Continuous Evolution Template

Hypothesis driven approach to avoid guesswork

@harikrishnan83

Solutioning Exercise

Ice-Breaker

Problem -> Solution

Problem Statement –

“Search nearby hotels is about 10x slow during our peak hours”

Developer –

“Let us cache the results for popular locations to avoid hitting the DB”

Is the above solution

- Valid?
- A Quick Fix?
- An Educated Guess?

What is the issue with “Guess Work”?

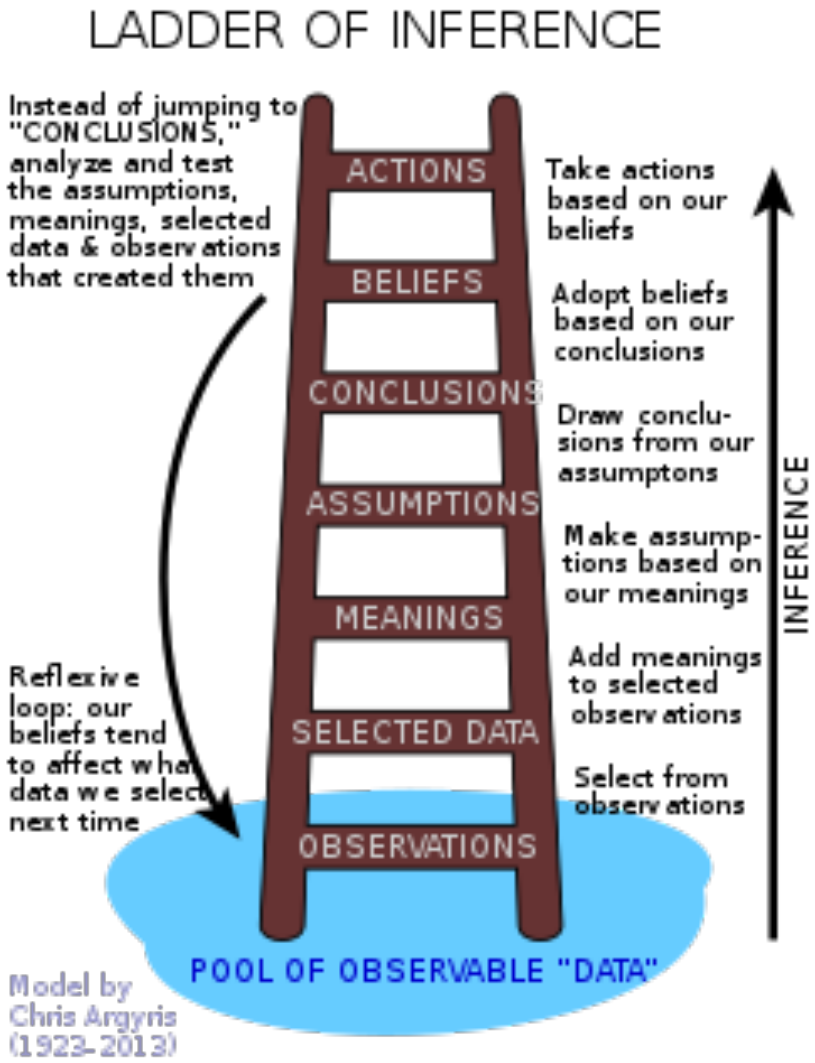
- Assumptions
  - Cause - We seem to have concluded that DB is the cause of the problem.
  - Result - Are we sure that the cache will help? If yes, by how much? Is it possible the problem may get worse?
- How did we get here? Where are we going with this?
  - No insight into why we arrived at this solution.
- Problem Solved != Solved Right
  - Assuming this works, are we sure this solution is just enough? Not less not more.
    - Example: Why solve for 100 MS when the acceptable load time is 300 MS?
  - Are we introducing side-effects.
    - Example: Cache Invalidation

Pressure of getting it right

- Solutions are expected to resolve the problem.
- Credibility on the line
- Release anxiety

Problem -> Solution

Are we jumping too quickly?



