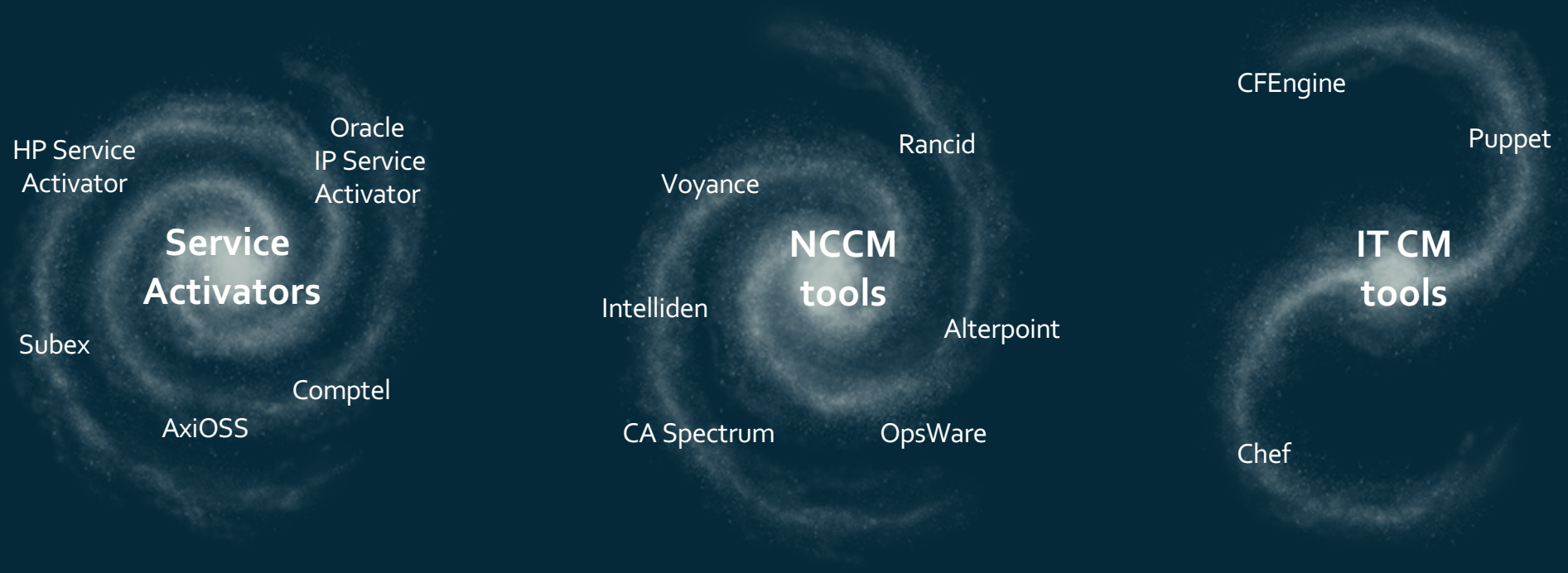
Developer Days
Network Services Orchestrator

NSO History and Design Principles

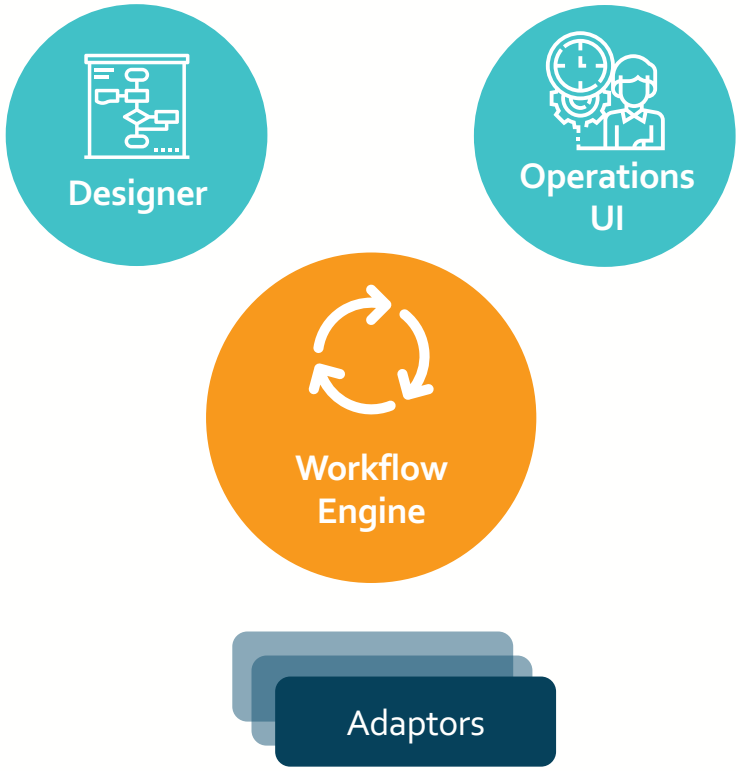
Stefan Wallin – NSO alumni product manager
Viktor Leijon – Technical Leader, Cisco

The Year is 2010 - Galaxies

The Year is 2010 - Stars



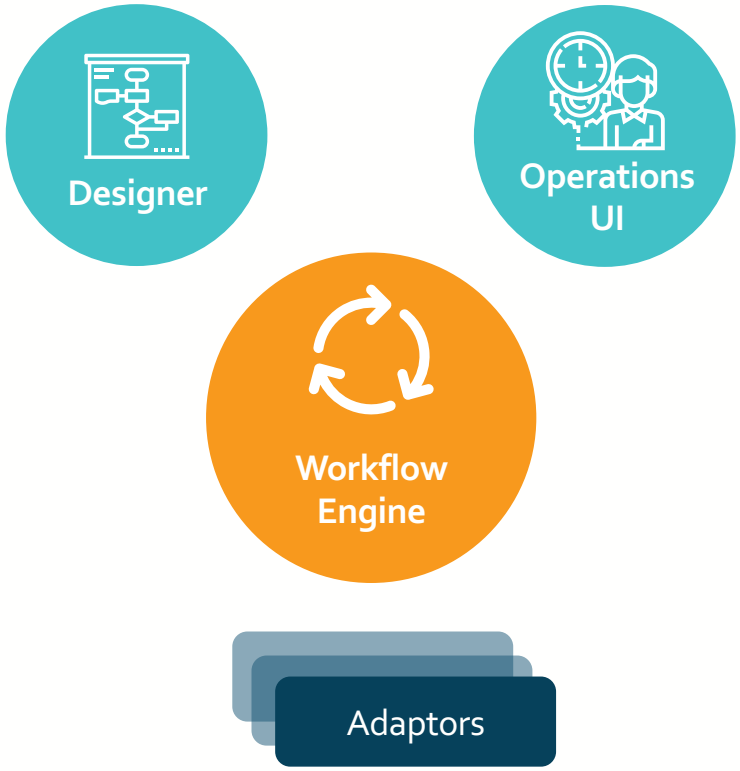
The Service Activators



- Execute workflow steps in sequence to **create** a service
  - And other workflows for other operations
- Mimic the manual steps performed
  - Support humans in the loop
- No focus on device configuration management
- Good visualization
  - Design
  - Operations - progress

