The Network Automation Map: Navigate Your Journey with the NAF Automation Framework

Wim Henderickx, Nokia
Damien Garros, OpsMill

About Us

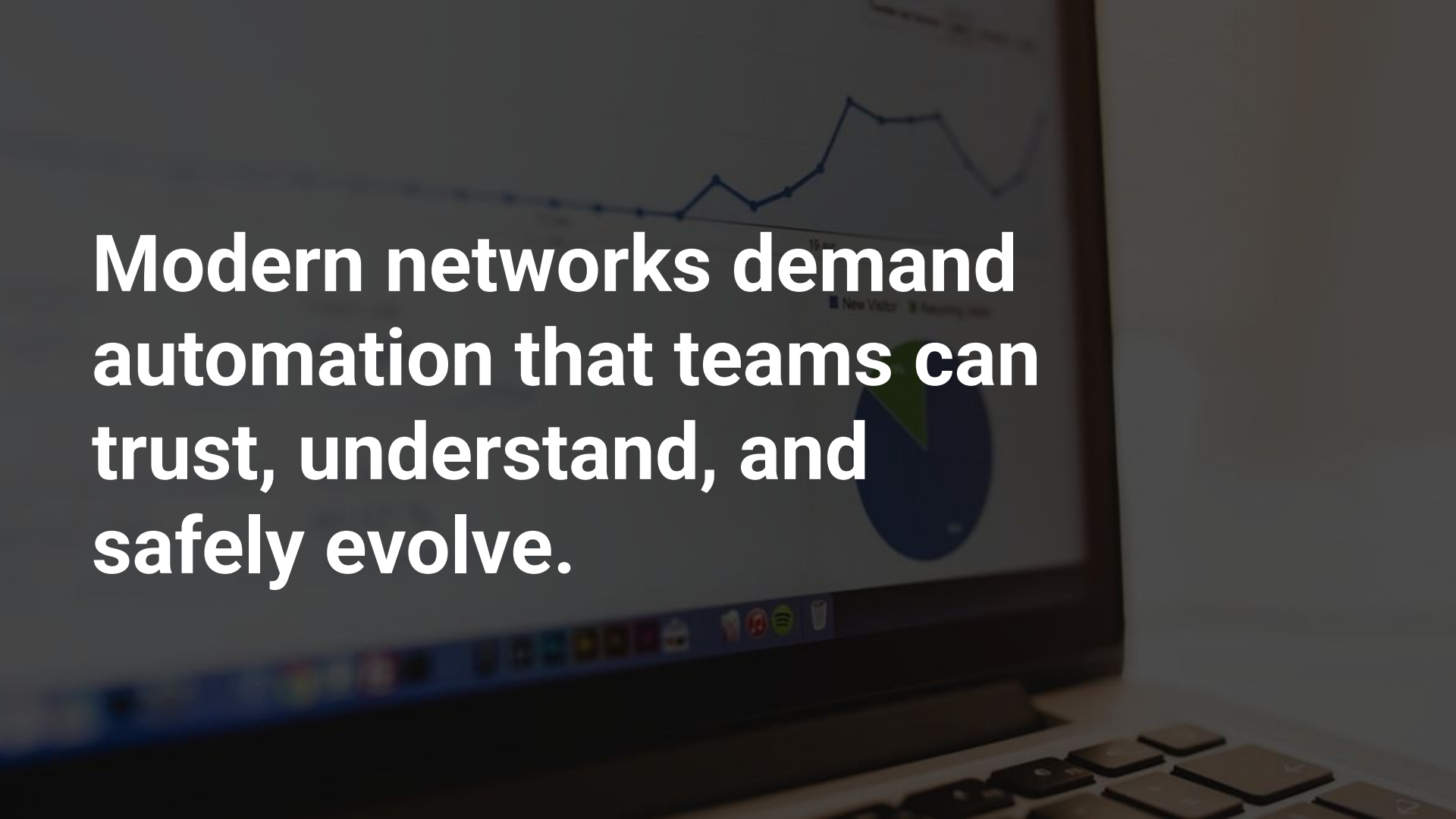
Wim Henderickx

CTO IP Division
Nokia

Damien Garros

Co-founder and CEO
OpsMill

Introduction


Modern networks demand automation that teams can trust, understand, and safely evolve.

Yet, it’s still overly complicated for most organizations

Building it is hard, maintaining it is almost impossible

The pattern we keep seeing

Rebuild from scratch

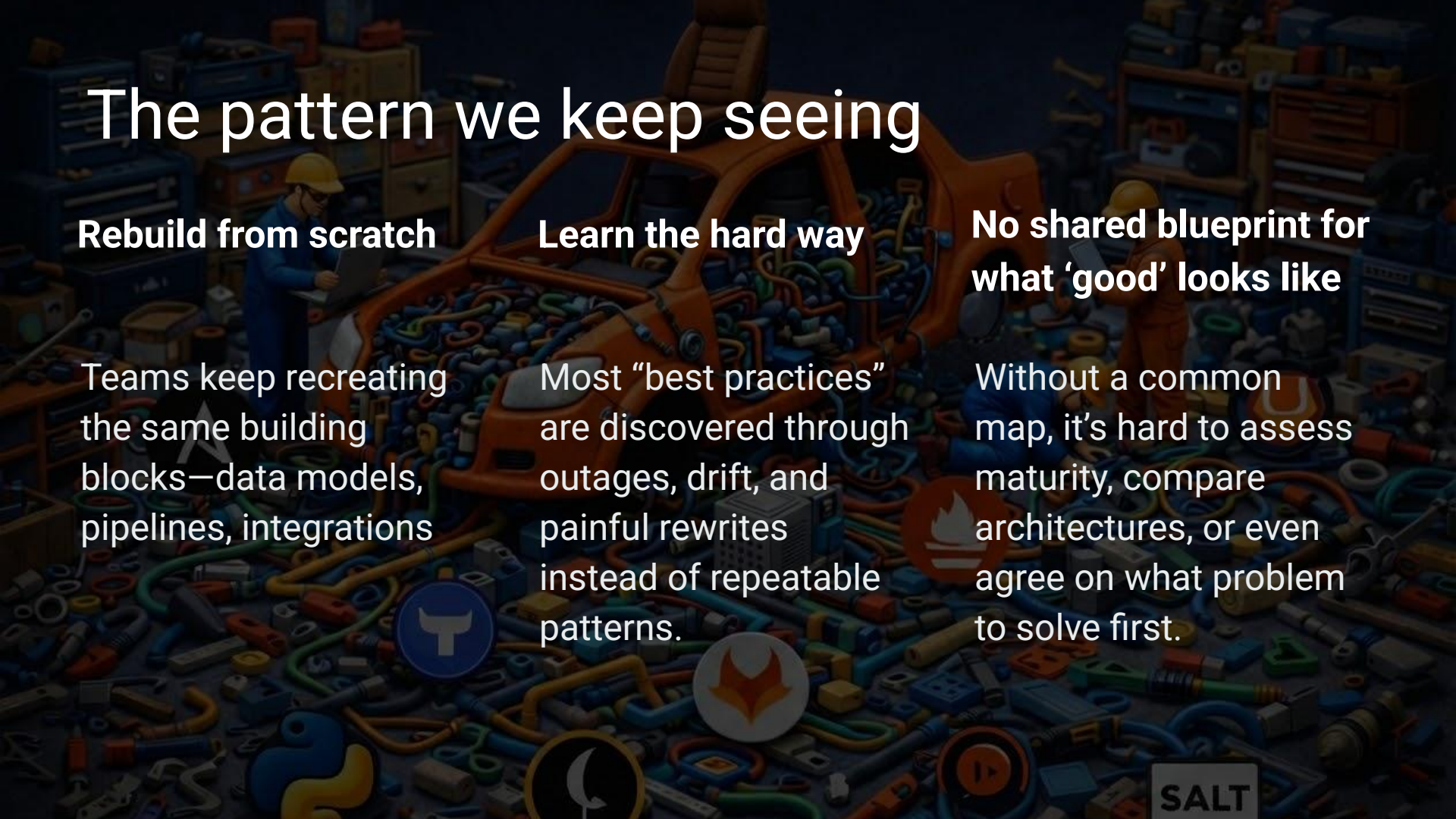
Teams keep recreating the same building blocks—data models, pipelines, integrations