Let us try this one more time

Problem Statement –

“Search nearby hotels is about 10x slow during our peak hours”

Developer –

“*Based on our analysis of all the steps in page load process*
the SQL search query is the slowest step (takes an extra 2500 MS at peak time),
By adding a Geo-spatial index,
we may be able to reduce page load time to 500 seconds during peak hours”

What is different about this statement?

It is **not a Solution**. It is more of a Solution **Hypothesis***.

*Well almost, we can do much better than this.

Hypothesis

Researcher’s prediction of the study,

Which may or may not be supported by the outcome

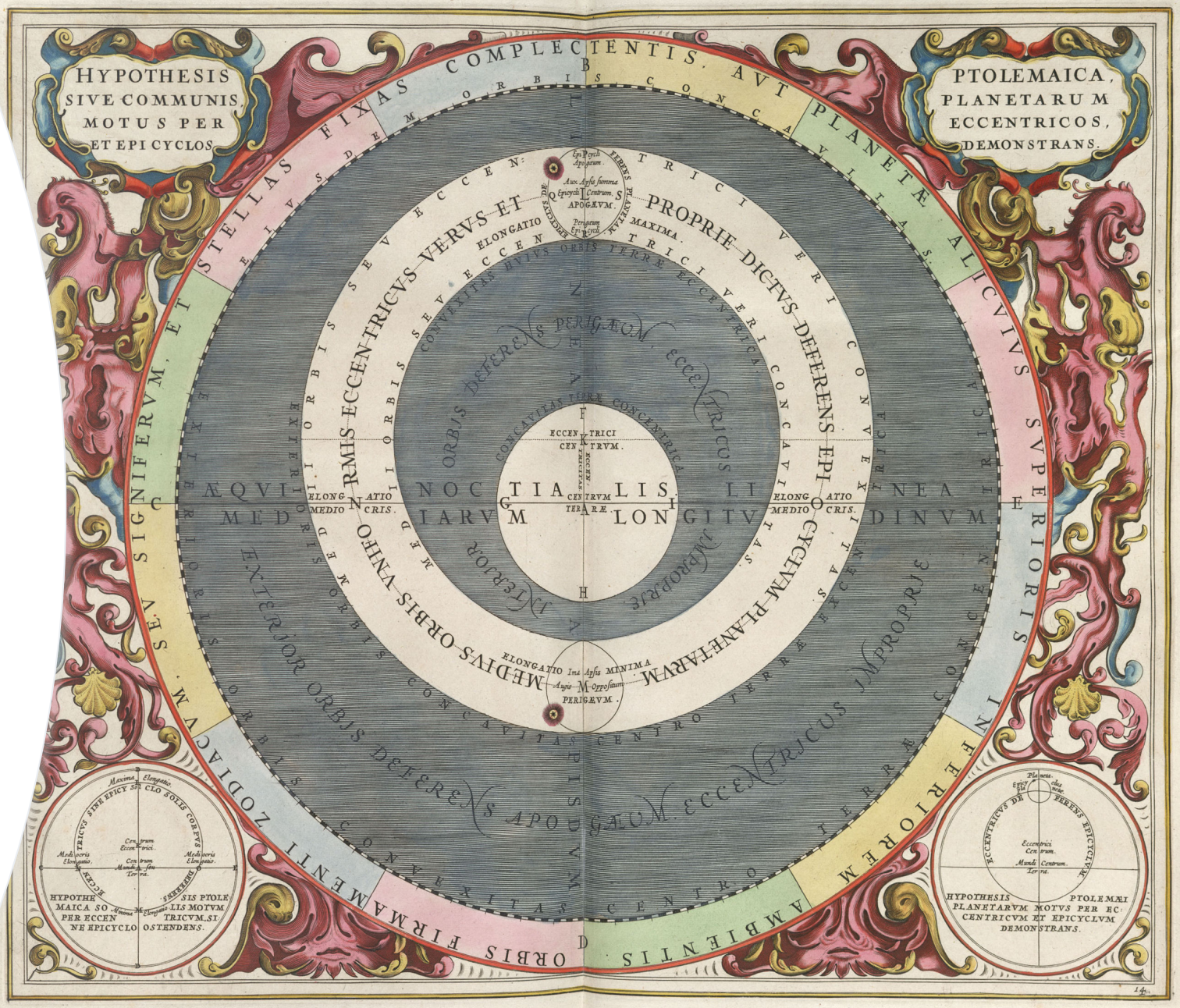


Image Attribution: By Andreas Cellarius - http://www.phys.uu.nl/~vgent/cellarius/cellarius_plates.htm, Public Domain, https://commons.wikimedia.org/w/index.php?curid=6839956

Problem -> **Hypotheses** -> Solution

Why is this two-step process to Solution better?

- **Okay to Fail** – By definition, the outcome may or may not match our prediction. We are only Hypothesizing.
- **Validated Learning** – Even a disproved Hypothesis is a valuable lesson.
- **Safety** - Encourages an environment we are happy to be proven wrong.
- **Weighing your options** – Instead of jumping at the first option (which may not even be the easiest), this approach facilitates us to generate several hypotheses and test viable options before narrowing on a solution. **g**

Authoring Great Hypotheses

Understanding the Problem Statement, Setting KPIs, Generating Hypotheses, Designing Experiments and arriving at Solutions.

Step 1 – Problem Statement

Clear Problem Statement

When there are around 200 RPS at Bookings API
We see lost updates in ticket status column in Tickets DB
The issue was established by cross referencing Bookings API logs

Operative Words

- **Context** – 200 Requests Per Second at Bookings API
- **Problem** – Lost updates on ticket status column in Tickets DB
- **Data** – Bookings APIs Logs latest request status and ticket status are not the same

Step 2 – Establishing the Cause

Hypothesis *- The recently introduced Kafka partitioning*
May be leading to out of order deliver of Kafka messages
Because of which we see lost updates in tickets DB.

Experiments

- **Disprove** – After removing partitioning, problem goes away
- **Prove** – At 200 RPS we see messages pertaining to a single ticket id going into more than one partition

Why should we do this?

- If we can disprove this hypothesis, or we are unable to prove it, we can try others
- We get to learn that Kafka partitions have been designed correctly

Step 3 – Solving the Problem

Hypothesis *- By adding ”ticketId” as the partition key,*
We can expect Kafka to deliver the messages in order,
Which should let us go beyond 200 RPS without lost updates