The Service Activators - Challenges

For networking



- Lack of service life-cycle support
- Number of workflows
- Dependencies to inventory

→ Integration Cost

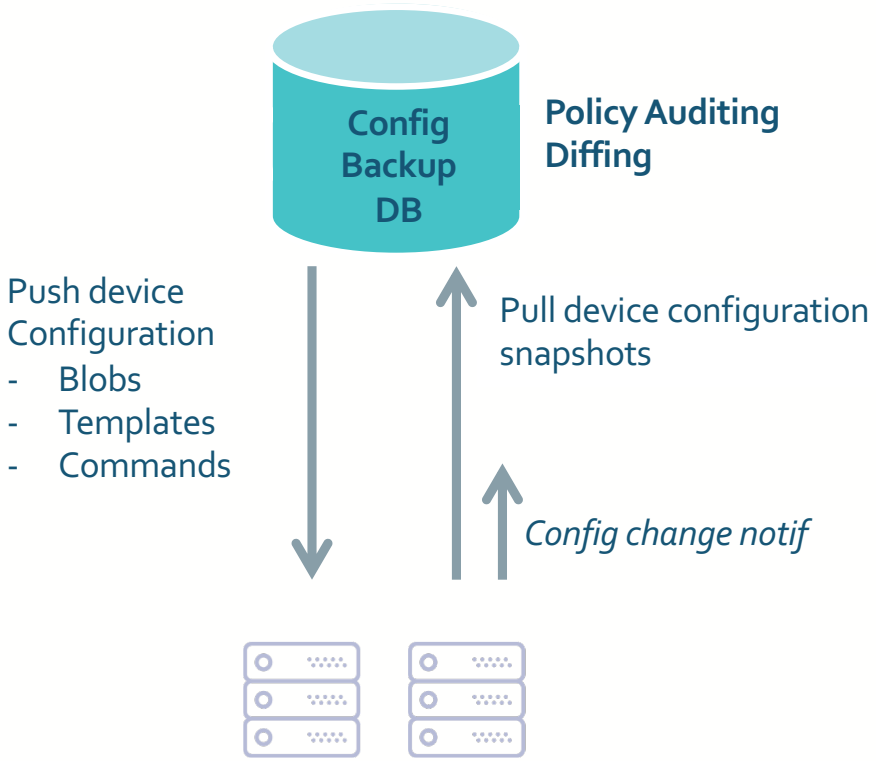
- Lack of transactionality
- Manual fallouts
- Broken network configuration

→ Operational Cost

- Looks easy initially

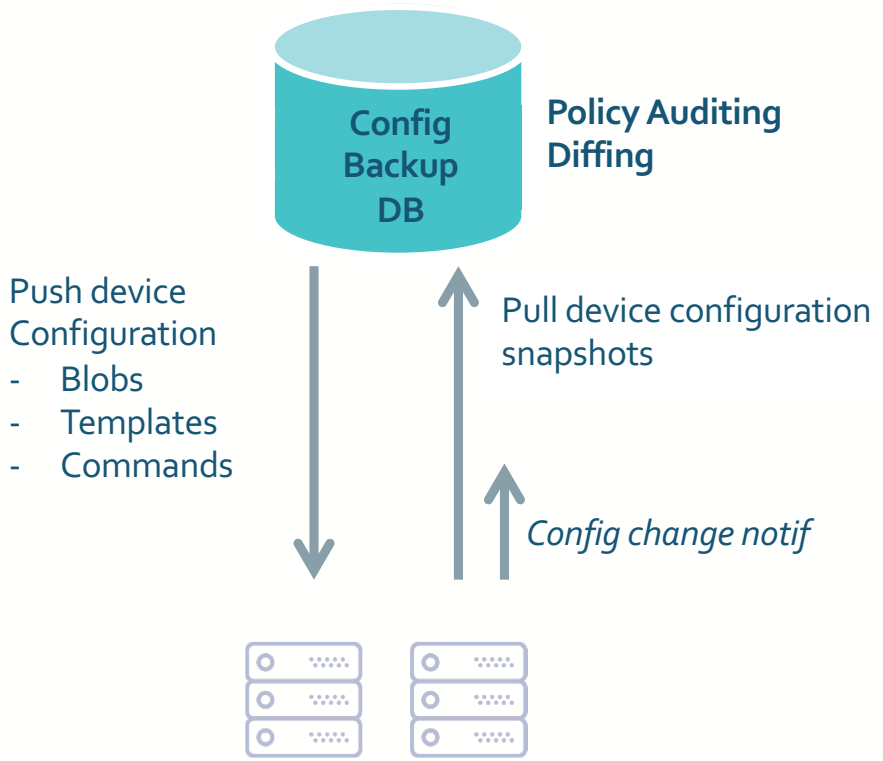
→ Total Cost

The NCCMs



- Focused on backup & restore of device configuration.
- Provision a new network device
- OS upgrades
- Compliance reporting
- Easy to understand for network engineers

