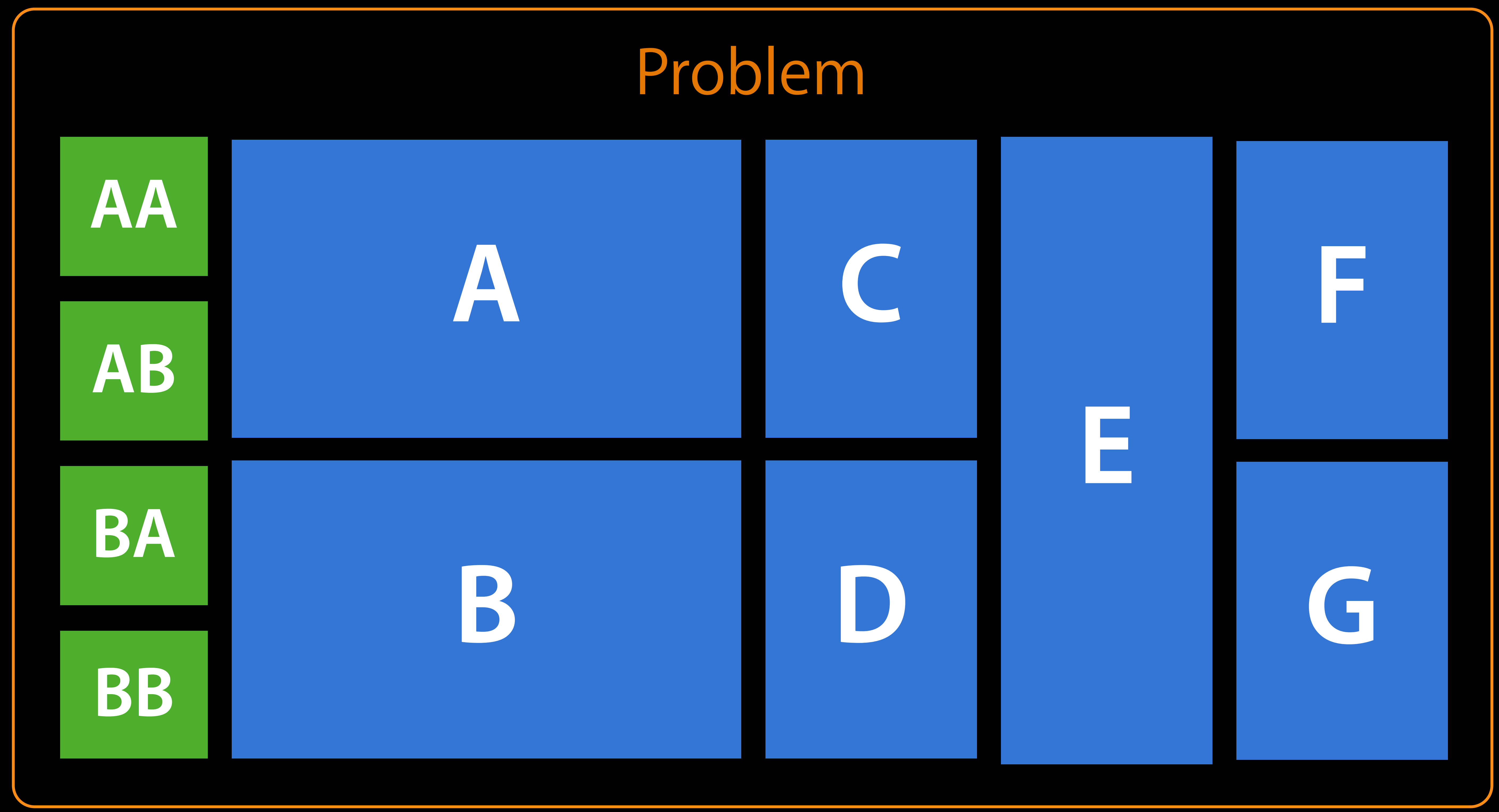
We could chip away at it, and may get somewhere if we're lucky.



To effectively solve any problem:

Break it up

Break it up



Stage 5: Break it up

A big problem is hard to solve

Smaller chunks are easier to solve

- a piece or chunk is far more workable
- each piece may have specific but different requirements
- completeness (individually solved = collectively solved)
- can be delegated or allocated

A Piece or Chunk is likely to be

- an activity or task
- attribute or category

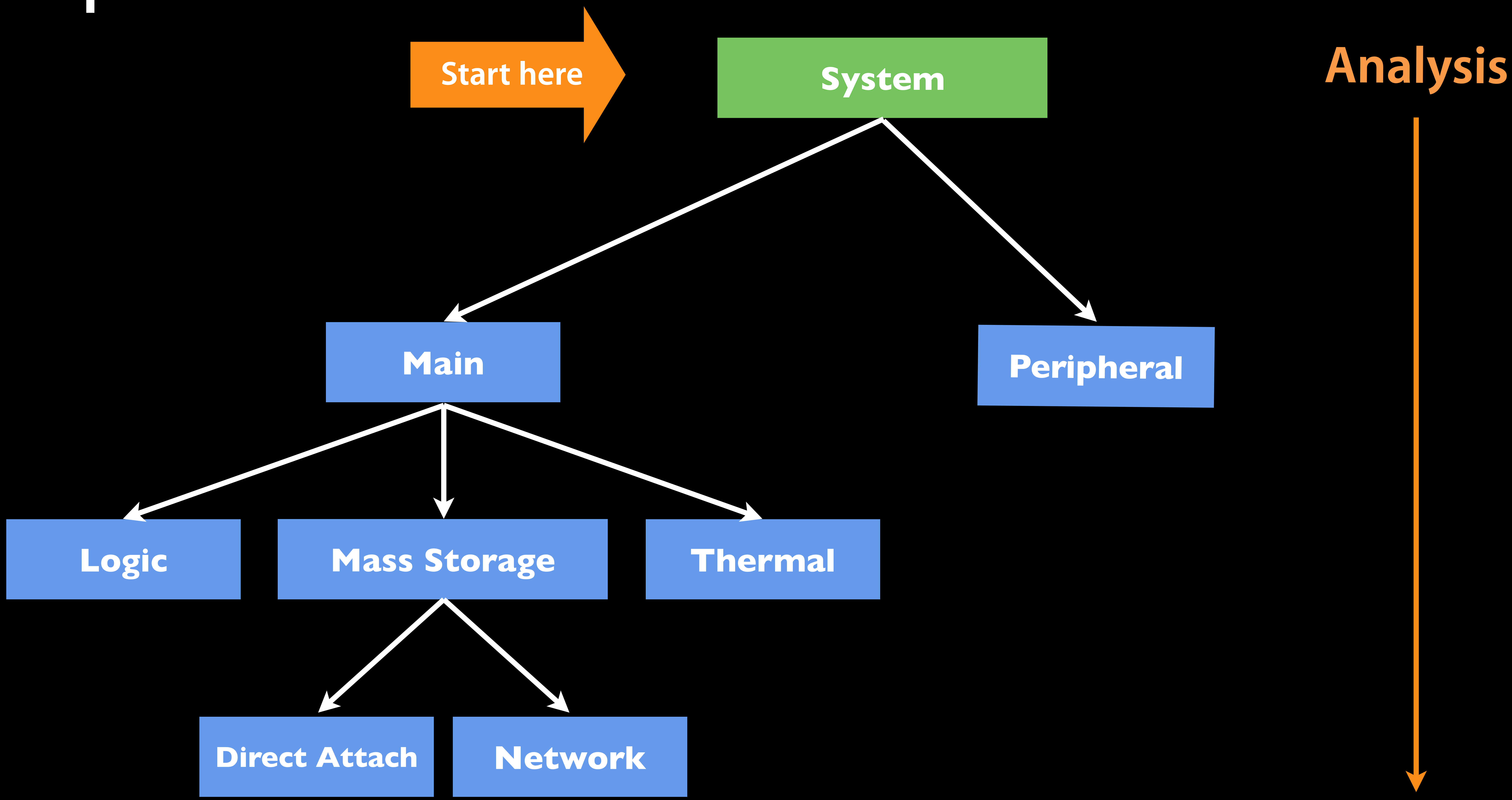
Top - Down Method

Tightly Defined Problem

Top-Down Analysis:

- Start at highest level of system
- partial understanding of sub-technologies
- You know what you want from a solution
  - maybe not at module or piece level

Top - Down


Start here
System
Main
Peripheral
Logic
Mass Storage
Thermal
Direct Attach
Network
Analysis