Experiments

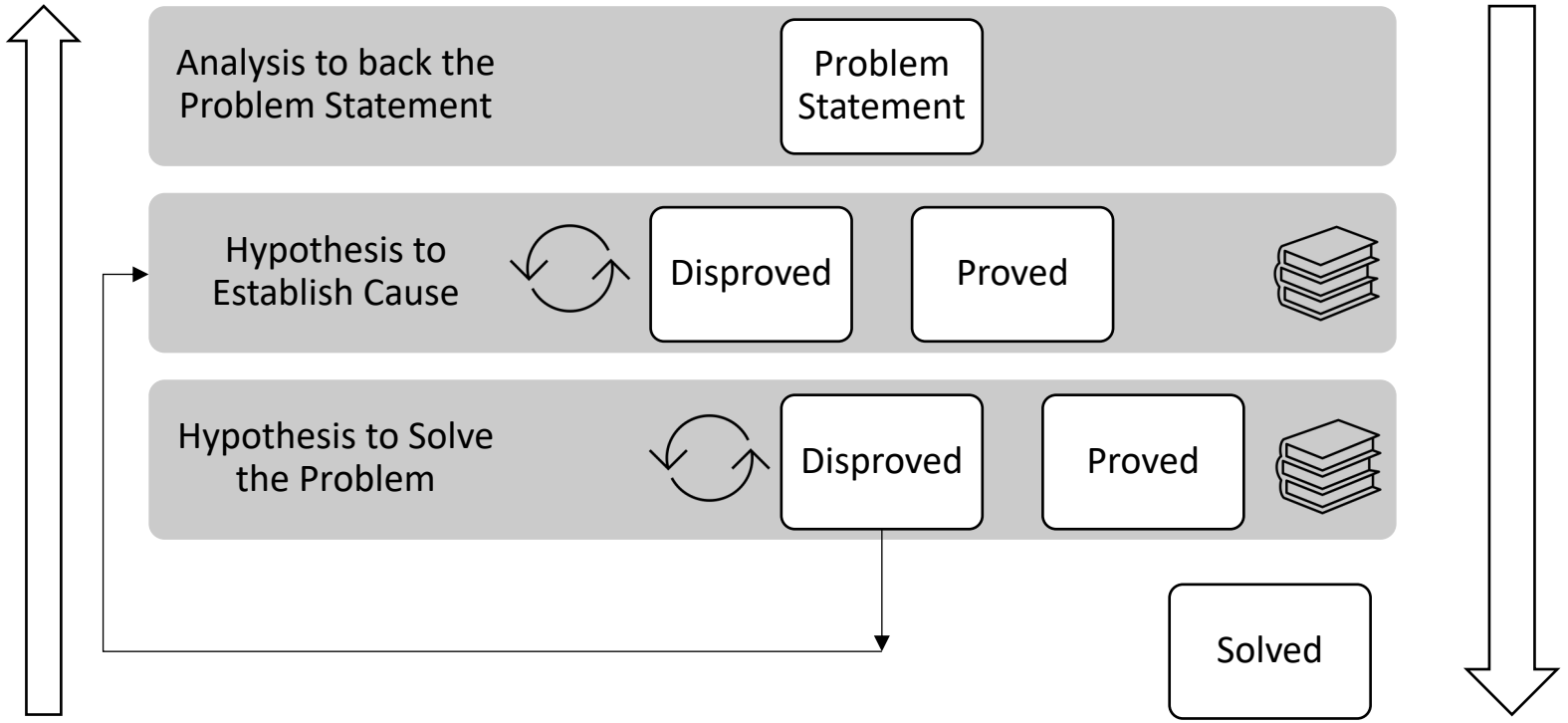
- **Disprove** – At 200 RPS we do not see messages pertaining to a single ticket in more than one Partition
- **Prove** – After adding “ticketId” we should not see lost updates in TicketsDB even beyond 200 RPS

Great these experiments should be successful, lets celebrate

- Not so soon, what if Verifiability Experiment fails. Even after adding partition key, we see lost updates.
- You guessed it right, back to “Establishing the Problem”. Maybe the lost updates was a combination of Kafka partition key issue and Ticket Service. We need a hypothesis to see what is wrong with Ticket Service.