The NCCMs - Challenges *For networking*

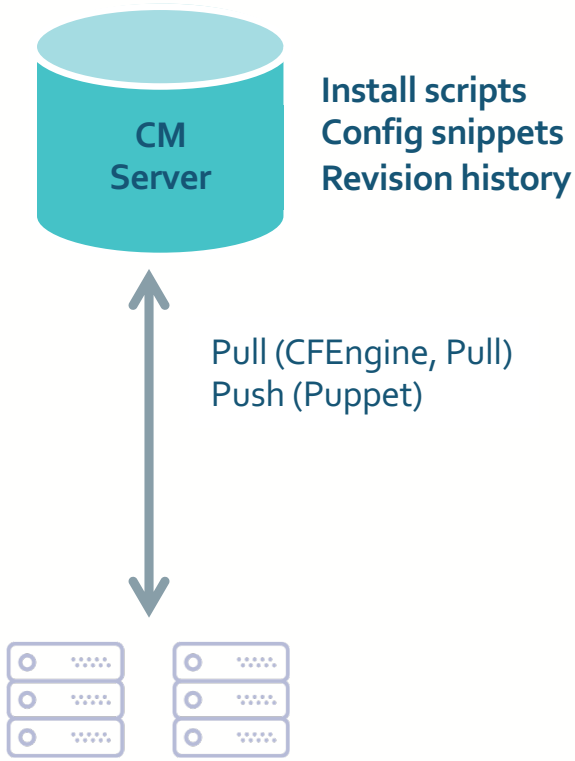


Limited automation support
No awareness of services
Native device configuration

Low
Automation
Value

Expensive configuration backup tool

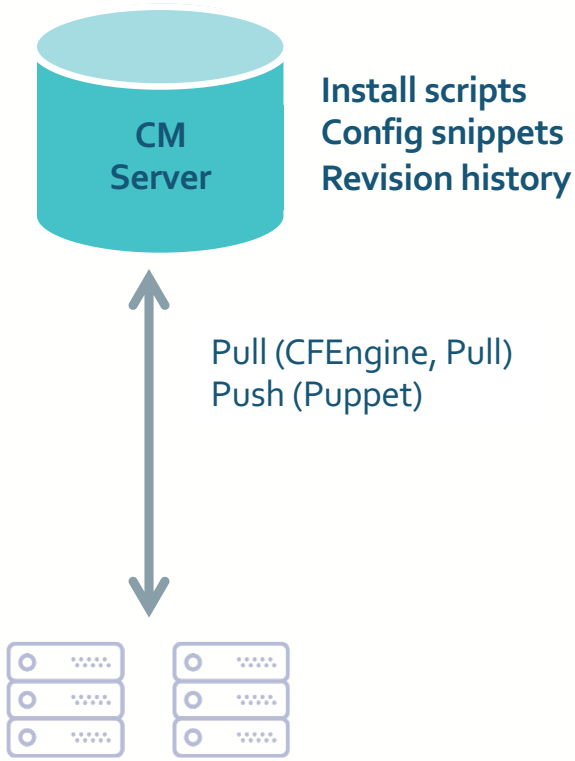
The IT CM Tools



- Managing long-lived desired state
- Mass deployement
- Migration
- Rollback versions
- Infrastructure as code
- Pull-push (not read-write)

The IT CM Tools - Challenges

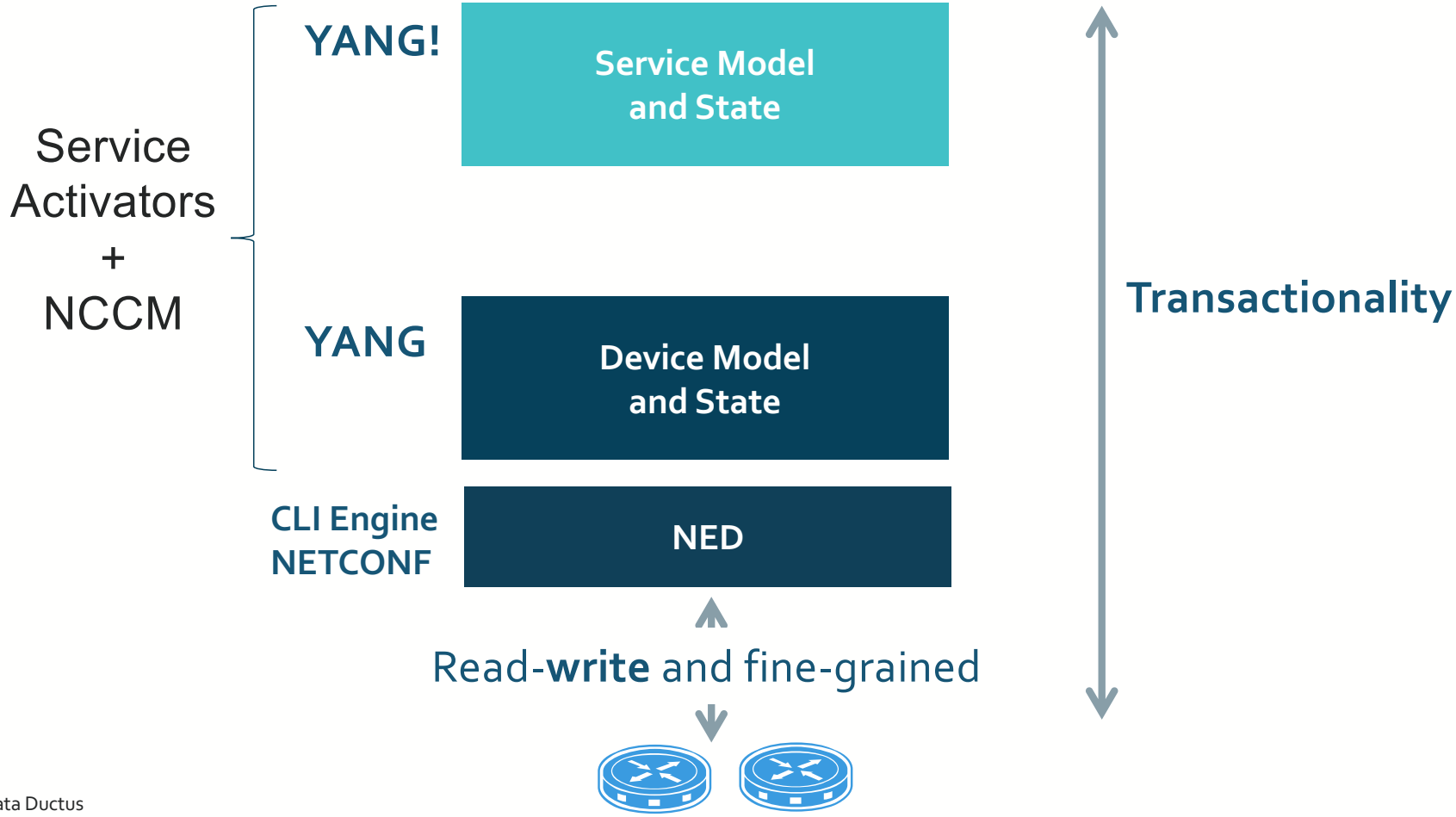
For networking



- No API towards state
- Lack of CRUD support
- Coarse-grained configuration
- Can lead to script mess
- Limited service representation

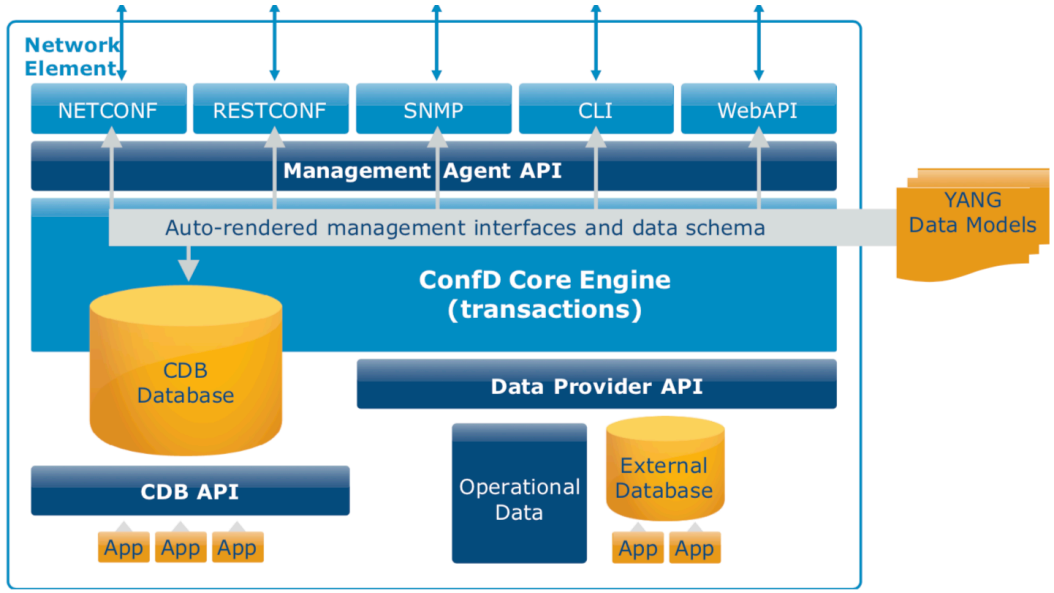
Limited value for
network service automation