Learn the hard way

Most “best practices” are discovered through outages, drift, and painful rewrites instead of repeatable patterns.

No shared blueprint for what ‘good’ looks like

Without a common map, it’s hard to assess maturity, compare architectures, or even agree on what problem to solve first.



Multiple people from the NAF community got together to find a solution

It quickly turned into a working group with weekly meetings

It was hard at first
Everyone was biased by their own tools and habits
Ironically, we were missing a shared lexicon

The solution was to stop focusing on tools and focus on functional building blocks

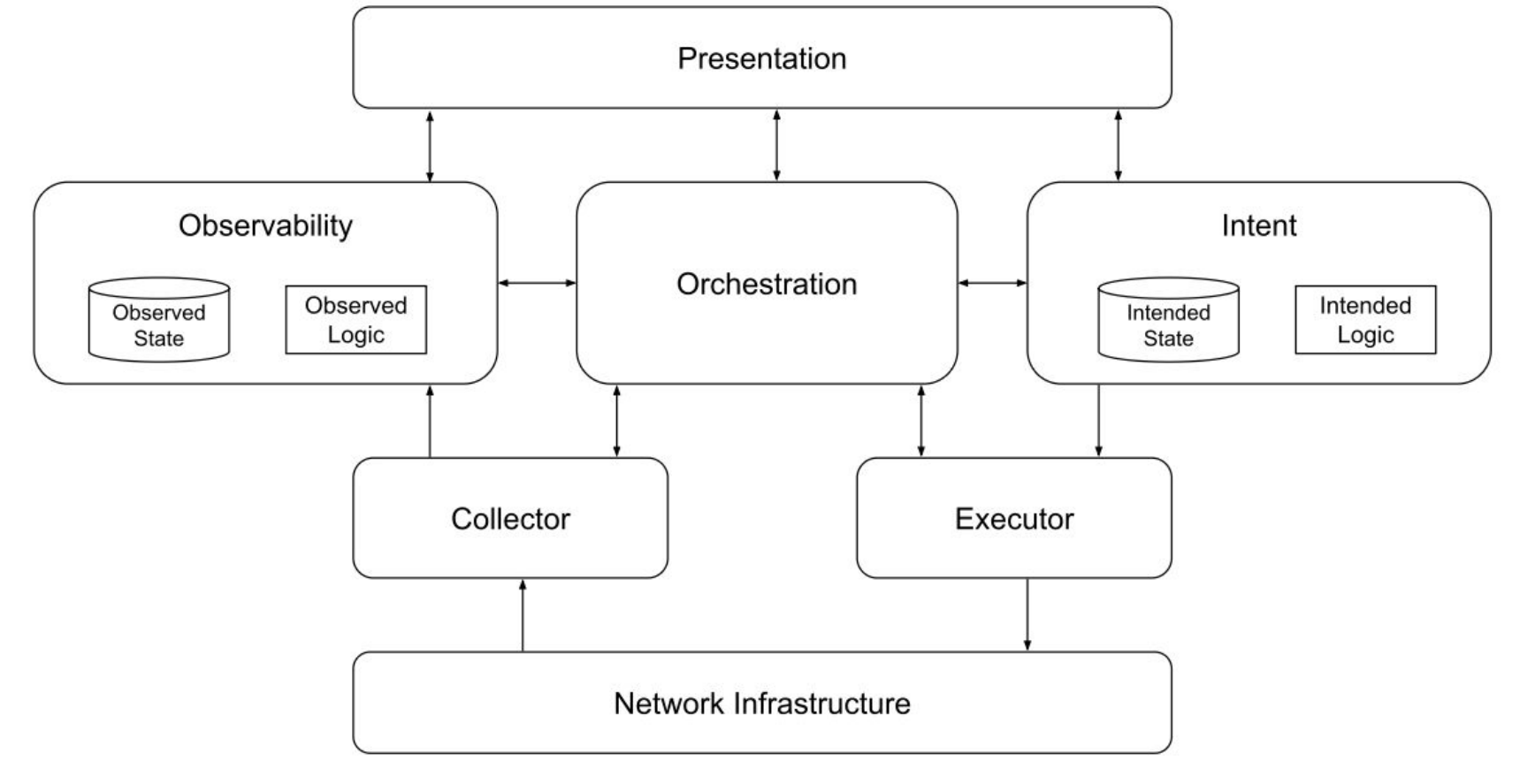
Quickly, everyone started talking the same language

NAF Automation Framework

Simple building blocks that underpin any network automation framework

Not a reference architecture — it’s a shared vocabulary + responsibilities.

NAF Automation Framework



NAF Automation Framework Fundamentals

Functional Building blocks

The NAF automation framework is meant to be tool agnostic to avoid comparing apple and orange.

Each functional building block can be map to one or more tool and each tool can map to multiple functional blocks

Living Document Not a protocol

The NAF automation framework is meant to evolve over time.