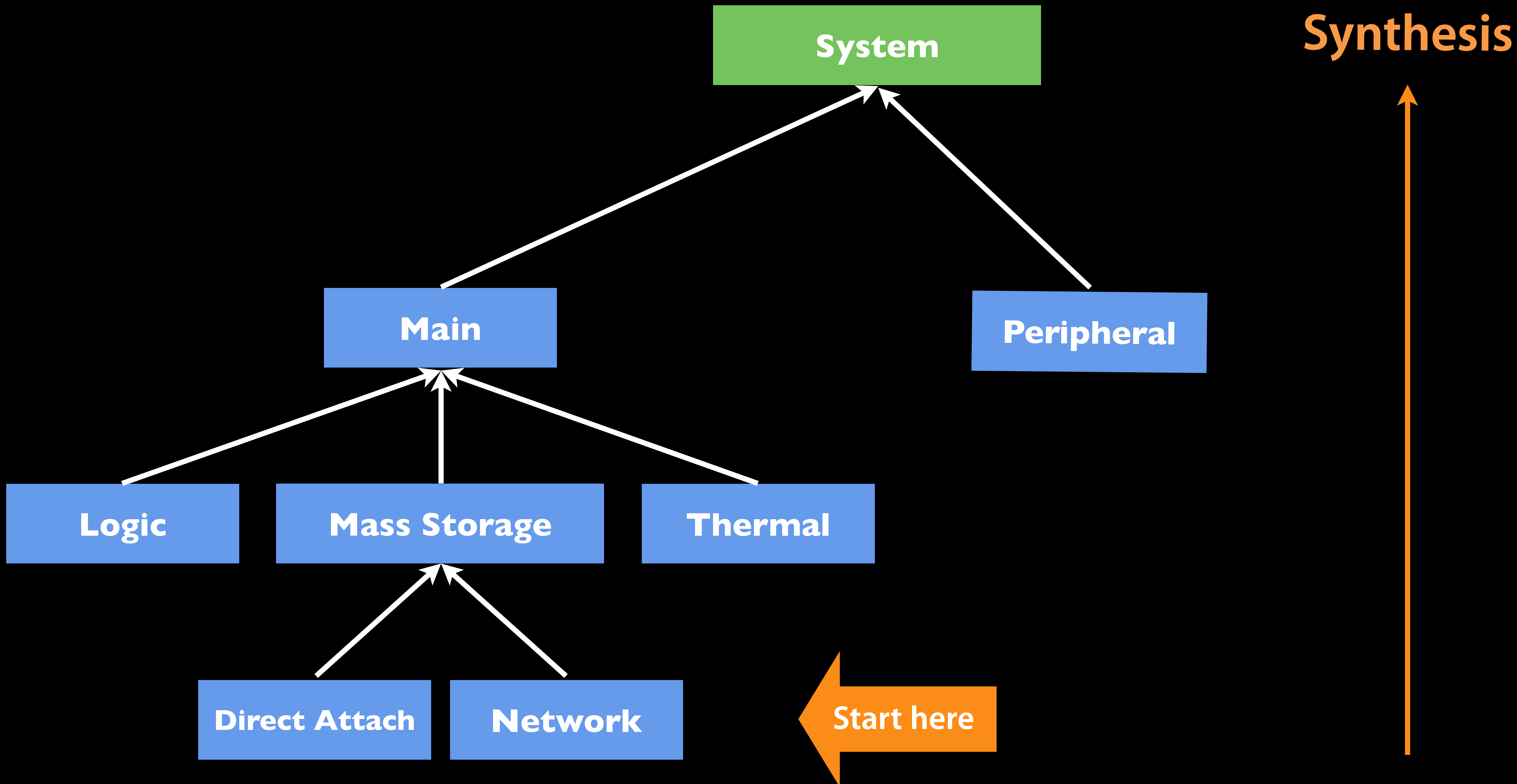
Bottom - Up Method

Tightly Defined Problem

Bottom - Up Synthesis:

- Start at lowest level of system
- Individual modules collectively build the system or solution
- You understand what is happening at module level,
  - unsure on individual relationship to whole

Bottom - Up



System
Synthesis
Main
Peripheral
Logic
Mass Storage
Thermal
Direct Attach
Network
Start here

Finding the Pieces

Order in chaos

Ways 'pieces' of the problem become obvious (things to look for):

- Natural Grouping
- Functional or Procedural Grouping
- Modular
- Derived from First Principles or Architecture

Funnel Method

Loosely Defined Problem

Recall:

- Broad, non-specific goals
- Ideal-based
- Experimental / Trial / Future Projects

- The problem may not be fully understood, and solution options are completely unknown.

Funnel Method

Loosely Defined Problem

Inputs:

· new or unproven Ideas

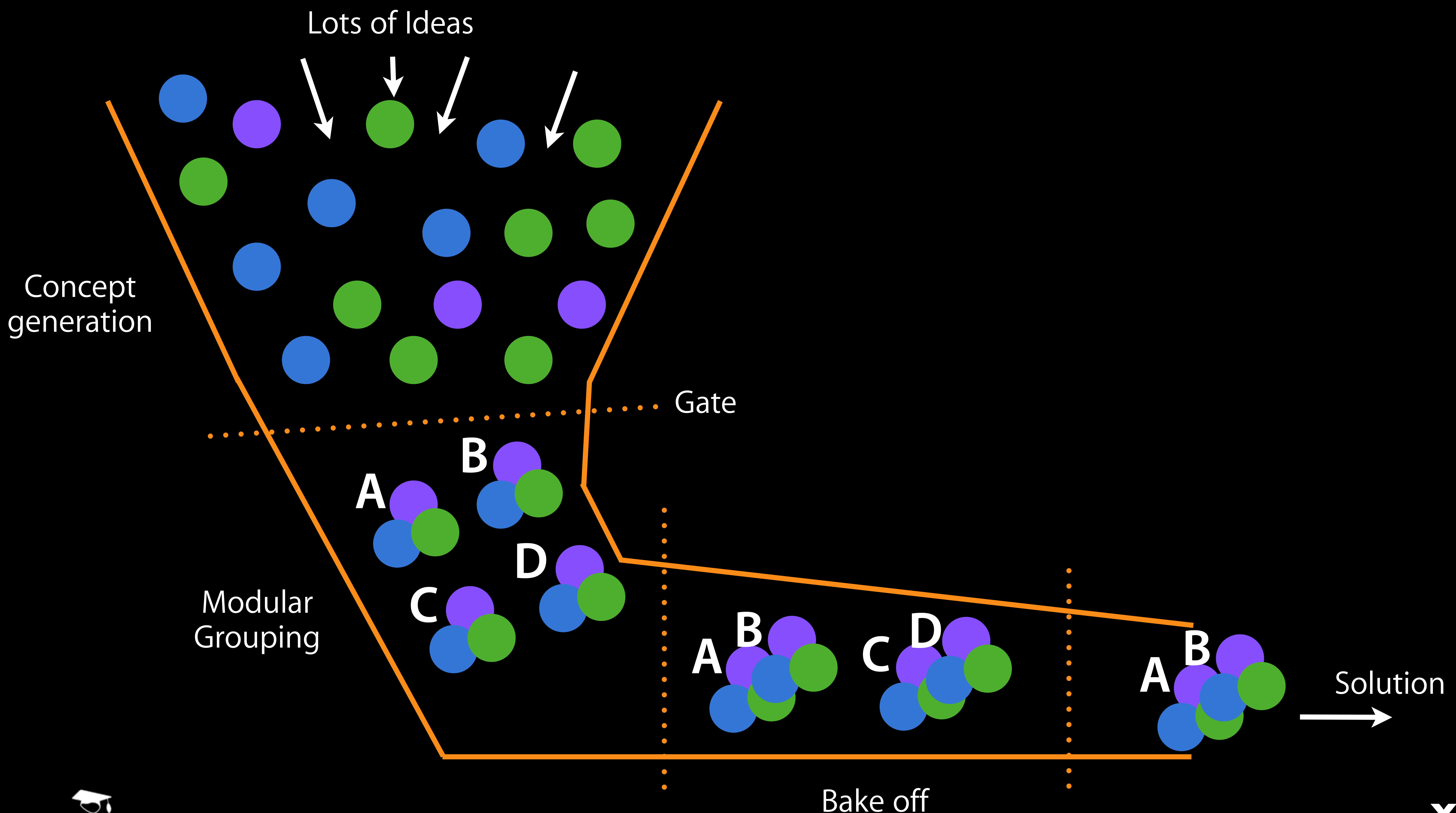
· parallel prototyping (project bake-off)

· experimentation and discovery

Output:

- Evolutionary goal
- The best solution (progressive)

Funnel Method



AUC

XWII

Group Solve

Group Solve

Solve for X

- Likely to encounter this scenario in your organisation

- Problems progressively revealed as you traverse the scenario

- individually / pair up & think of the problem
  - and how you might start to solve it
  - modules / categories / attributes

Scenario

< scenario removed >

Why Problem Solving Hurts

Ouch

- If it was easy, you'd have solved it already
- It typically involves **learning** new stuff, while simultaneously developing a solution
- Chances are you will not immediately know the answer.
- You're under pressure.

Constraints

Fixed vs. imposed Constraints

- Some constraints will be fixed and are physically determined.
  - ie. Cable breaking strain of 1200KG
- Other constraints are imposed or we unintentionally limit ourselves with prior convention.

Think outside of the problem as well.

- is the problem part of a bigger picture?

Consider this

Imposed Constraint

AUC

XWII

Consider this

Down under (& NZ too) is on top

No! It’s all wrong.

Why?

N

Someone decided North goes at the top.

No Problems

I’m awesome, No problems here.

... yet

Discover weaknesses in your systems

- use same approaches
- module by module analysis
- understand what ‘normal is for your system’
- understand utilisation and capacity
- If you do have a problem, you’ll know how each module normally behaves

Part 2. Troubleshooting

Troubleshooting Concepts

What is Troubleshooting?

Dictionary says...

troubleshoot |ˈtrəbəlˌ sh oōt|

verb [intrans.] [usu. as n.] (**troubleshooting**)
solve serious problems for a company or other organization.

– trace and correct faults in a mechanical or electronic system.

What is troubleshooting?