The Birth of NSO – Invent and Learn from History



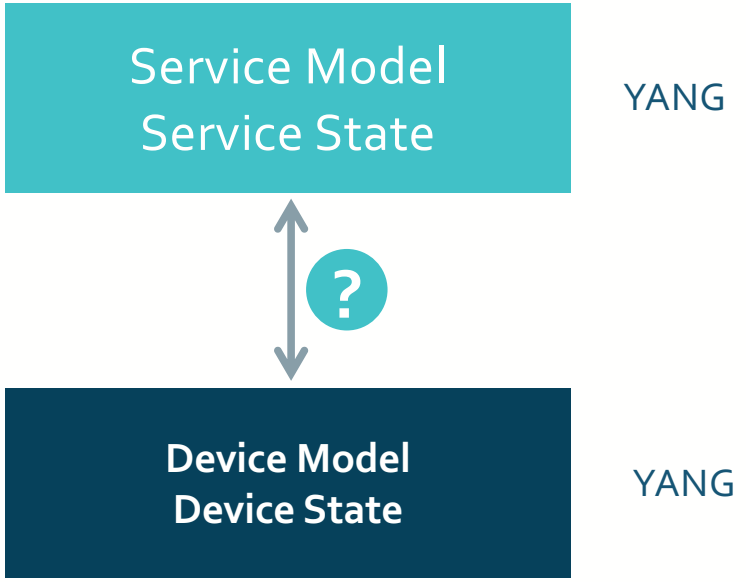
The NSO Core Principles from ConfD

Kudos Tail-f engineers!



- Data-Models
- Transactions
- APIs
- Automatic persistence and schema management
- Model transformation
- CLI engine
- Rendering

The 10 Million Dollar Question

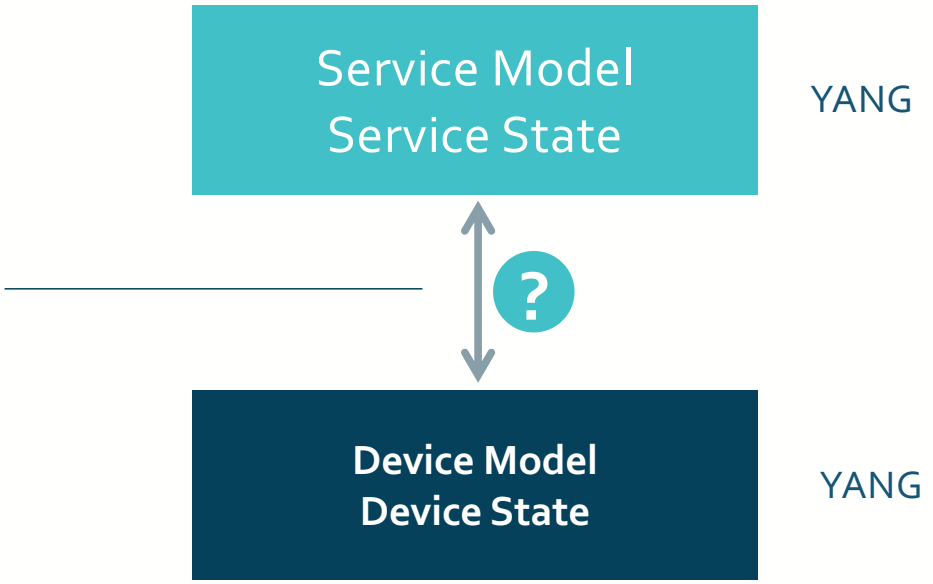


The 10 Million Dollar Question

"**Getting rid of workflows kills complexity … and believe me,**

We've got enough experience with classic order management and activation tools"

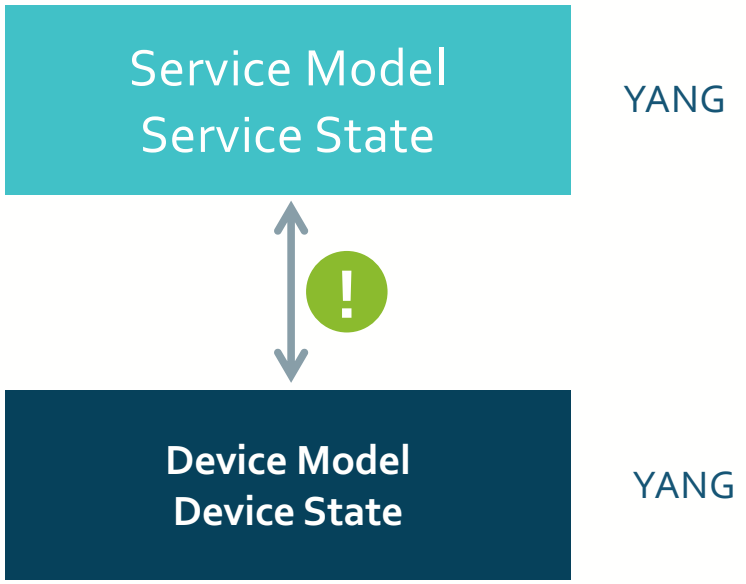
Rolf Eberhard,
Head of Service Orchestration at
Hewlett Packard Enterprise, 2016



FASTMAP was born

Data model transformation

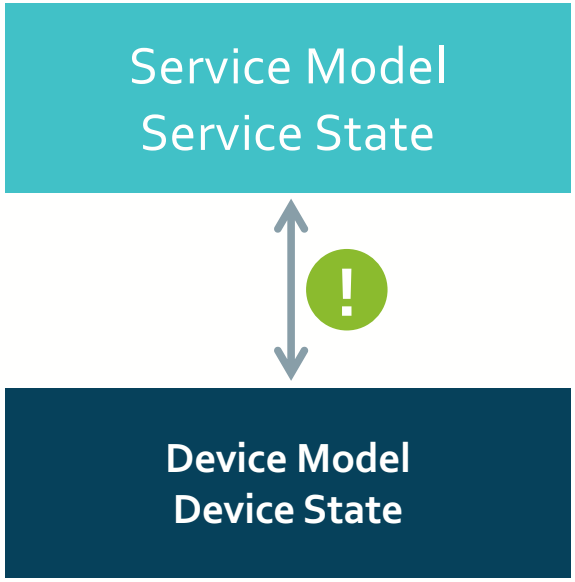
- Let the programmer express the simplest case (CREATE)
  - **Not** as a flow or sequence of actions
  - As a transform
- Full service life-cycle support
  - Update D is inferred at run-time from C