It’s not a strict implementation that imposes a specific way of doing things and in many cases, not all components are required to get started

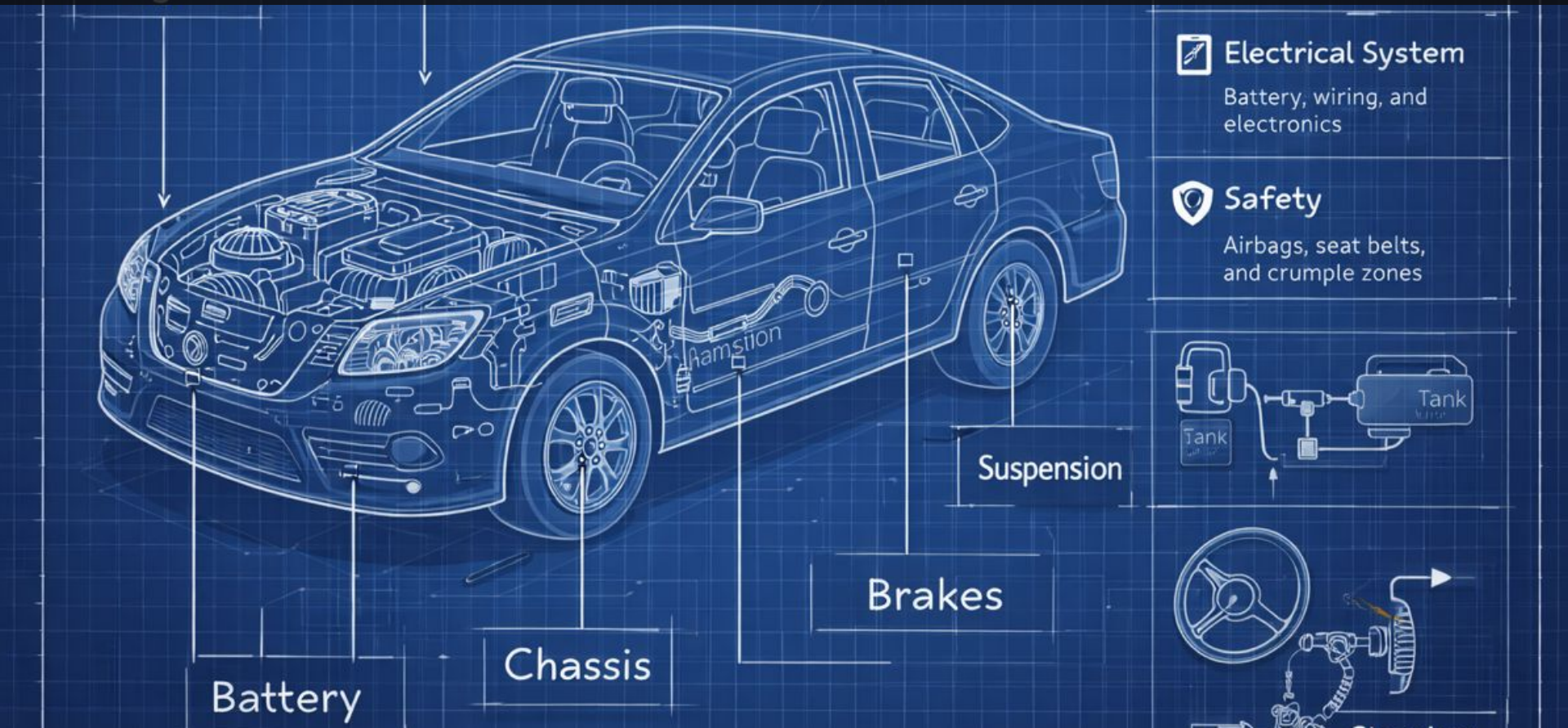
Both for the beginners and the experts

The NAF automation framework initiative is focusing initially on laying out the main building blocks to lay the foundation

In the future the goal is to go deeper into each block.

A map to help navigate the terminology and understand how things are related to each other

A functional blueprint to help everyone understand and build Network Automation Stack (system thinking)



Target Audience

Builders

Network Automation Engineer

Automation Architect

Software Engineer

Network Reliability Engineer

Users

Network Engineer

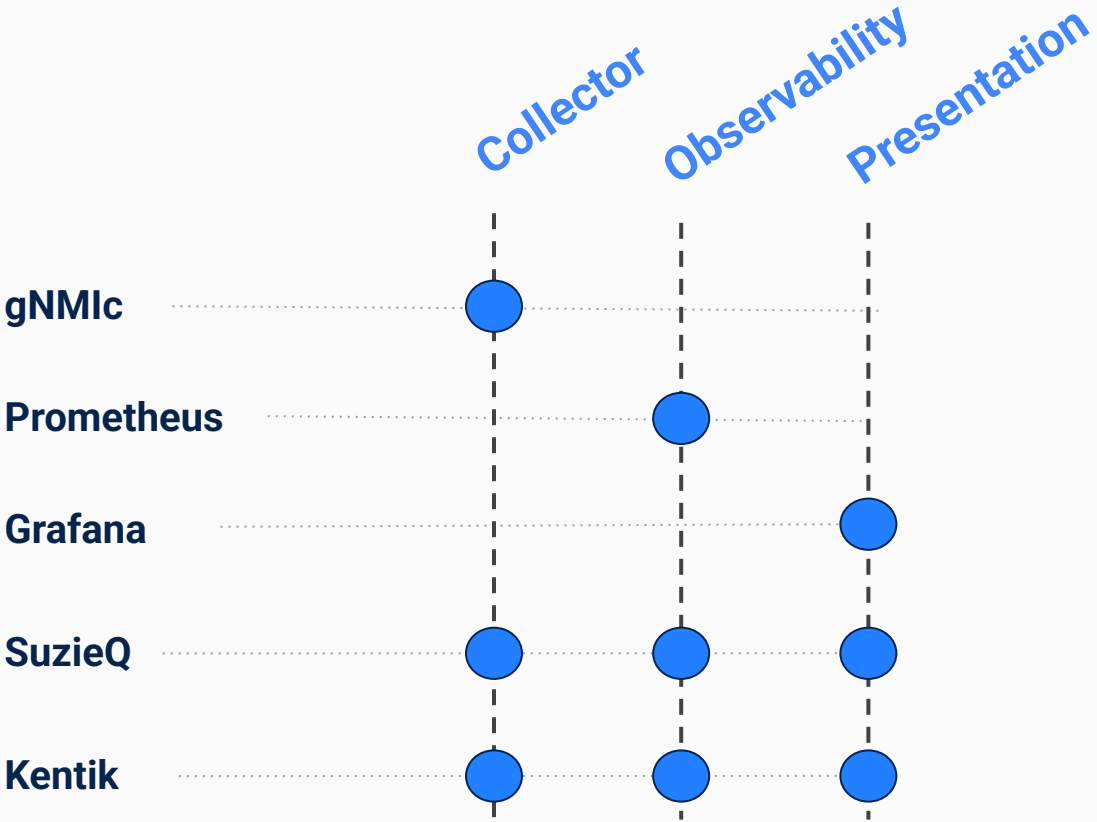
Network Ops

Network Architect

Engineering Manager


Wait, what about my tools ? Are they gone ?

Mapping between functional blocks and tools



Each tool / product / project
can map
Completely or Partially
to
One or Multiple blocks

Where do I start ?

Do I need to implement all blocks to get started ?

No, you should only implement what is required for your use case(s)

Is there a specific order to get started ?