Applied Problem Solving

Inherit: Problem Solving methods

It's reusable

Core points retained

- Define what the issue is
- Understand what you are trying to fix
- Break the issue down into smaller parts

Types of Failure

3 Common Types

Technical Failures usually fall into three top level categories

- **Bogus** (there is no failure)
- **Outright** (it's dead)
- **Intermittent** (the most problematic)

Influences

Influences on Troubleshooting accuracy

- Quality of Symptom description
- Symptoms often do not have a 1:1 correlation with failure mode
- Data may be incorrect

How not to fail

The most important part

Symptom Description

- An accurate and concise **Symptom Description** is critical to your troubleshooting success

- Without an accurate Symptom Description
  - You'll be chasing the wrong thing
  - It'll be unclear where to start

Symptom Description

It’s easy to spot a bad one

It’s dead.

It doesn’t work.

There’s something wrong with my computer.

I can’t download the internet.

A System

and its parts

Any 'System' is a collection of modules

- It's normally a module that breaks, not the entire system

- A web server is a system - I/O, network, authentication, db, content, config
- A washing machine is a system - pump, motor, controller, valves, sensor

Accurate Troubleshooting

Report of System Failure

Verification or Replication of fault

where there is an actual, verifiable fault

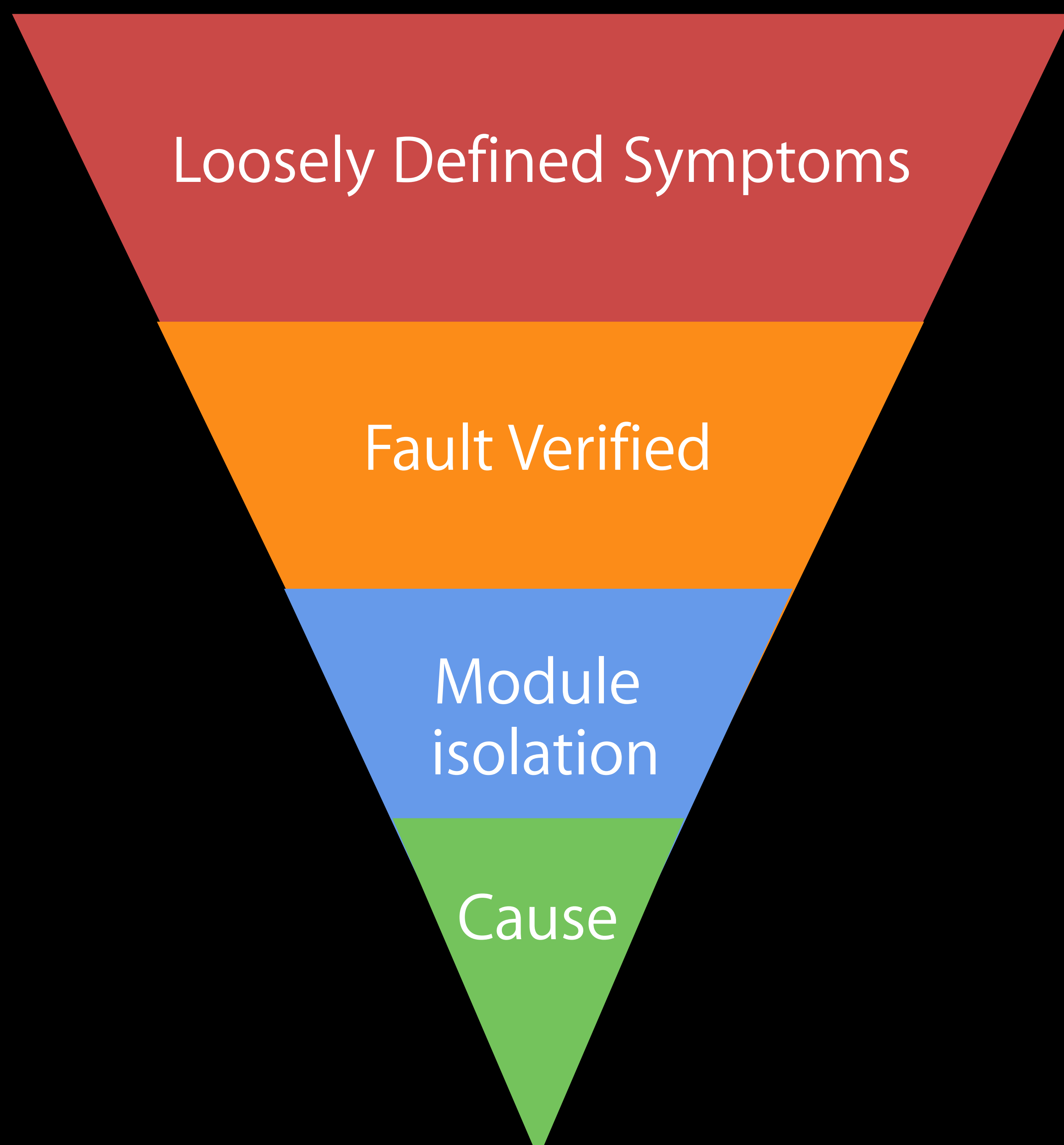
locate the faulty module within system

Fix **only** the faulty module or part

Return Correctly functioning system to operational status

What is Troubleshooting

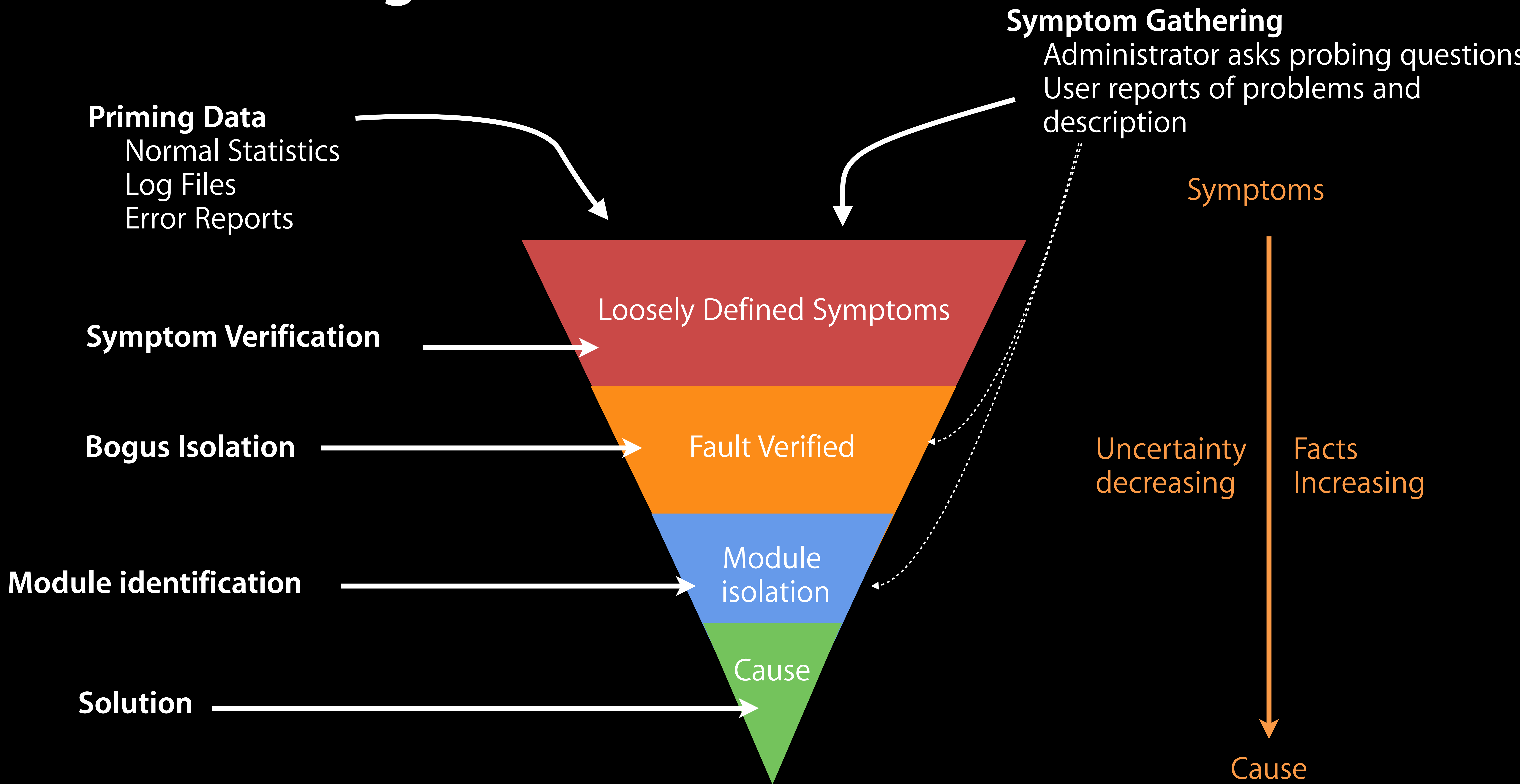
Sequential Fact Building



Progress through the troubleshooting process should

- reduce the uncertainty
- progressively isolate the modules
- increase the number of known states

Fact Building



Feedback Concept

We like to know whats going on

Humans like feedback in the form of progress.

We like to know that our interactions are changing the environment we are attempting to influence.

It gives us the sense of "getting somewhere".

Feedback Concept

Managers are human too

Managers are human too (!)

Uninformed managers can become a larger problem than the technical issue you are trying to resolve.

Feedback Concept

Keep it in mind

When determining the steps you are going to take in your troubleshooting task:

- keep in mind the result you are looking for at each step
- and what result a normal, correctly operating module would return.
- If you have progressive results, you can keep others informed.
  - ie, we're ruled *X* out, established *Y* is working, need to test *Z*.