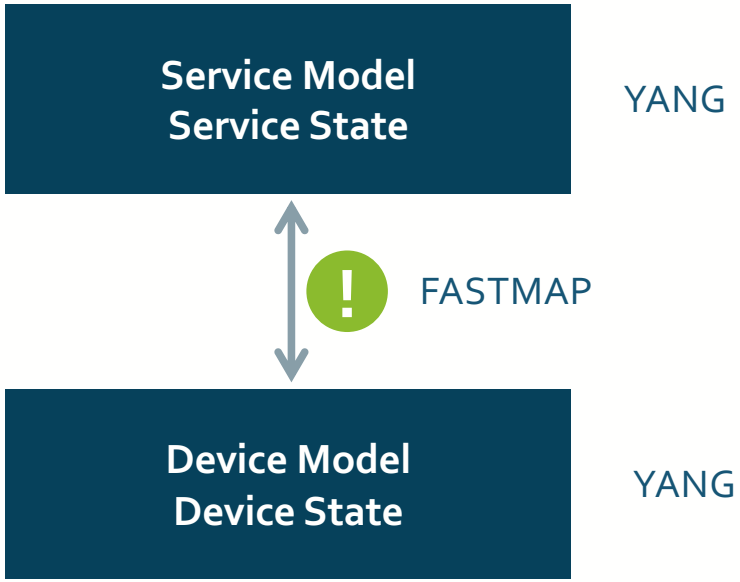
FASTMAP - the Unbeaten Value

- **Any** modification
  - Change QoS on branch
  - Modify Item in firewall rule list
- **Dependencies**
  - Change VLAN on service
  - Mode parameters
- **Redeploy** services
  - PE change from IOS to JunOS
- **Un-deploy**
  - Clean-up
- **Check-sync**