Hypothesis Driven Solutions

Traceability
Why we reached this Solution



Evolution
Arrive at the solution instead of guessing it

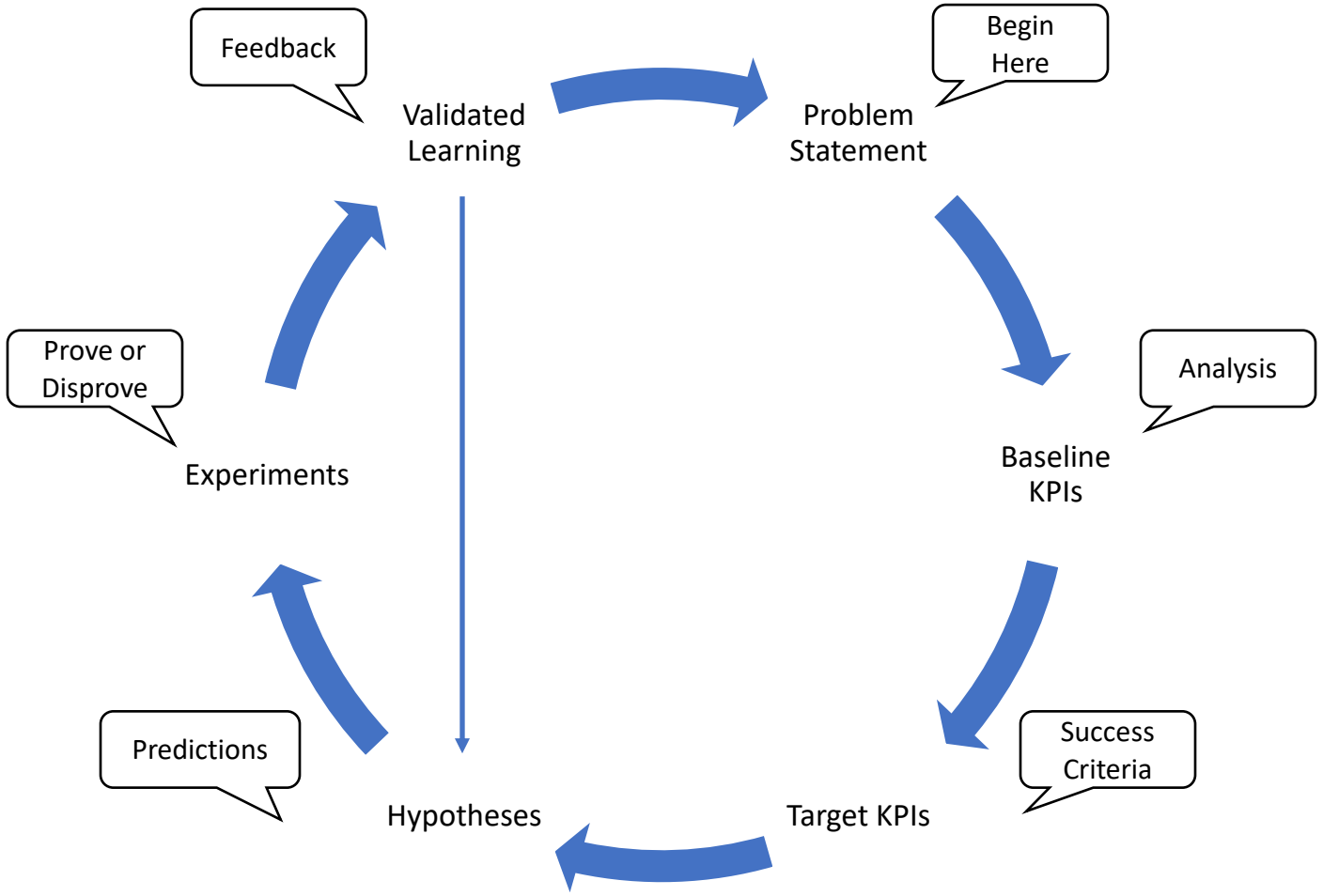
Templatizing the approach

Continuous Evolution Templates

Templatized Version

Problem Statement	Baseline KPI	Target KPI	Hypotheses	Experiments	Validated Learning	Action Item
			Establish Cause			
			Solution			
			Establish Cause			

Thinking Scientifically



Stakeholders

Keeping them involved

Key Stakeholders

- Who are they?
  - Hypotheses can be suggested by any team member
  - However Key Stakeholders are the SMEs for a given Problem Statement
- Roles and Responsibilities
  - Authoring Problem Statement and providing the supporting data
  - Setting Target KPIs
  - Accepting Experiment Setup
  - Accepting Results to sign-off on Validity and Actions