Why Feedback Matters

Consider this

A theoretical moving car

Input → Process → Output

Steering Angle — Wheels turn — Change in Direction

Feedback:

Visual Recognition
Sensory Feedback (g-force)

Feedback Delayed

Feedback altered

A theoretical moving car

Input → Process → Output

Steering Angle | Wheels turn | Change in Direction

Feedback:

30sec

Visual Recognition
Sensory Feedback (g-force)

Feedback Removed

Feedback altered

A theoretical moving car

Input → Process → Output

Steering Angle | Wheels turn | Change in Direction

Feedback:

Visual Recognition
Sensory Feedback (g-force)

Oh no!

You crashed and burned.

Why?

- Multiple wrong inputs
- Situation becomes progressively worse
- progress is unknown

Each Troubleshooting stage should result in usable information.

- Even if that is “this part works as expected”.
- You now have one less module to isolate.

Troubleshooting Methodologies

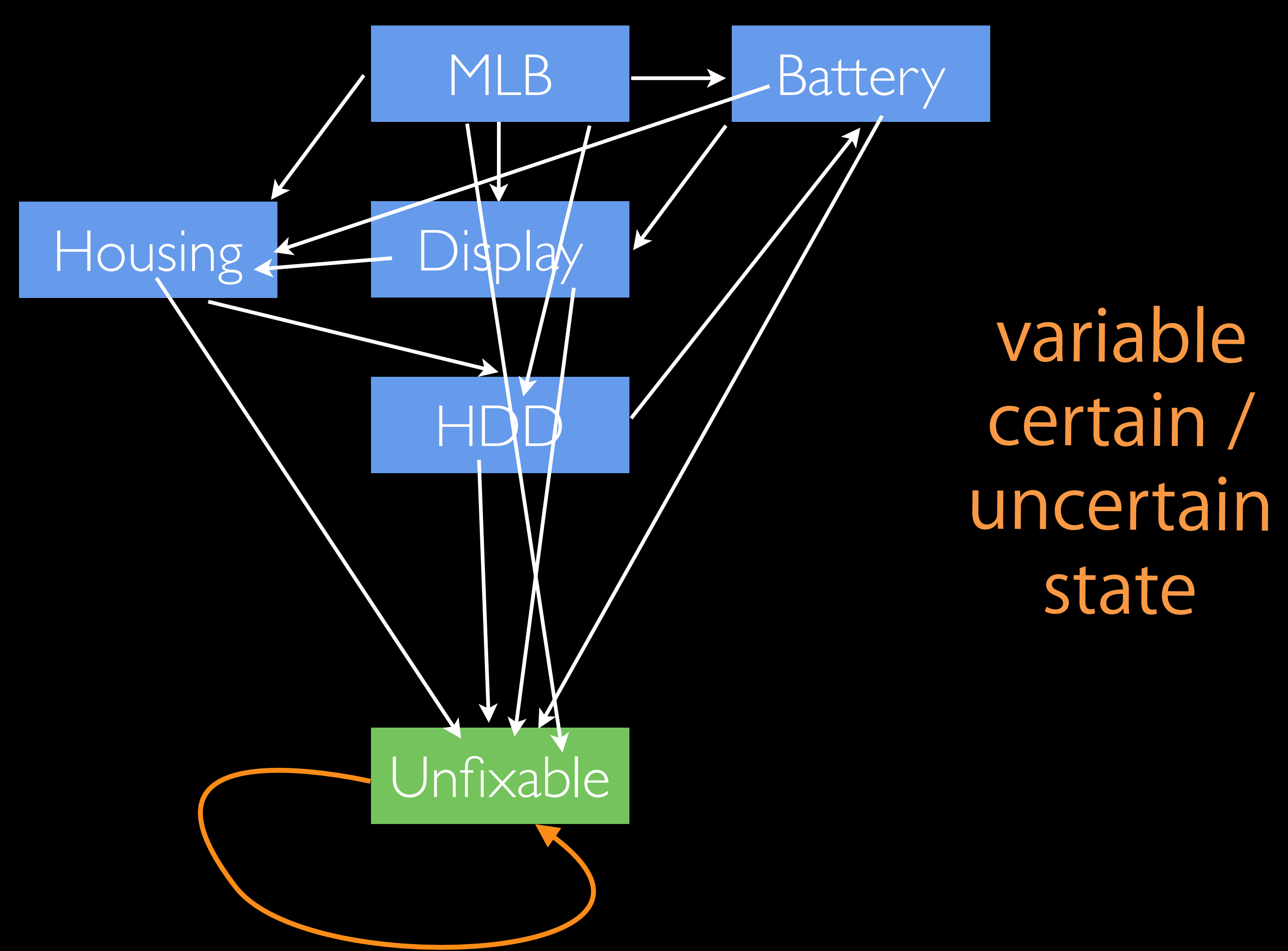
Gather info and verify

First Steps

- Gather info
- Verify situation against information
- Establish a baseline of a correctly operating system
- Rule out really obvious factors
  - Storage full, No IP address, No AC input, etc.

Brute-Force Guesswork

Troubleshooting Methodologies

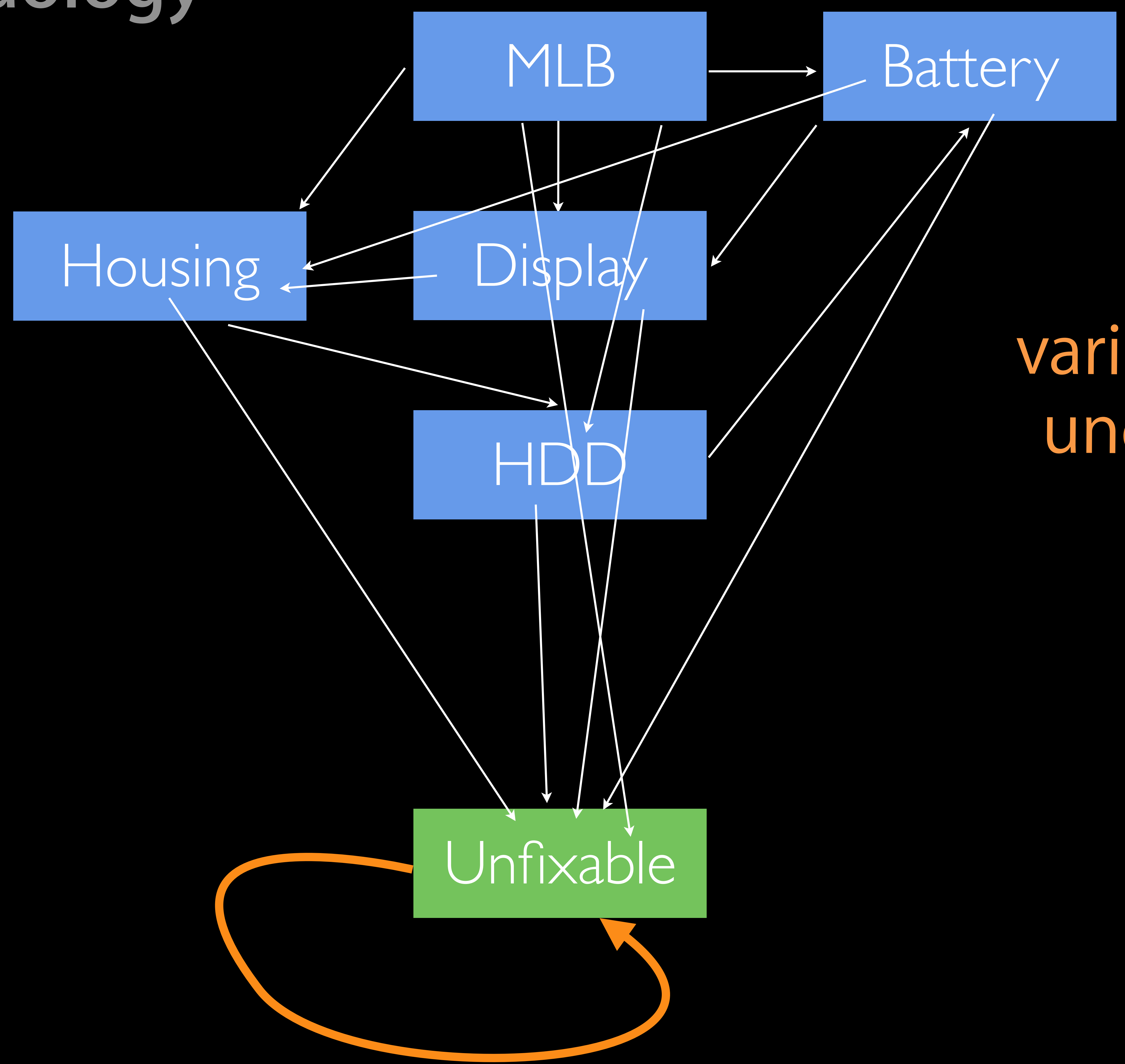


Brute-force Guesswork

- Belief based
- Evidence poor
- Procedurally inadequate
- highly uncertain if correct cause identified
- occasionally works for some experienced techs. Common cause of "it **must** be this part".