For Free



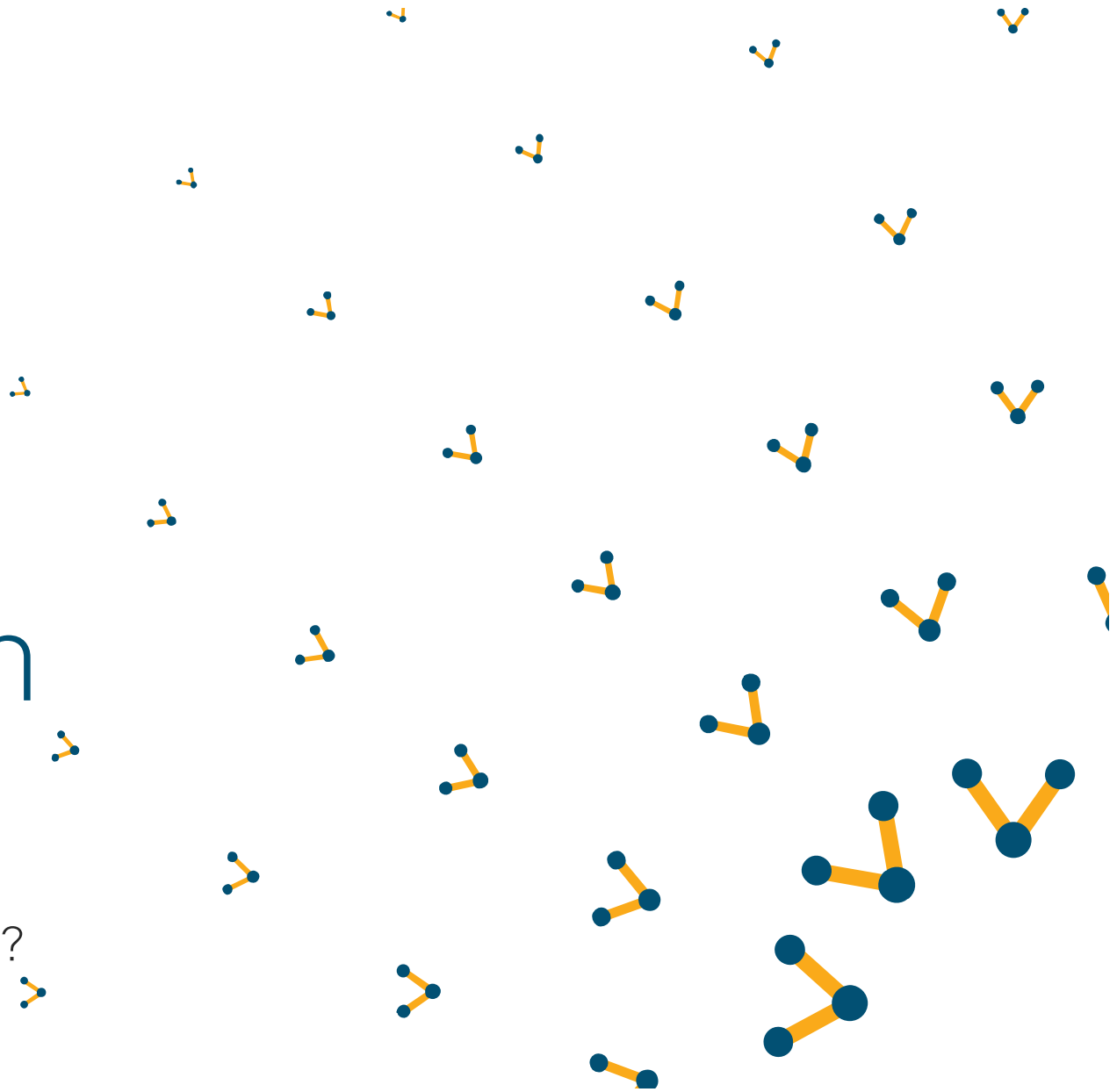
FASTMAP – The shortcut to…



Intent-Driven Orchestration

Designing Orchestration

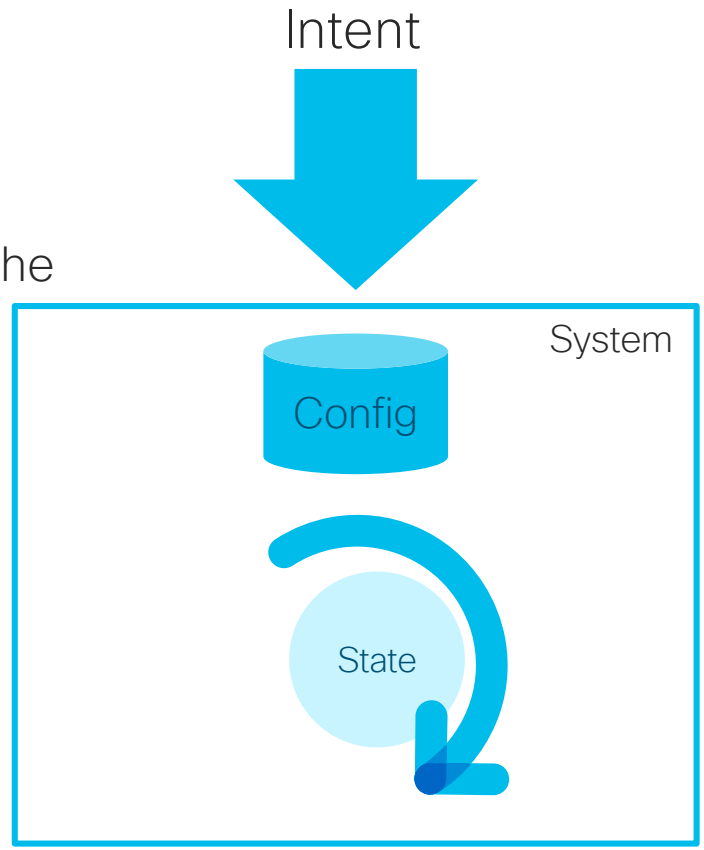
Intent Based Networking?
Intent-driven Orchestration?



Intent based interface principles

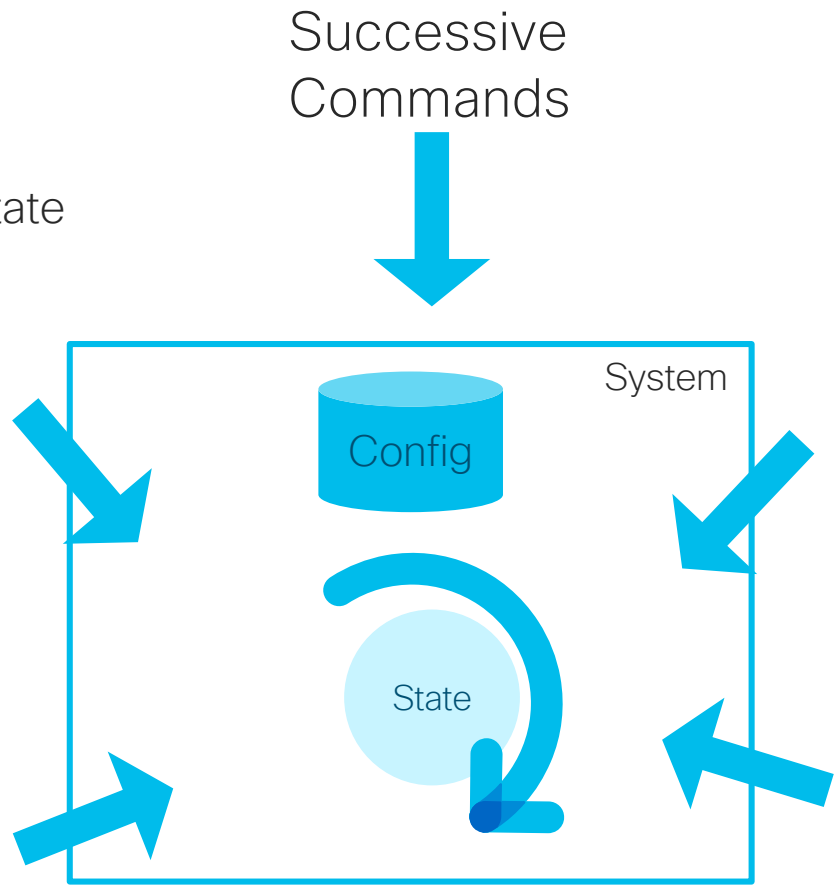
1. Writing your intent is enough
2. The system strives to execute on the intent
3. Intents are idempotent – multiple requests with the same intent has no additional effect
4. You can always write intent regardless of current state
5. The system never modifies received intent

Intent is configuration done right!

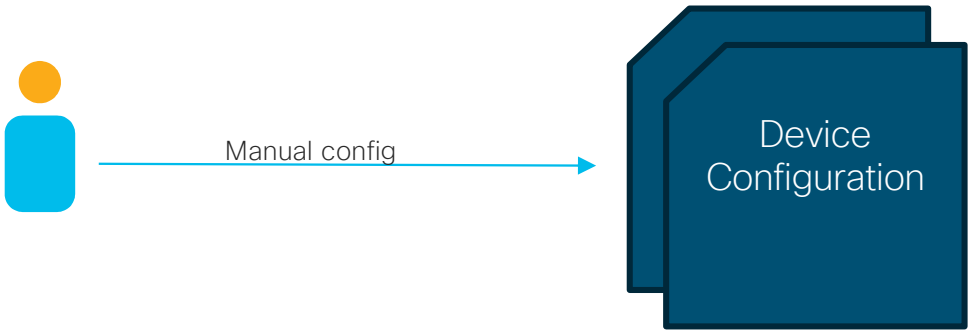


Command driven interfaces

1. Explicit commands to move between state
2. The correct command depends on the current state
3. The state is exposed to the user
4. Requires timed sequences of commands to achieve complex effects
5. Automation is by workflows/runbooks

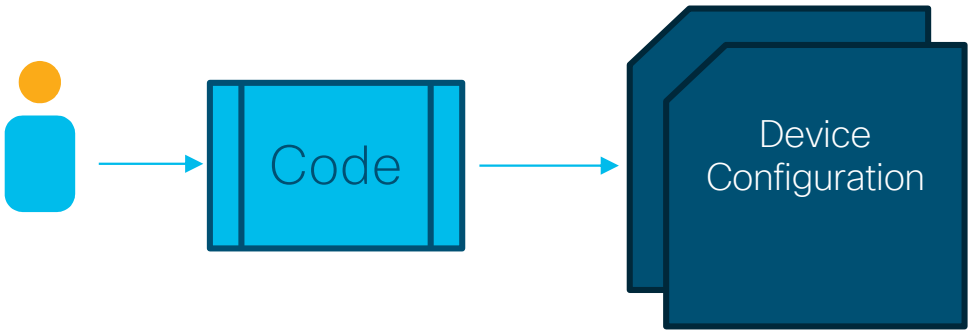


Manual operation



Feature	How?
Configuration	Manual
Roll-back	Manual
Compliance check	Manual
Service creation	Manual
Service modification	Manual
Service deletion	Manual
Brown-field	Manual
Cross-device coherence	Manual
Human approval	Automatic
Progress information	Automatic