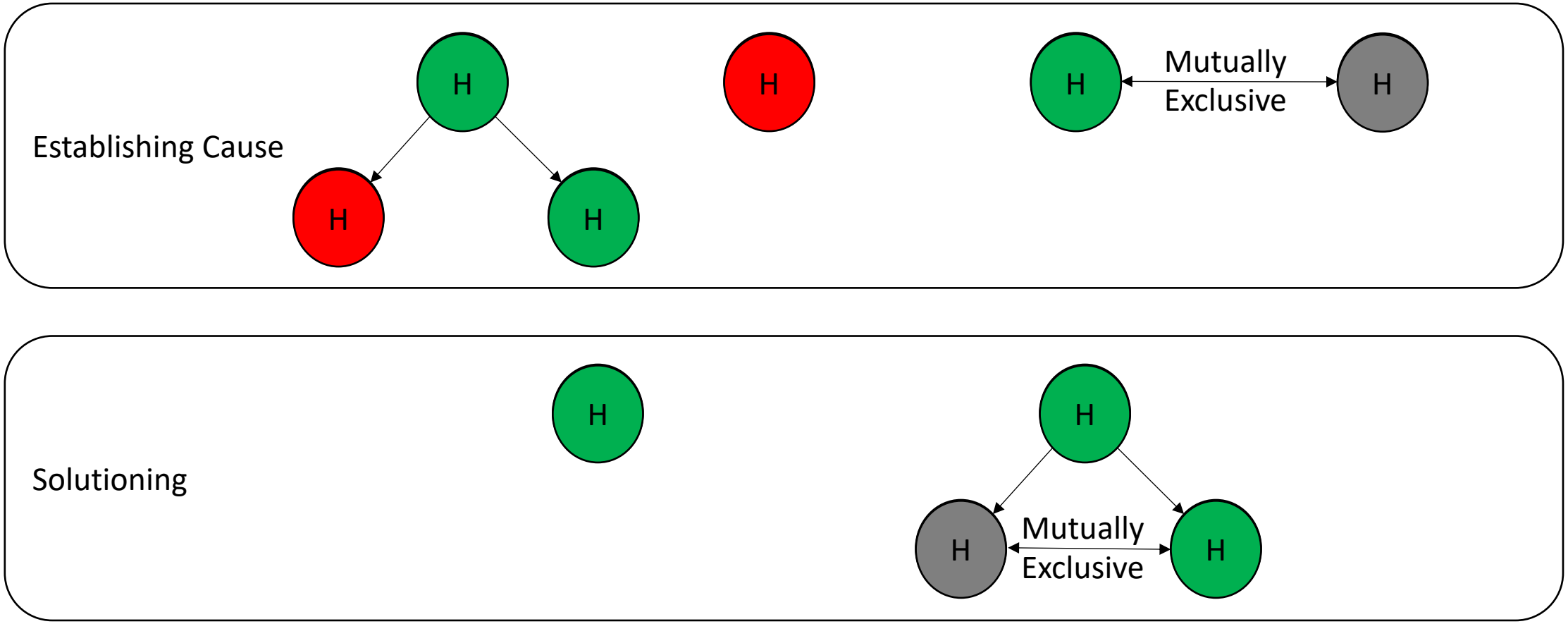
Lazy Evaluation

Evaluate Hypotheses only on Need Basis

Hypotheses Relationships

- **Mutually Exclusive** – In a group if Hypotheses Hs, if we prove any one of them to be valid, others are immediately invalid
- **Related** - Hypothesis 1 must be valid for Hypothesis 2 to be plausible
- **Unrelated** –
  - In a group if Hypotheses Hs, validity of one Hypothesis does not influence the others.
  - It is possible that all of them are valid

Hypotheses Patterns



Prioritization

- Heads of related Hypotheses
- Members of mutually exclusive groups
- Unrelated
- Evaluate Solution Hypotheses only after establishing related Cause