Brute-Force Guesswork

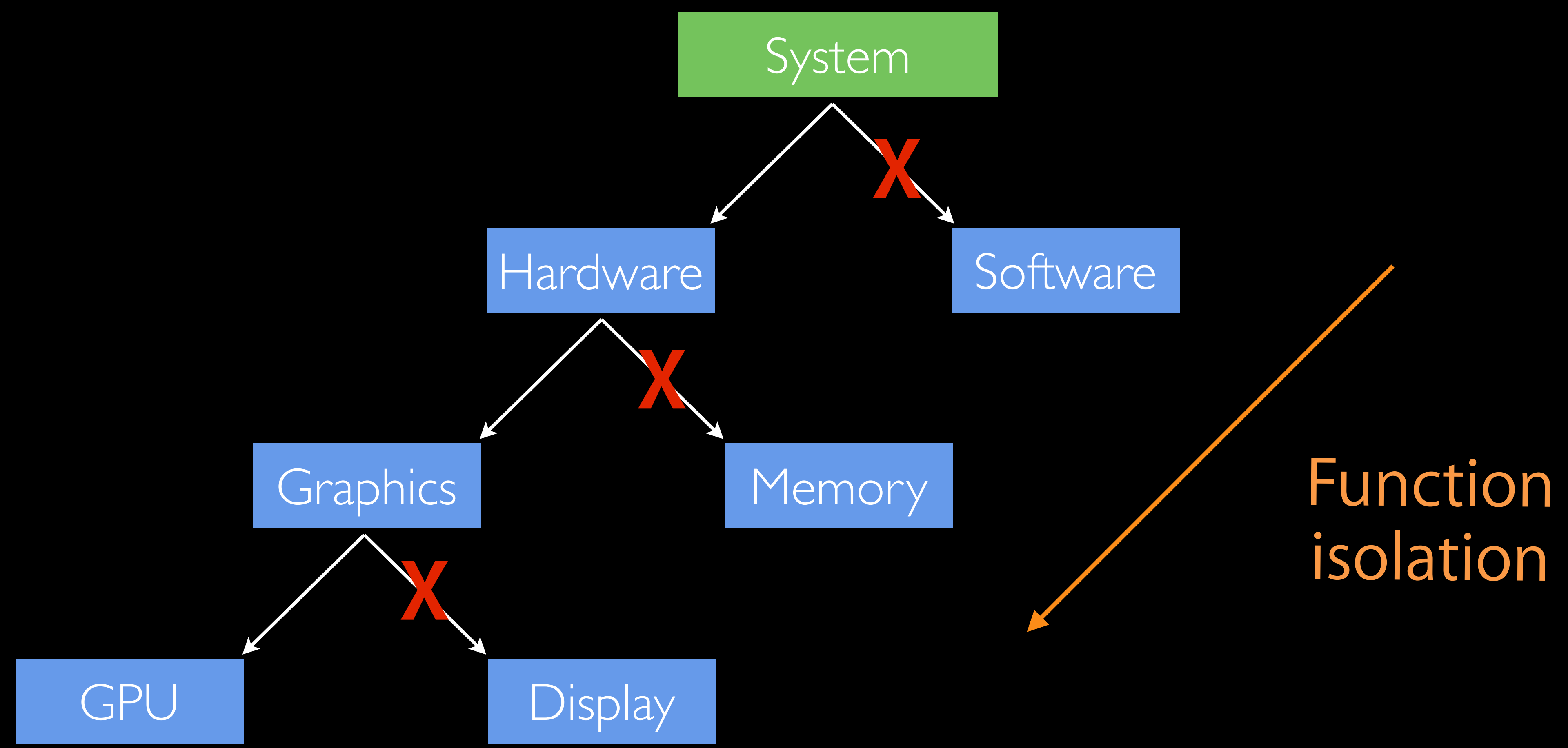
Methodology



variable certain / uncertain state

Split-Half

Troubleshooting Methodologies

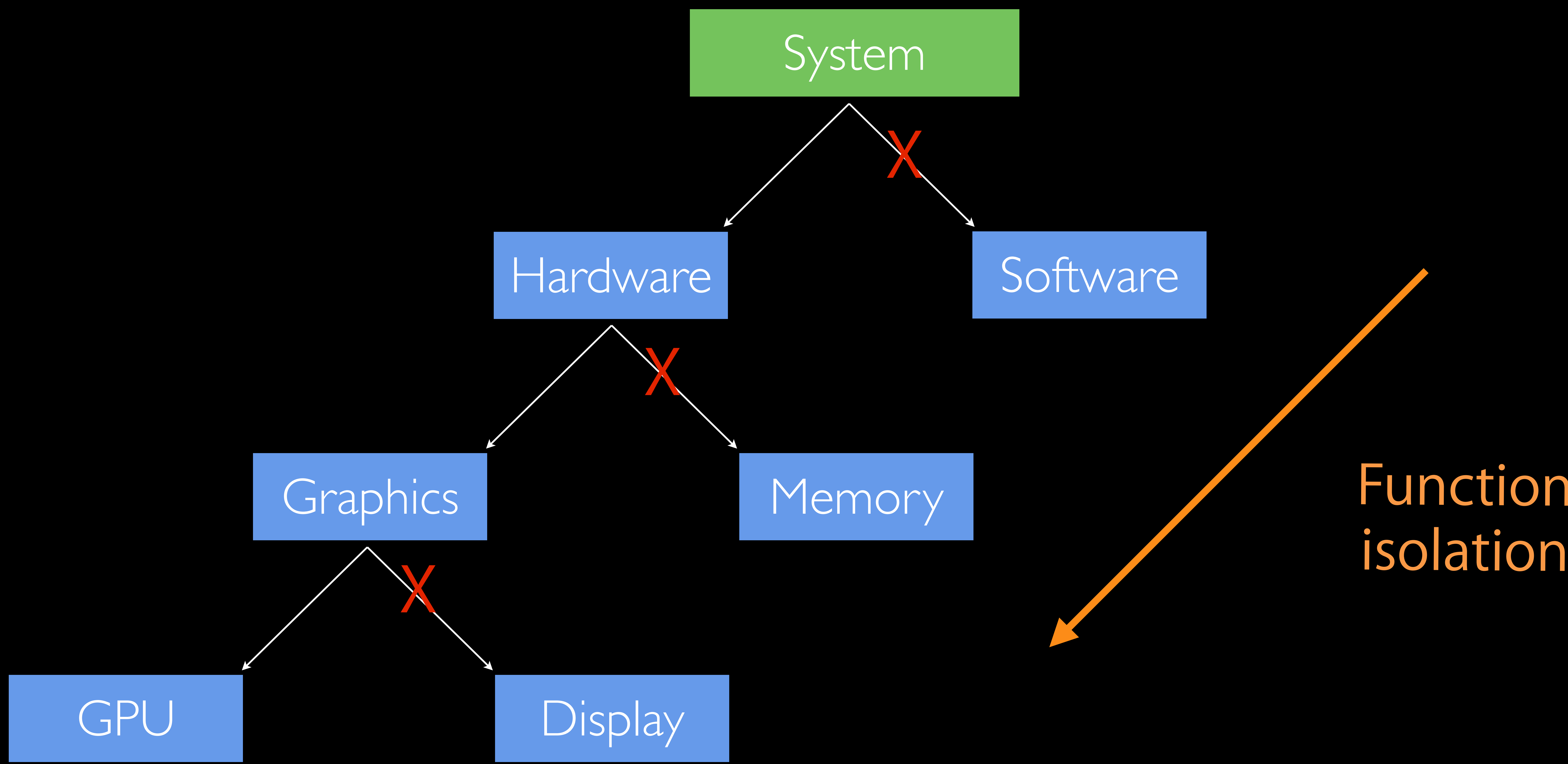


Split-Half

- Eliminate half of the probable cause at each level
- Requires understanding of common issues
- Requires understanding of core functions of each function area or differentiating behaviour
- highly structured, complete but can be time consuming and indirect if starting point is vague.
- Works best for isolate/verify function areas where there is no obvious likely cause