No, it really depends on your use case(s).

Ryan Shaw

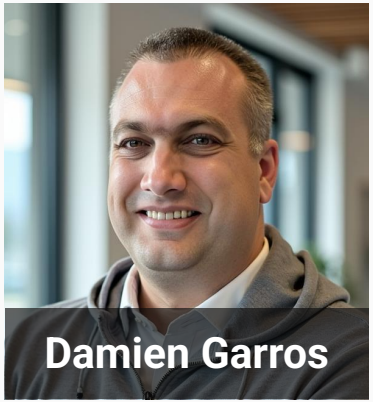
Dinesh Dutt

Christian Adell

Claudia de Luna

Srividya Iyer

Wim Henderickx



Damien Garros

The NAF Community

The Team behind the Working Group

Main functional Building blocks

What defines a functional building block

A **tool-agnostic function** in the automation system —
a **chunk of responsibilities** with clear goals, inputs/outputs, and capabilities

Purpose / Goals

what outcome it exists to deliver

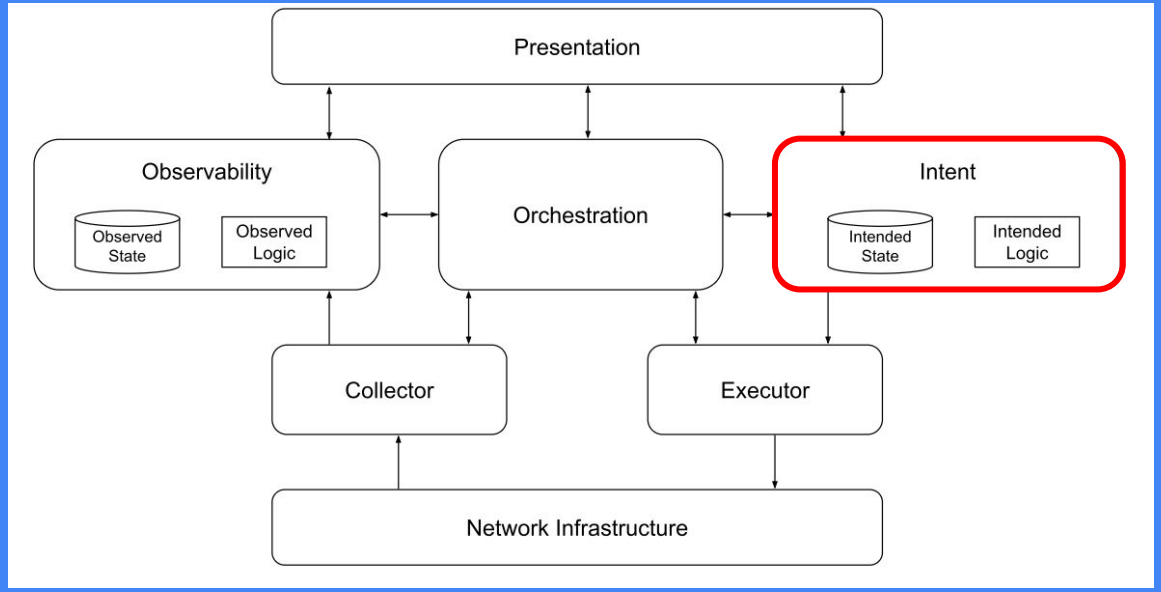
Capabilities

expressed as MUST / SHOULD / MAY (requirements, not products)

Boundaries

what it does and what it does not do (inputs → outputs)

Intent


Presentation
Observability
Observed State
Observed Logic
Orchestration
Intent
Intended State
Intended Logic
Collector
Executor
Network Infrastructure

Intent

Defines the logic to handle and the persistence layer to store the desired state of the network, including both configuration and operational expectations.

Source of Truth 

Source of Truth has been the root cause of many misunderstandings within our industry.

Intent 

Intent is meant to provide a more accurate definition

Intent - Introduction

Inventory	Cabling / Topology	DCIM
Circuit	IPAM	Roles/Statuses
Configuration Templates	Routing Information	Services

What do you need to rebuild it
if the network was completely lost?

Why do we need the Intent

Declarative Vs Imperative

```
configure terminal
interface GigabitEthernet0/1
switchport access vlan 10
exit
Exit
write memory
```

Imperative - HOW

- Manually describe the step-by-step recipe.
- If something goes wrong halfway, state may be inconsistent.

```
interface:
  name: GigabitEthernet0/1
  vlan: 10
```

Declarative - WHAT

- You describe the desired end state, not the steps to get there.
- Easier to make idempotent and retry safely.
- Vendor neutral

Differences between Cloud and Networking

	Cloud	Networking
Declarative		   
Imperative	Optional	Optional

Intent - Requirements

MUST be capable of representing, in a structured form, any network-related aspect.

MUST support create, read, update, and delete operations

MUST be exposed through a standardized, well-documented API

SHOULD provide a consistent and unified view of the desired state

SHOULD use a neutral representation that will be derived into vendor-specific configuration artifacts

SHOULD include metadata that supports effective data governance

SHOULD be transactional, and provide a versioned access to data.

MAY include all the logic related to intended state management

Version Control

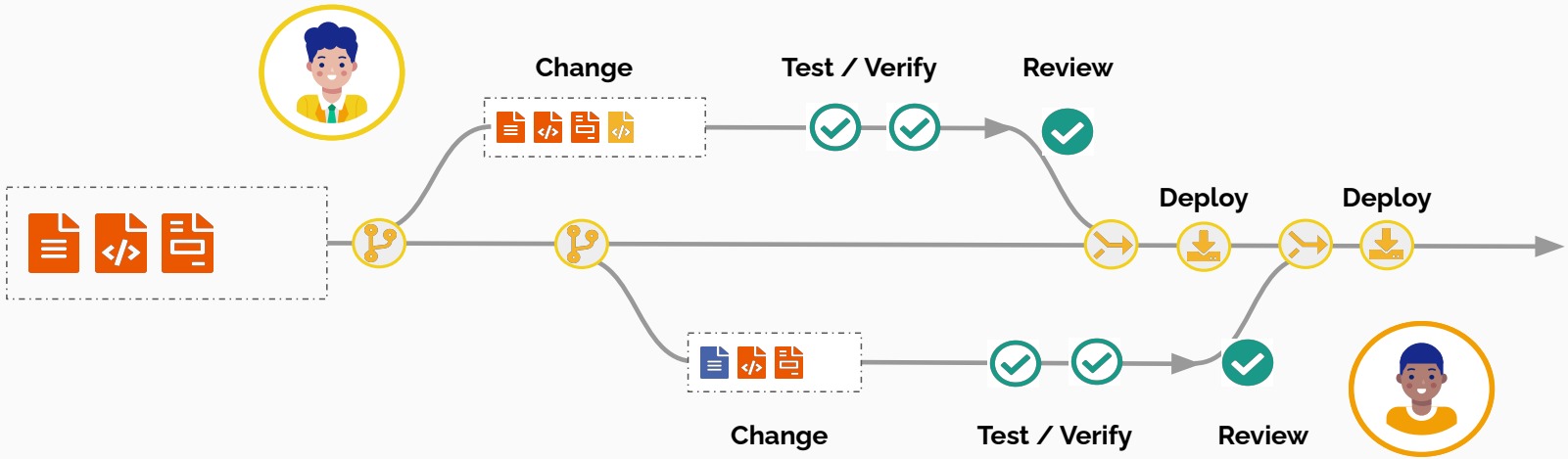
SHOULD be transactional, and provide a versioned access to data.