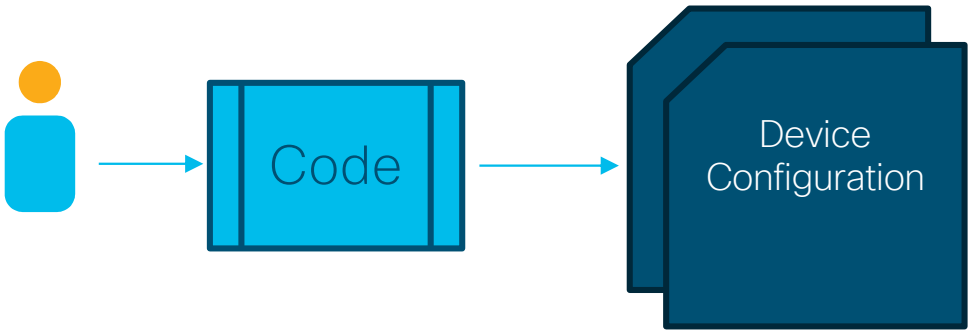
Scripting



Ansible, Salt, Nornir, NAPALM, …

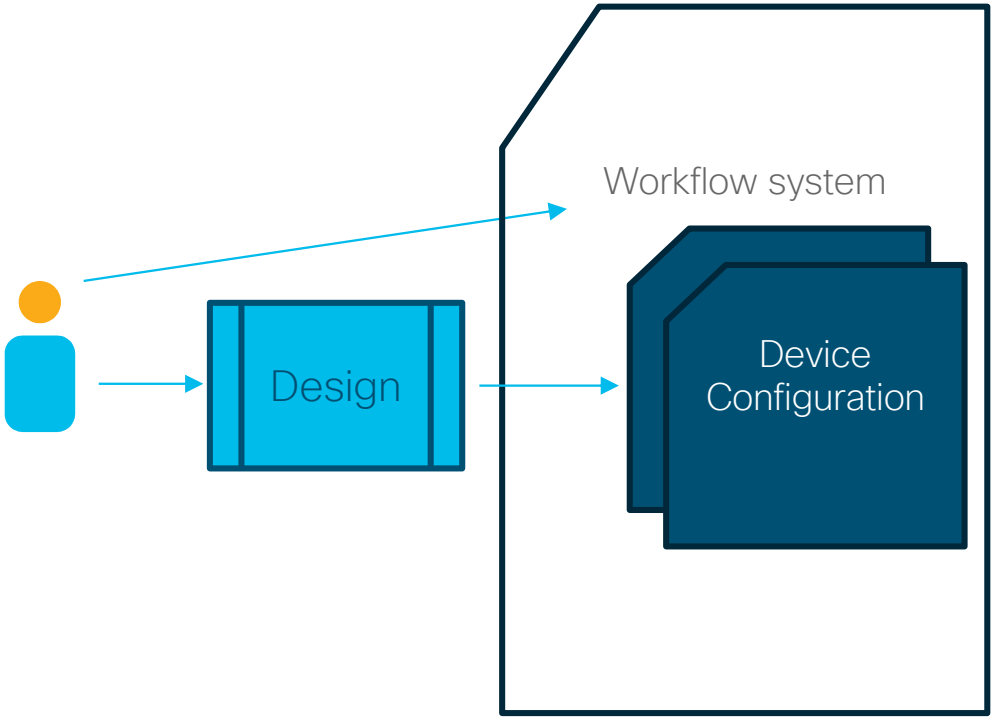
Feature	How?
Configuration	Explicit
Roll-back	Explicit programming
Compliance check	Explicit programming
Service creation	Explicit programming
Service modification	Explicit programming
Service deletion	Explicit programming
Brown-field	Manual
Cross-device coherence	Manual
Human approval	Explicit
Progress information	Explicit programming

Scripting + “Service Designer”



Feature	How?
Configuration	Explicit
Roll-back	Explicit programming
Compliance check	Explicit programming
Service creation	By designer
Service modification	By designer(?)
Service deletion	By designer(?)
Brown-field	Manual
Cross-device coherence	Manual
Human approval	Explicit
Progress information	Explicit programming

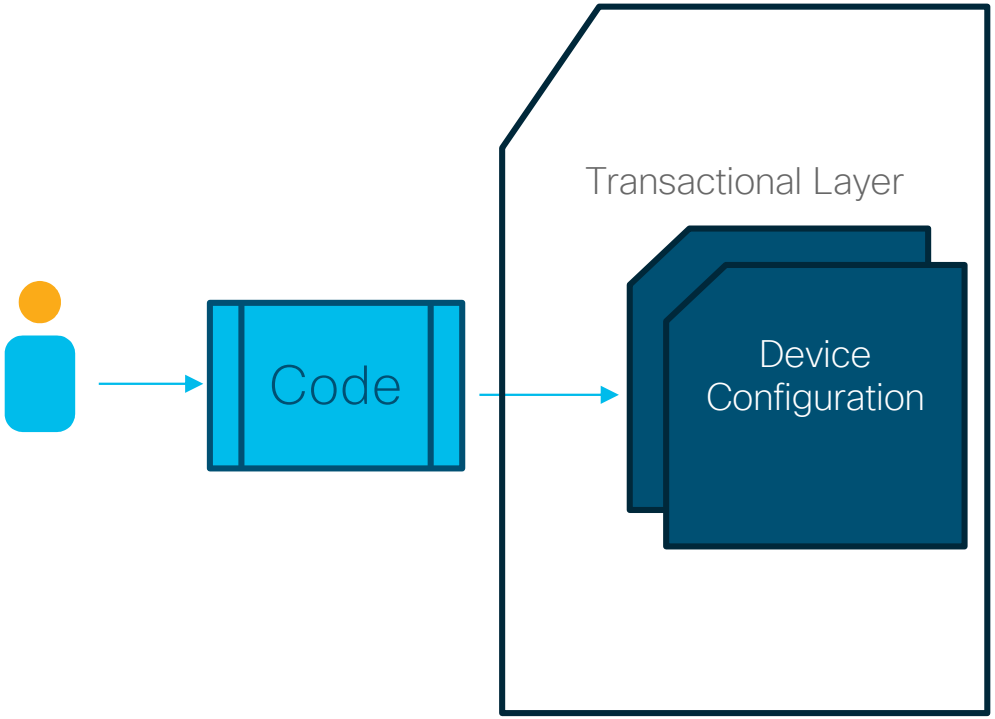
Scripting + “Service Designer” + Workflow



Feature	How?
Configuration	Explicit
Roll-back	Explicit programming
Compliance check	Explicit programming
Service creation	By designer
Service modification	By designer(?)
Service deletion	By designer(?)
Brown-field	Manual
Cross-device coherence	Manual
Human approval	Dedicated UI
Progress information	Dedicated UI

The classical **service activators**!