Split-Half

Methodology

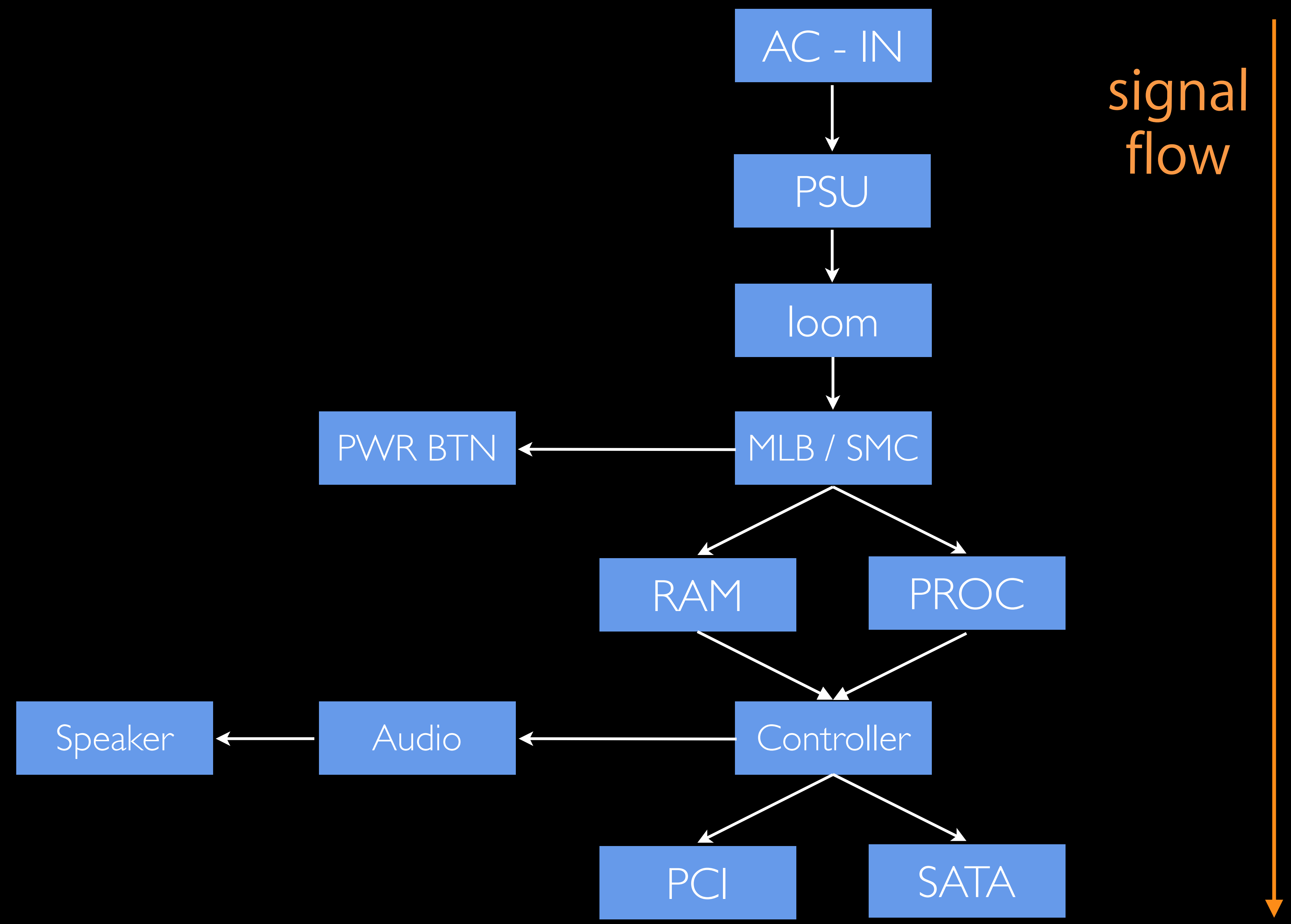


AUC

XWII

Power / Signal Flow

Troubleshooting Methodologies

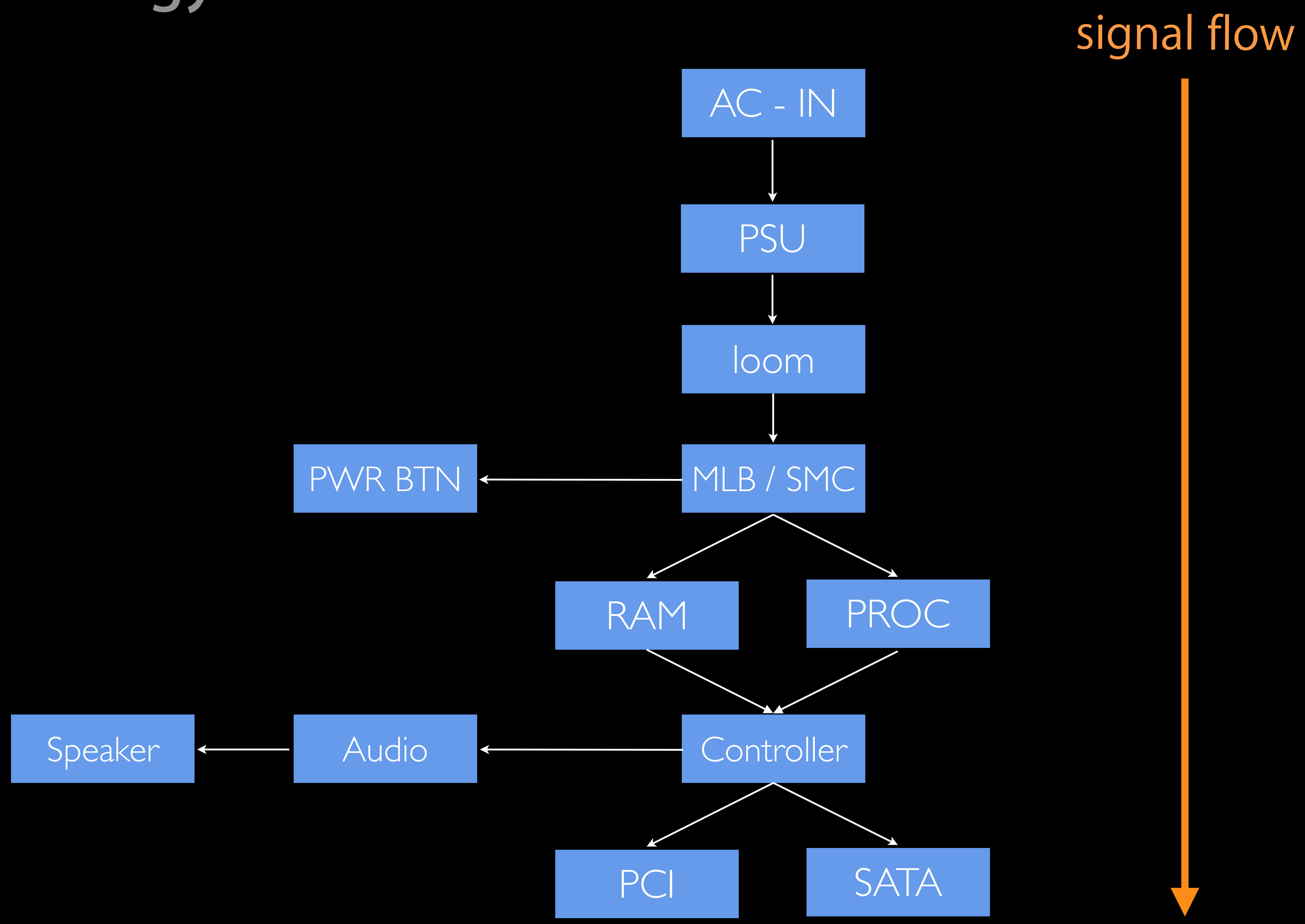


Power / Signal Flow

- Follow Signal sequence through system
- Highly sequential, must be performed in order
- effective for "no X" or "dead" symptoms
- often places core modules early in the troubleshooting, even if they may be a less likely cause.
- Requires understanding of signal flow in system architecture.