Version control allows changes to be:

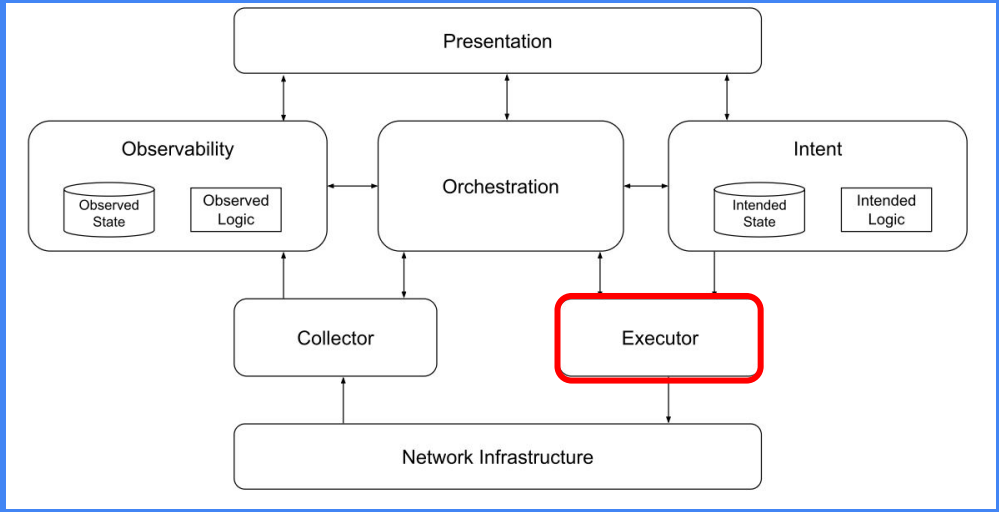
- Prepared in isolation
- Safely validated
- Reviewed

and only then integrated into the main automation environment.

Version Control Workflow



Executor


Presentation
Observability
Observed State
Observed Logic
Orchestration
Intent
Intended State
Intended Logic
Collector
Executor
Network Infrastructure

Executor

Encompasses the actual tasks applied to the network to drive changes (e.g., updating configuration) as guided by the intended state.

Executor - Requirements

MUST be capable of interacting with any of the supported network write interfaces

SHOULD provide a dry-run operation to check the expected result of the execution

SHOULD come from the intended state or be derived from it

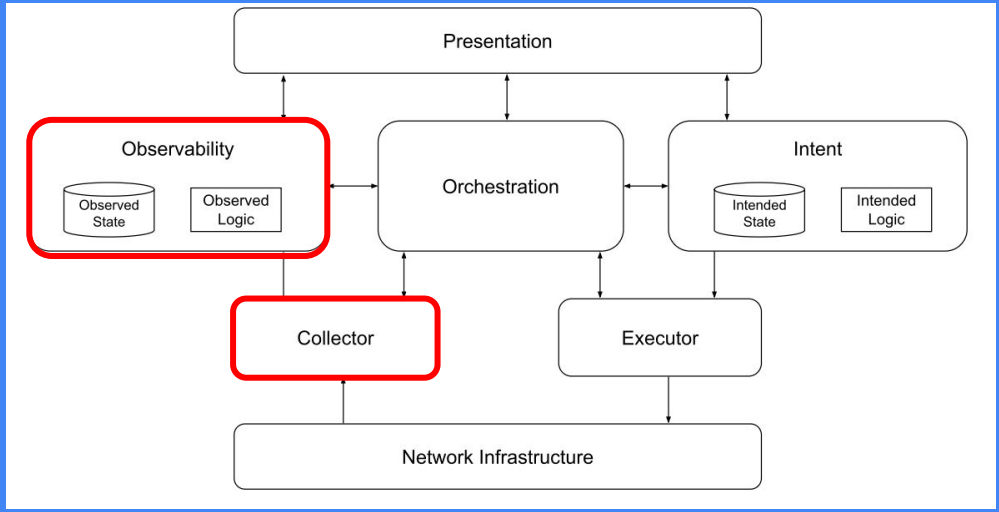
SHOULD support transactional execution of the changes

SHOULD support any network operation that alters the network state

MAY support both imperative and declarative approaches

Observability

Collector



Observability

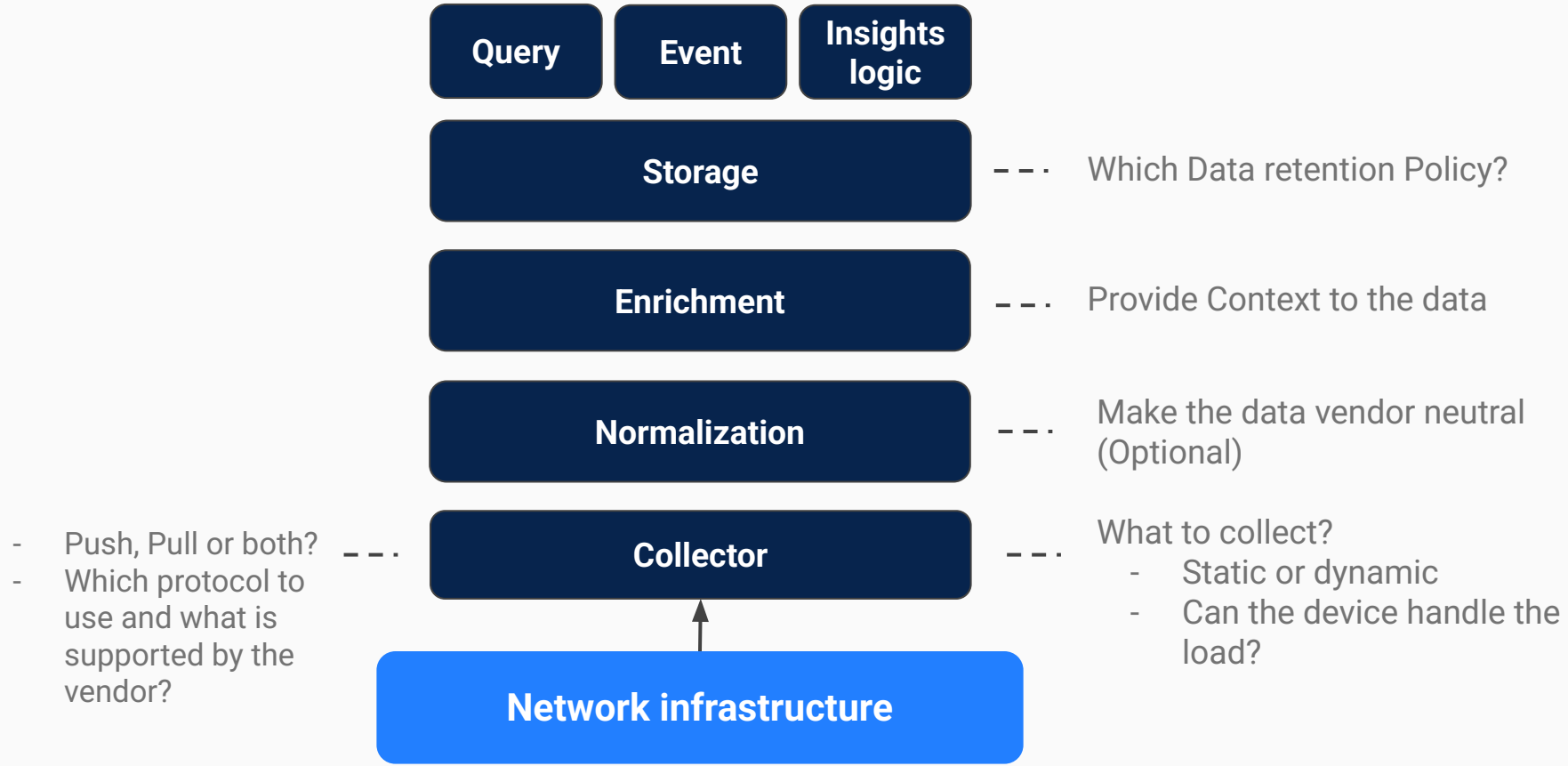
Stores the actual network state, and defines the logic to process the observed data.