Customization

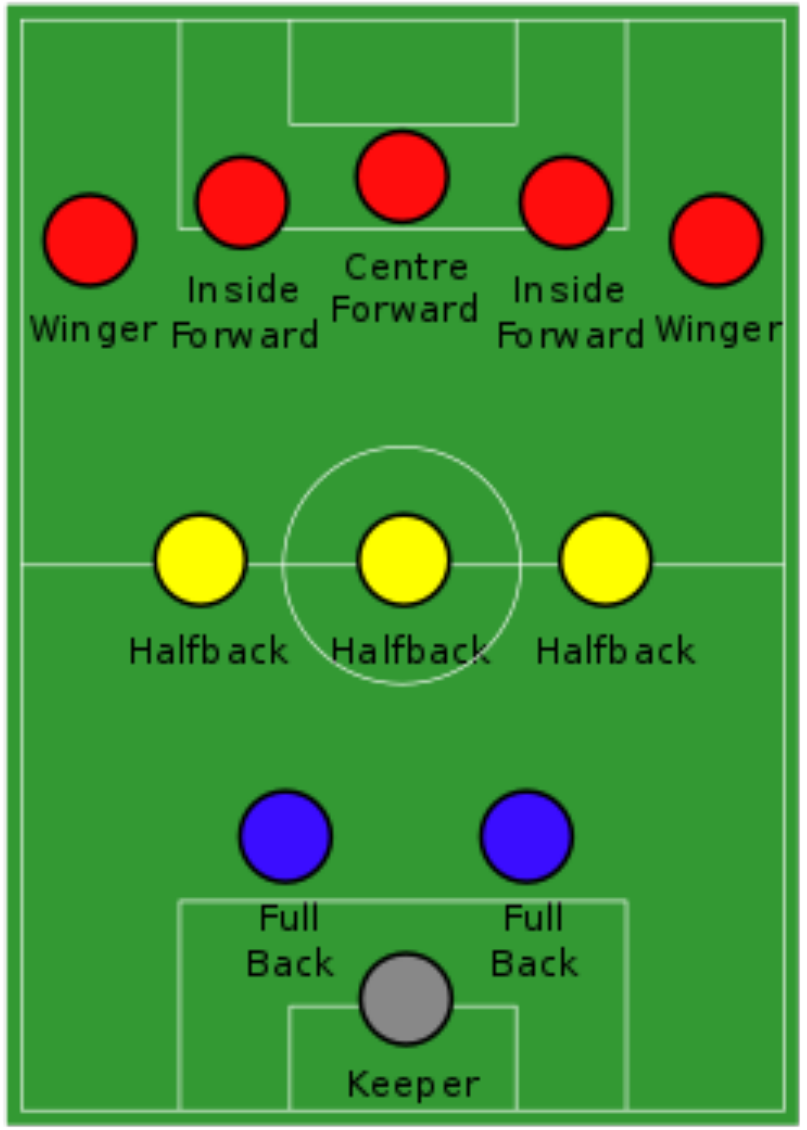
Tweaking Continuous Evolution Templates to your Context

Bending the Rules

- The idea of ”Continuous Evolution Template” is to facilitate structured thought process
- Once we have internalized the spirit, it is okay to skip steps
- Spreadsheet is not the only way to do this. Mind Maps are equally effective.

When, Where and How does it fit in?

Incorporating Continuous Evolution Templates into your existing process



Where, When and How

- **Backlog Grooming** – User Story Grooming, Tech Task Analysis sessions are great opportunities to author Hypotheses and fill up your CET. Based on these you can add stories, tasks, spikes into your backlog.
- **Demo** – Leverage these meetings to update and / or accept the experiment outcomes and decide on action items
- **Issue Triage** – Continuous Evolution Templates help analyze High Severity and Priority issues.
- **Keep it quick** – The objective is to generate hypotheses quickly and narrow viable paths without going down Rabbit holes. Practice timeboxing a single problem statement to about 15 minutes.

More Examples / Exercises

Live walkthrough to demonstrate Product Management,

Database Tuning, Refactoring, etc.

Recap

- “Validated Learning” to reach “Solutions” instead of “Guessing”
- Removing the “Pressure to get it right” by improving articulation about our work
- Governance / Traceability
- Visualize progress in long running research

Thanks!

@harikrishnan83