Power / Signal Flow

Methodology

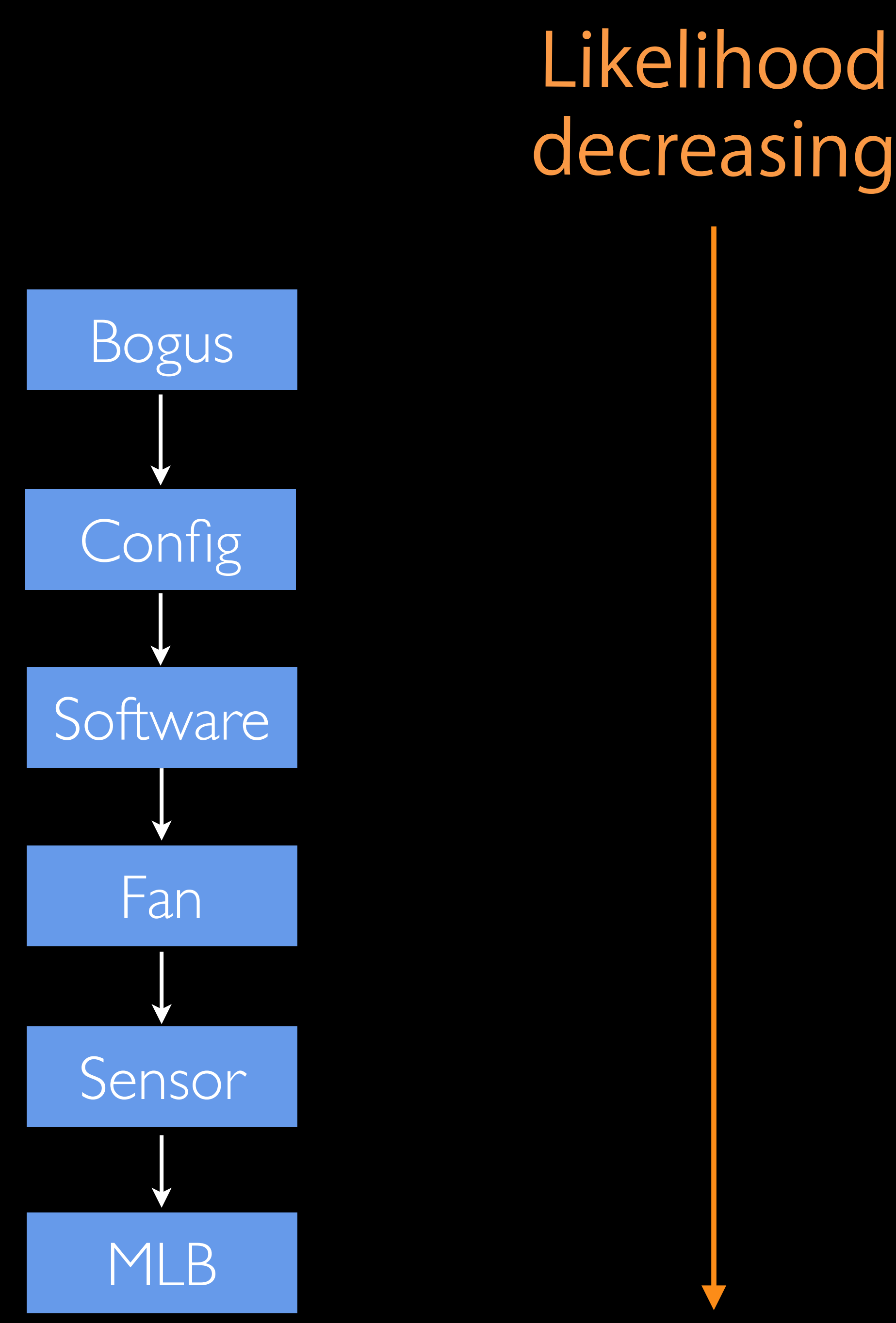


AUC

XWII

Likely Cause

Troubleshooting Methodologies

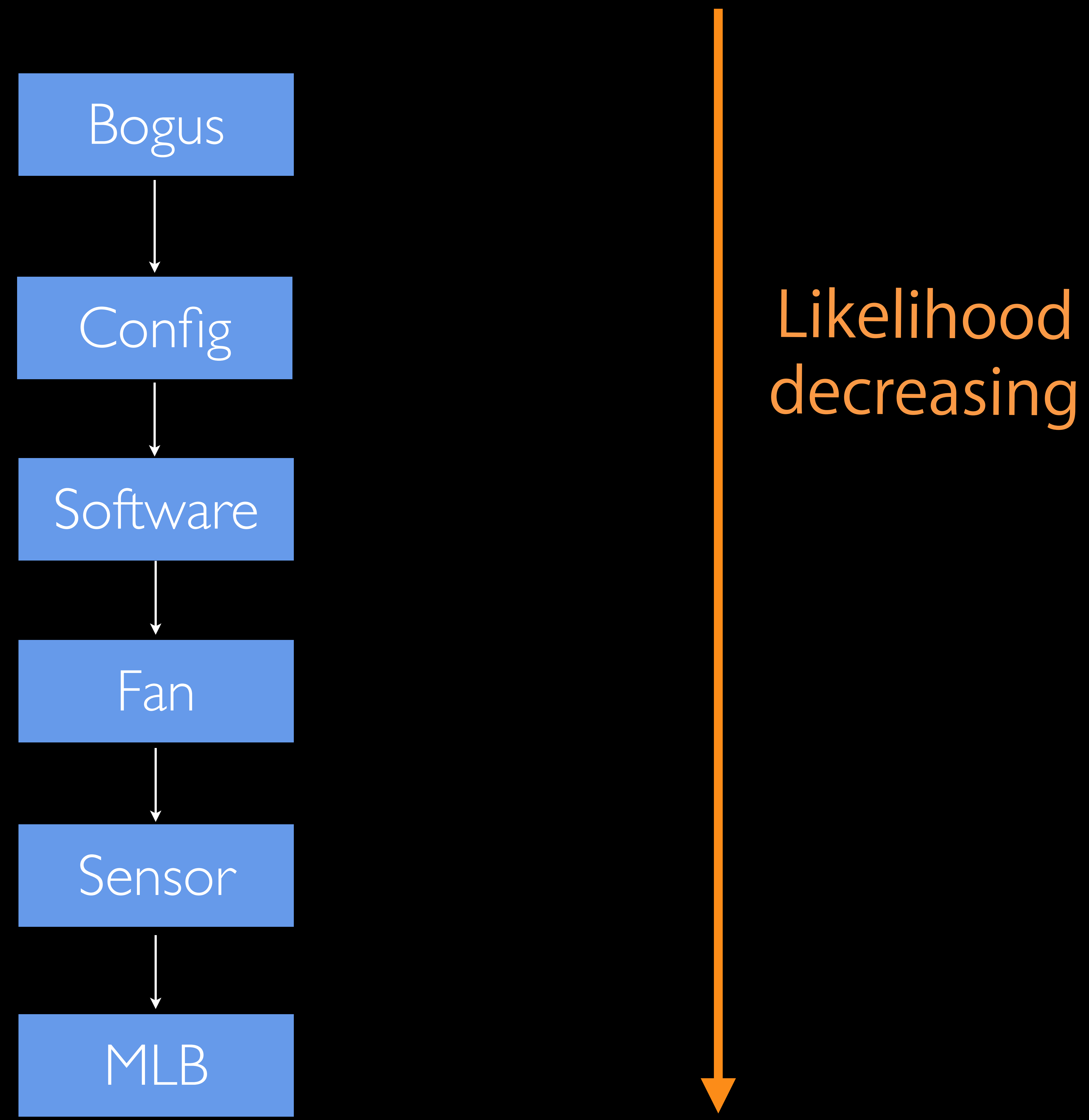


Likely Cause Identification

- Use known likely causes as starting point
- can often be reordered to promote more likely causes, demote less likely cause
- works best where
  - it is possible to identify all sources of possible cause
  - there are few causes
  - or the causes are well known
- less suitable for cases where there is no obvious cause

Likely Cause

Methodology



Likely Cause + Weighted Matrix

Troubleshooting Methodologies

Order	Possible Cause	Likelihood		Possibly Bogus	Isolation Priority
1	Possible Cause A	High	HIGH	Yes	High, Dependencies
2	Possible Cause B	Low	MID	Yes	High, Dependencies
3	Possible Cause C	Low	LOW	No	Low

Weighted Matrix

- Use to assist prioritising the Likely Cause isolation order
- Promotes more likely / relevant isolation tests for the scenario
- Demotes less likely causes
- Use to correctly "weight" troubleshooting priority.

Likely Cause + Weighted Matrix

Methodology

Possible Cause	Likelihood	Possibly Bogus	Isolation Priority
Possible Cause A			
Possible Cause B			
Possible Cause C			

Likely Cause + Weighted Matrix

Methodology

Possible Cause	Likelihood	Possibly Bogus	Isolation Priority
Possible Cause A	High	Yes	High, Dependencies
Possible Cause B	Low	Yes	High, Dependencies
Possible Cause C	Low	No	Low

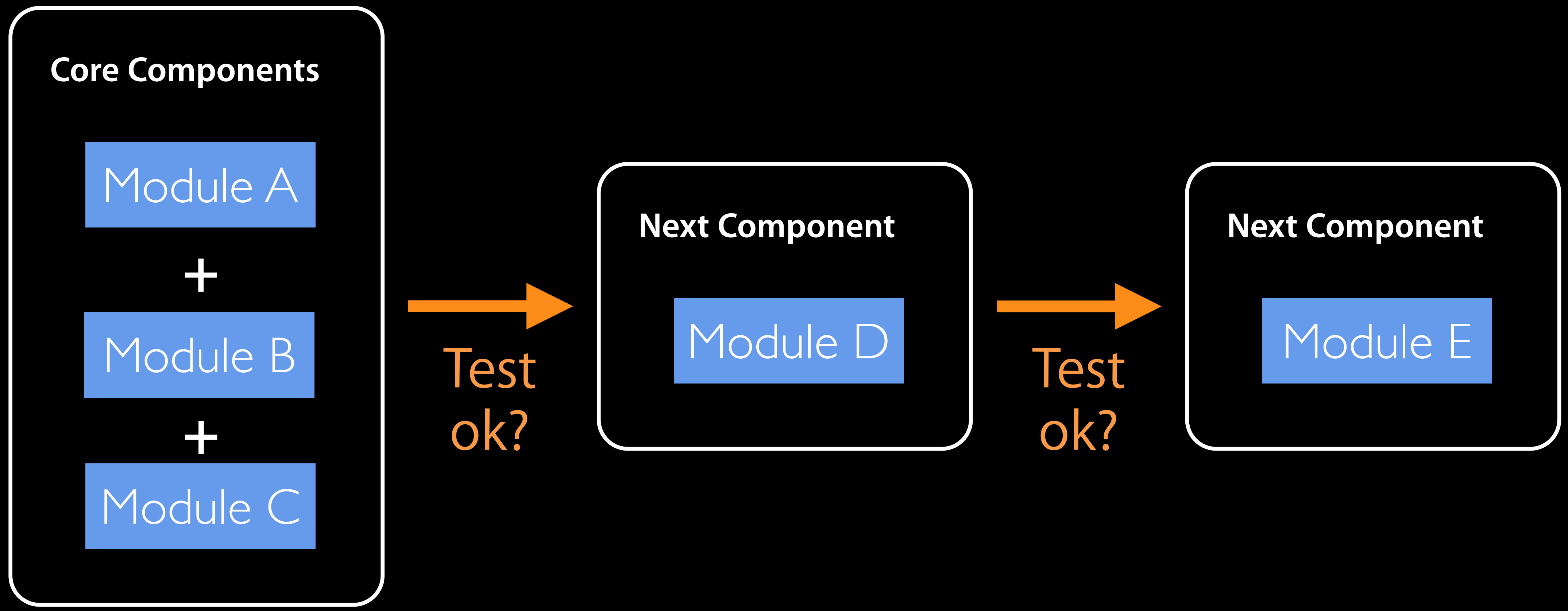
Likely Cause + Weighted Matrix

Methodology

Derived Order	Possible Cause	Likelihood		Possibly Bogus	Isolation Priority
1	Possible Cause A	High	HIGH RANK	Yes	High, Dependencies
2	Possible Cause B	Low	MID RANK	Yes	High, Dependencies
3	Possible Cause C	Low	LOW RANK	No	Low