Collector

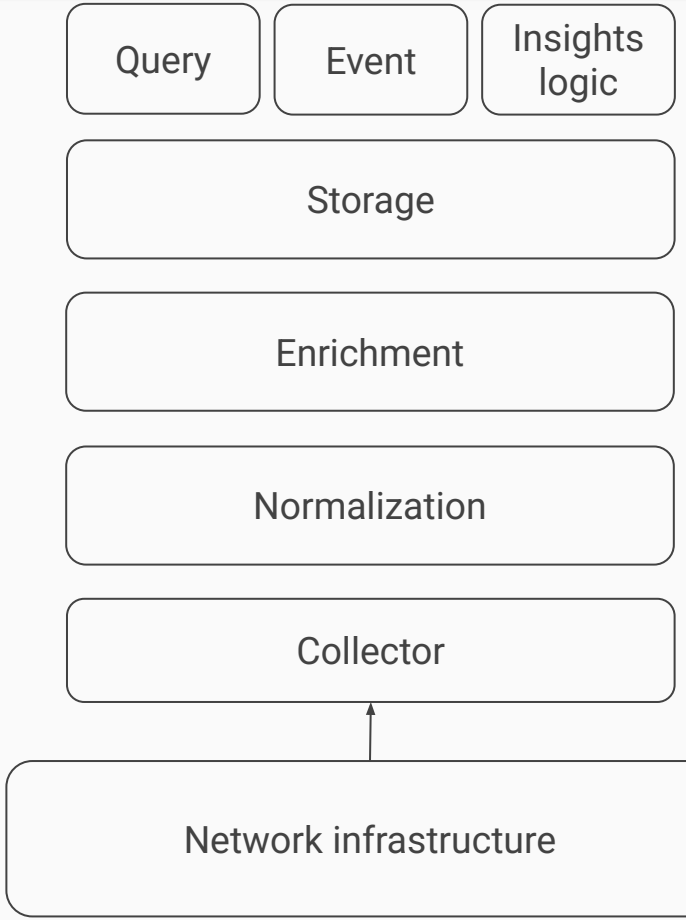
Focuses on retrieving (i.e., reading) the actual state of the network.

Observability Sources

Observability Stack: some questions to ask



Observability Stack: some questions to ask



Which Data retention Policy?

Provide Context to the data

Make the data vendor neutral (Optional)

What to collect?
- Static or dynamic
- Can the device handle the load?

- Push, Pull or both?
- Which protocol to use and what is supported by the vendor?

Observability - Requirements

MUST support historical data persistence

MUST offer programmatic access to this data

SHOULD offer a capable query language to extract the data.

SHOULD expose relevant insights into the current network state

SHOULD automatically generate events when discrepancies are detected

data SHOULD be normalized into vendor agnostic data model

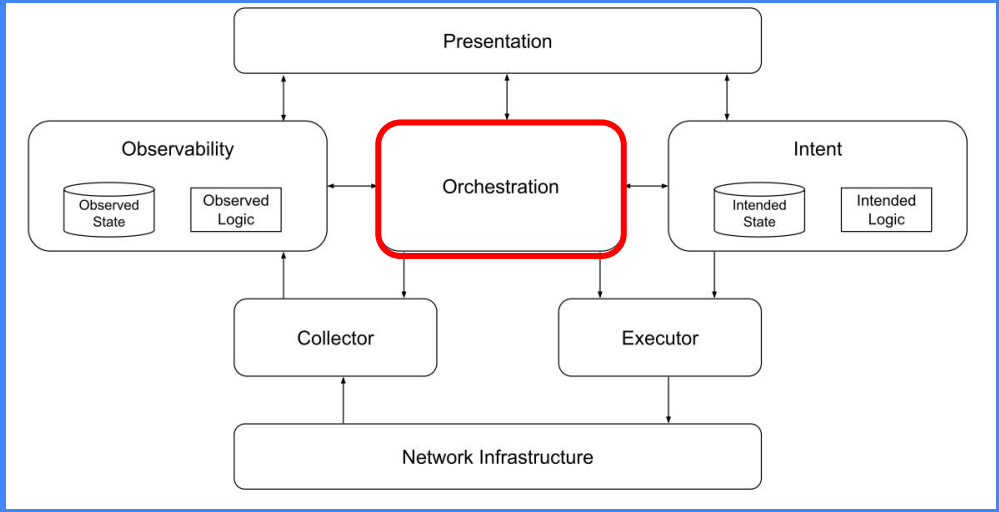
events MAY be processed by humans or connected to the Orchestration

MAY be enriched with contextual information from the intended state

Collector - Requirements

MUST includes capabilities for retrieving live data from the network using read interfaces (push, pull)

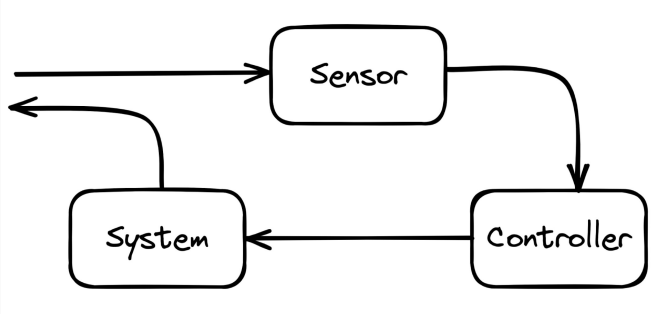
Orchestrator

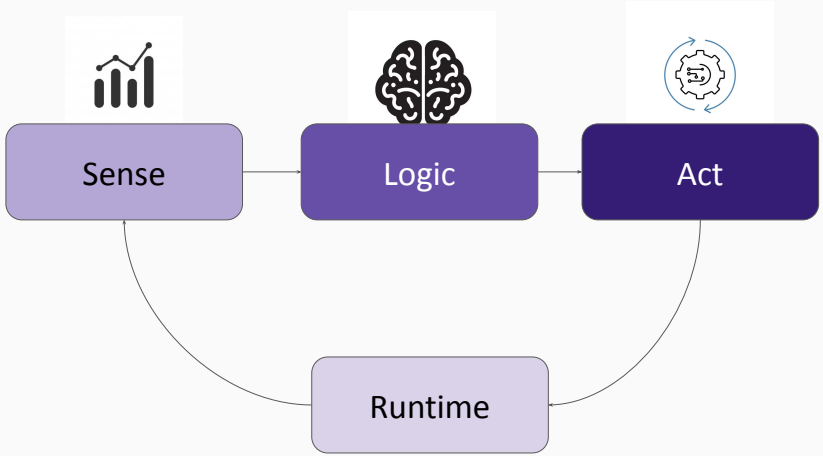


Orchestrator

Coordinates activities across various building blocks

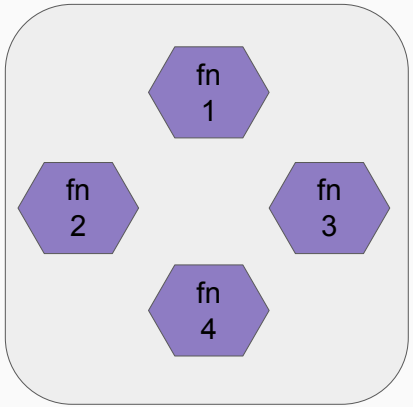
Event Driven Control Loop: Sense - Logic (Reason) - Act





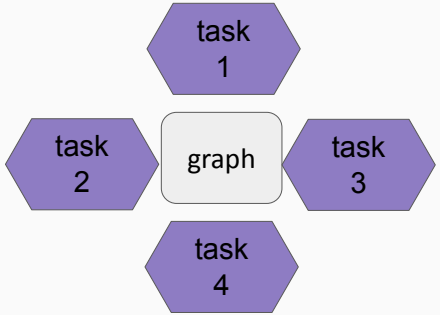
Orchestration implementation approaches

Monolith



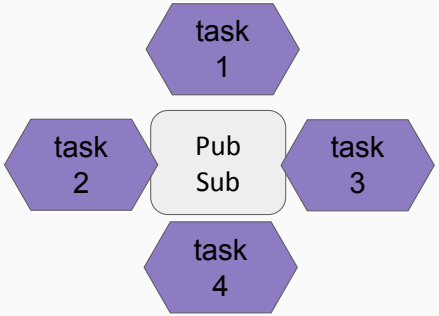
- E.g. a python program

Workflow



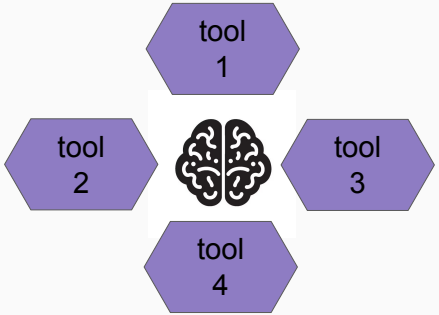
- Ansible (predefined workflow)

Choreography



- Event driven and distributed

Agentic



- Agentic

Orchestration: Trust

TRUSTED

- controller style

NOT TRUSTED

- change log/approval style
- Dry-run
- Diff (before/after)
- Rollback

Gaining Trust: Dry Run - Rollback - Diff

Diff:
- What changed?
- Review the changes before applying to the network

Try the workflow without applying it to the network

Rollback:
- Something bad happened, move back to a well known good state

Orchestrator - Requirements

MUST enable to coordinate processes across the various building blocks

Process execution SHOULD follow an event-driven approach

Execution logic SHOULD allow for reverse/compensating actions

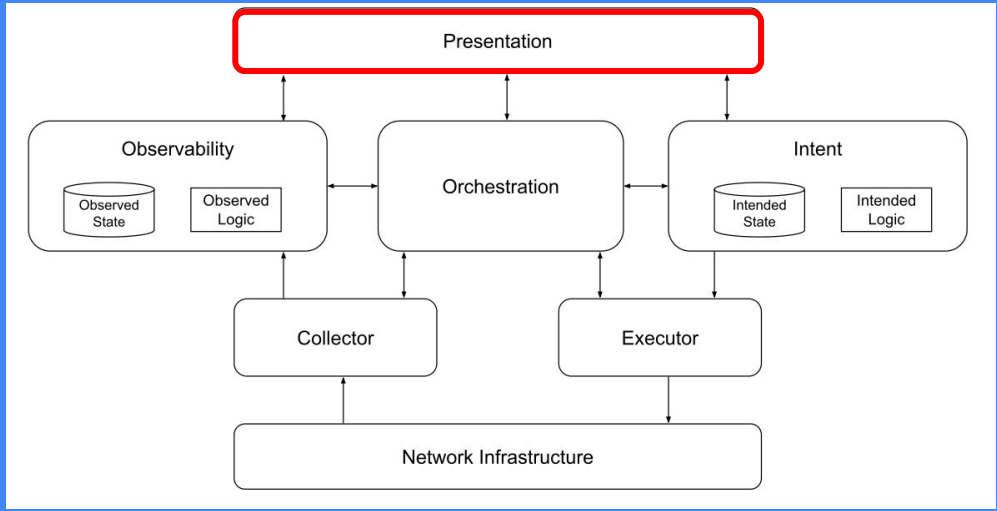
SHOULD provide the ability to schedule the execution of a workflow

SHOULD provide a dry-run operation to check the expected result of the workflow

SHOULD provide logging and traceability of past and current workflows

MAY include logic to correlate multiple events, infer relationships

Presentation



Presentation

Provides the interfaces through which users interact with the system, including dashboards, graphical user interfaces

Presentation

It’s important to separate the presentation layer to avoid building workflows that are specific to one tool / interface

It is designed to interface with any of the other building blocks as required, serving as the primary point of contact between humans and the automation system, but this does not imply the need for a single pane of glass.