Minimal Config

Troubleshooting Methodologies



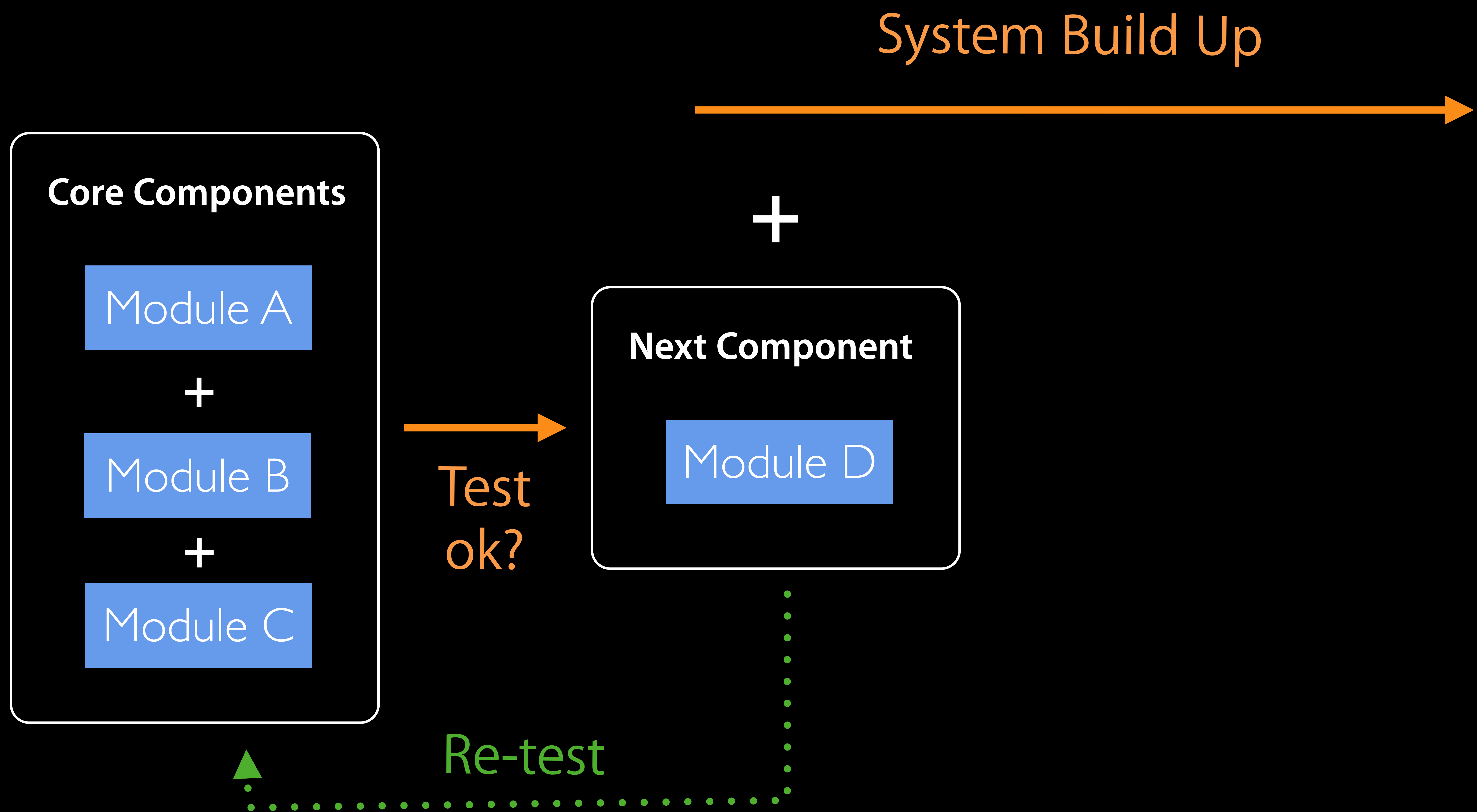
Minimal Config

- The Final Frontier
- Saviour when all else fails
- Highly time consuming,
- but high accuracy

- Must know what components are the absolute minimum for the system start

Minimal Config

Methodology

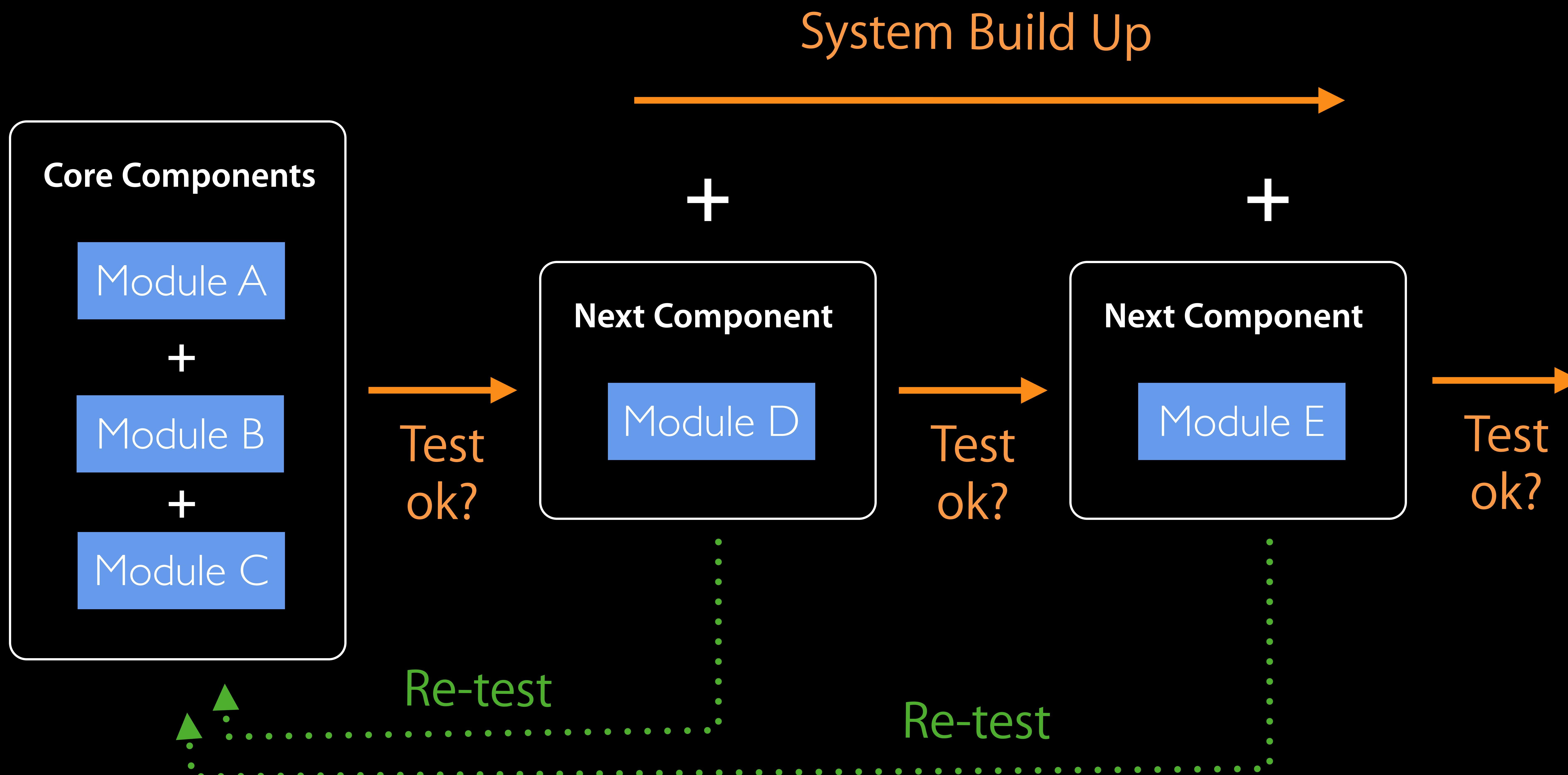


AUC

XWII

Minimal Config

Methodology



AUC

XWII

No Single Answer

Select-a-method

- No single method works for all types of symptoms or fault
  - complexity
    - simple, tightly correlated symptoms
    - complex, loosely correlated symptoms
  - nature of failure
    - electrical, mechanical
    - runtime, configuration, design, capacity
    - Intermittent

Known Good

Troubleshooting Methodologies

Known Good modules are modules, code or some other component that is known to be operating correctly.

It's often called "KG" or "golden".

For core components, you may need to use a KG module OR have a good understanding of the expected behaviour of the core modules.

... but they really need to be "good" or "golden" or you'll prime your troubleshooting for failure.

Tools To Help You

They’re often right there.

- Console (logs, would you believe have heaps of info!)
- Activity Monitor
- top & ps
- fs_usage & lsof
- iostat
- sc_usage & dtrace
- netstat
- wireshark
- rubbish webmin interface on your switch / fabric / CSS / FC array

Group Troubleshoot

Group Troubleshoot

Scenario

- Less likely to encounter this situation in your organisation

- You might not know all of the technology involved. Use first principle knowledge of IT systems to identify modules

- individually / pair up & think of the problem
  - and how you might start to solve it
  - modules / categories / attributes

Group Troubleshoot

Scenario

< scenario removed >

Workarounds

Where it's not something you can fix

Occasionally, there will some some issues you have isolated to a cause that you cannot directly fix.

For Example, a software bug.

- Using your troubleshooting results, you'll know where it's failing
- Use this information to develop a workaround until a permanent fix is available
- Report the bug to the product vendor or manufacturer
- When the fix is available, you'll know how to correctly verify its operation

X
World 2011