Presentation - Requirements

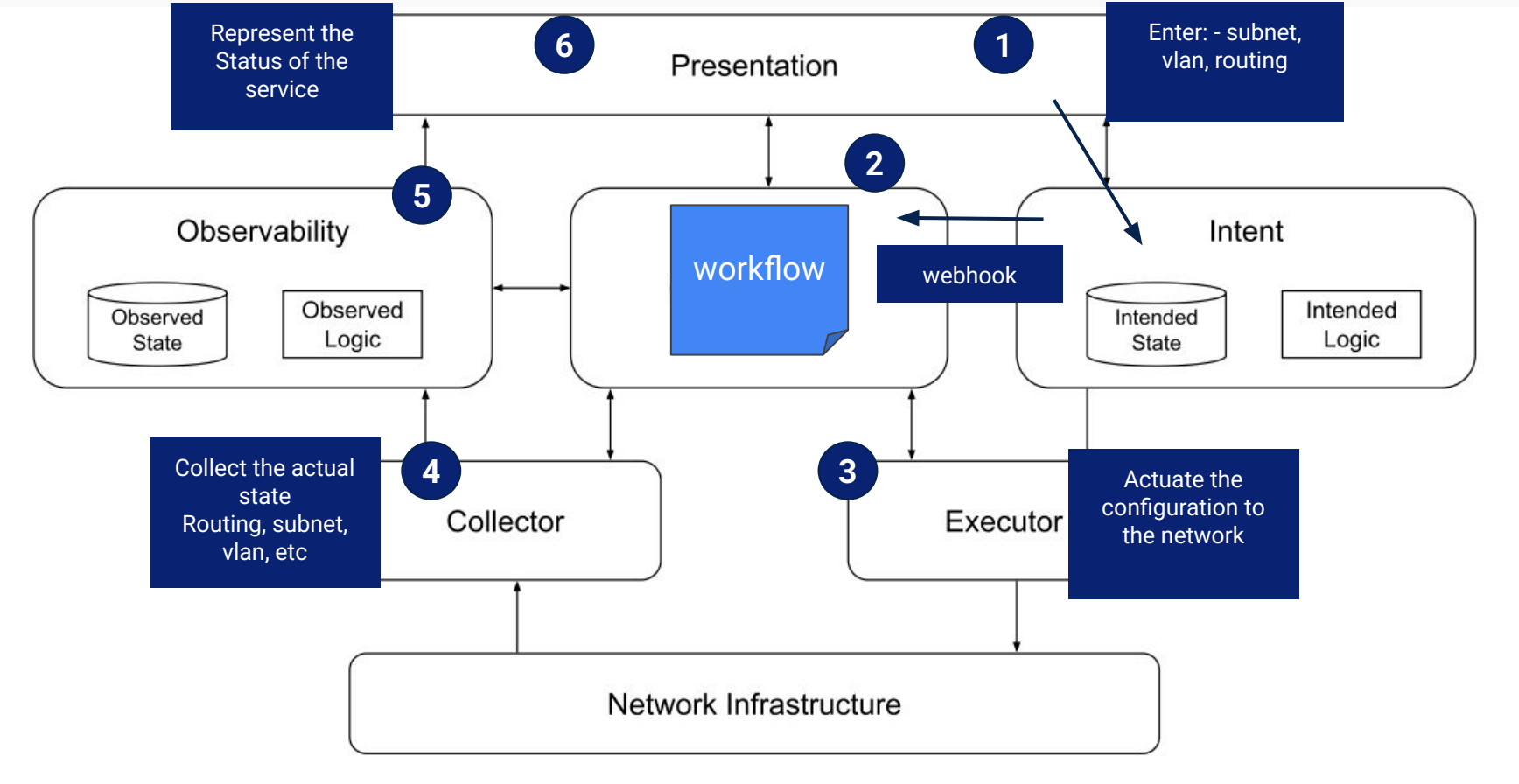
MUST provide robust authentication and authorization capabilities.

MAY take various forms depending on the needs of the end user

MAY support both read and write interactions

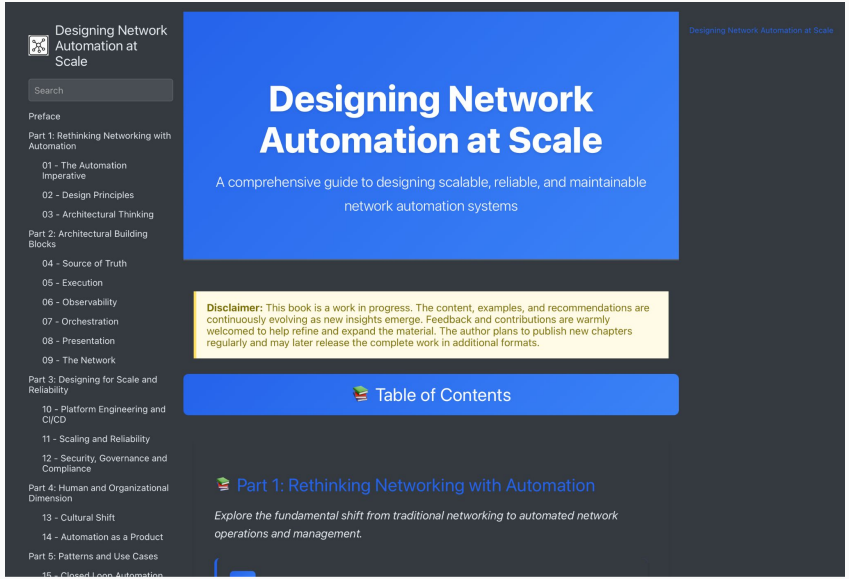
Practical Example

Use case: Rollout subnet 10.100.0.0/24 across 50 sites



Additional Resources

Additional Resources



https://designingnetworkautomation.com/
By Christian Adell

https://naf-naf-wizard.streamlit.app/
By Claudia de Luna

Summary

- Not a standard, it’s a guide
- You don’t need everything to get started
- Think Automation as a system
- Focus on functional first, tools later
- Each tool can map to multiple building blocks
- Don’t reinvent the wheel
- Map your existing workflow toward the framework
- Give us some honest feedback good and bad

How can you help

- Use case document Github:
  - https://github.com/Network-Automation-Forum/reference/blob/main/docs/Framework/Framework.md
- Tool mapping
- Expand details per building block
- NAF Slack channel:
  - https://networkautomationfrm.slack.com/archives/C082G6VGZHP


Questions ?

Agenda

Introduction	Damien	5 mins
High level overview	Damien	5 mins
Main Functional Building blocks		
- Intent	Damien	7 mins
- Executor	Damien	3 mins
- Observability + Collector	Wim	7 mins
- Orchestrator	Wim	7 mins
- Presentation	Wim	3 mins
Practical Example	Wim	5 mins
Additional Resources	Wim	1 min
Conclusion	Wim	2 mins
Questions		