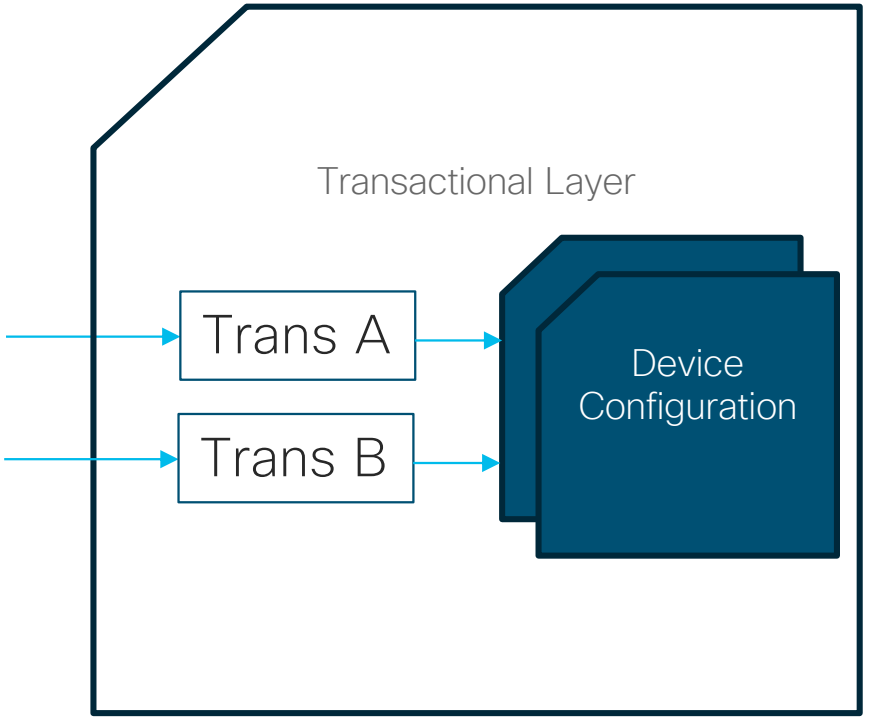
Transactions



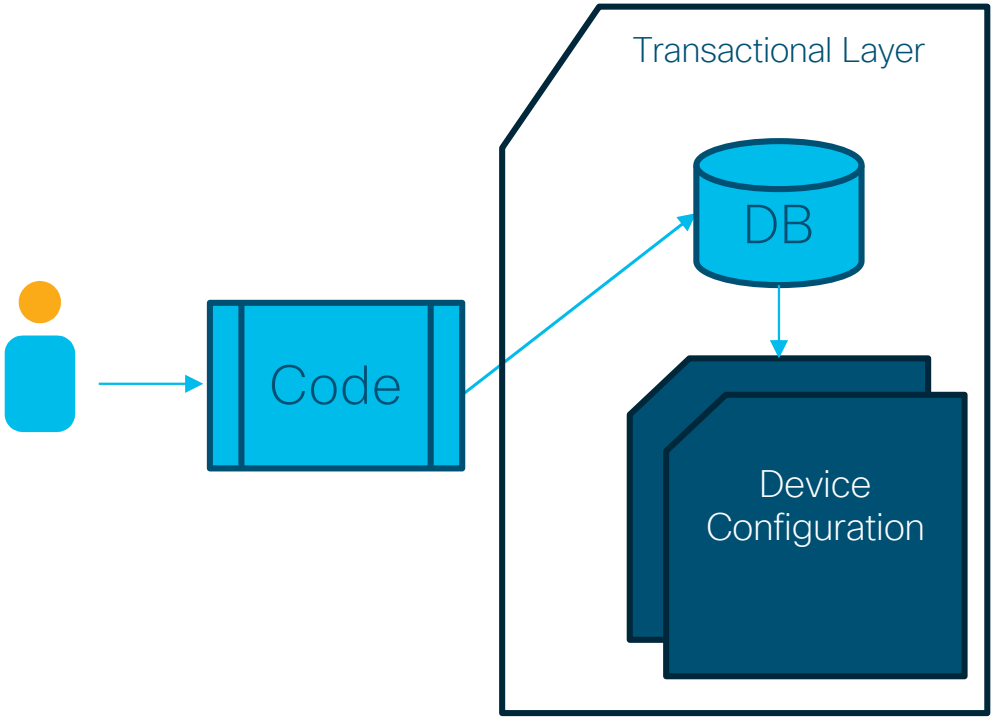
Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Explicit programming
Service creation	Explicit programming
Service modification	Explicit programming
Service deletion	Explicit programming
Brown-field	Manual
Cross-device coherence	Automatic
Human approval	Explicit
Progress information	Transaction result

What is a transaction?

- A mutable snapshot that can be committed to the network
- Allows you to read/write and then commit
- Multiple transactions open at once
- All-or-nothing when committing
- Guarantees consistency



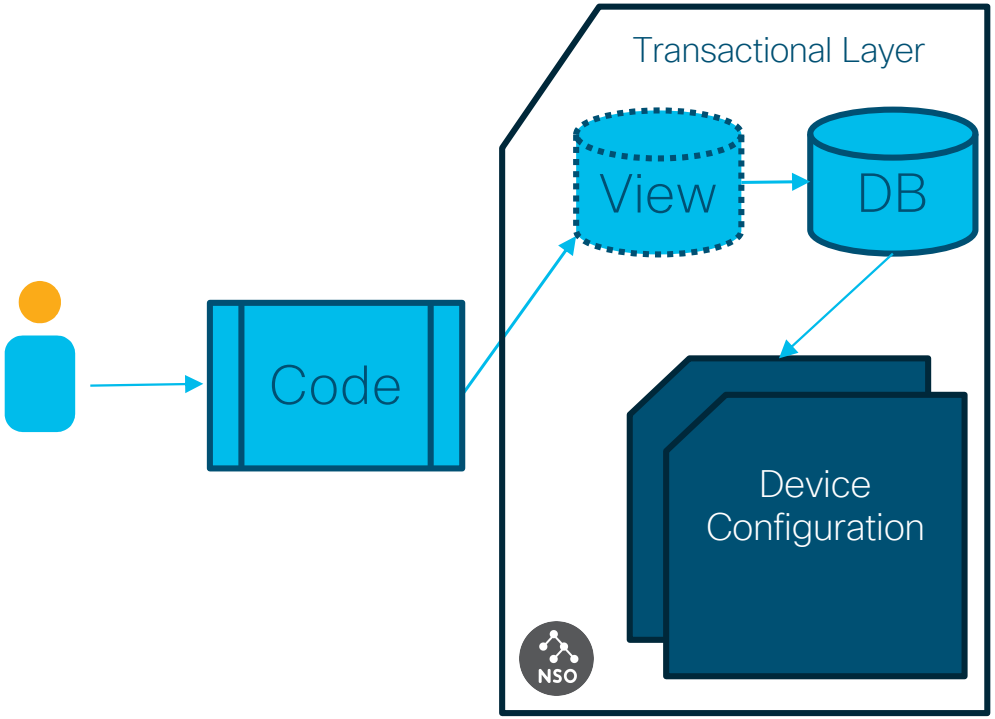
Transactions + Database



Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Automatic
Service creation	Explicit programming
Service modification	Explicit programming
Service deletion	Explicit programming
Brown-field	Manual
Cross-device coherence	Automatic
Human approval	Explicit
Progress information	Transaction result

Enables subscription, analysis, et c.

Transactions + DB + FASTMAP view

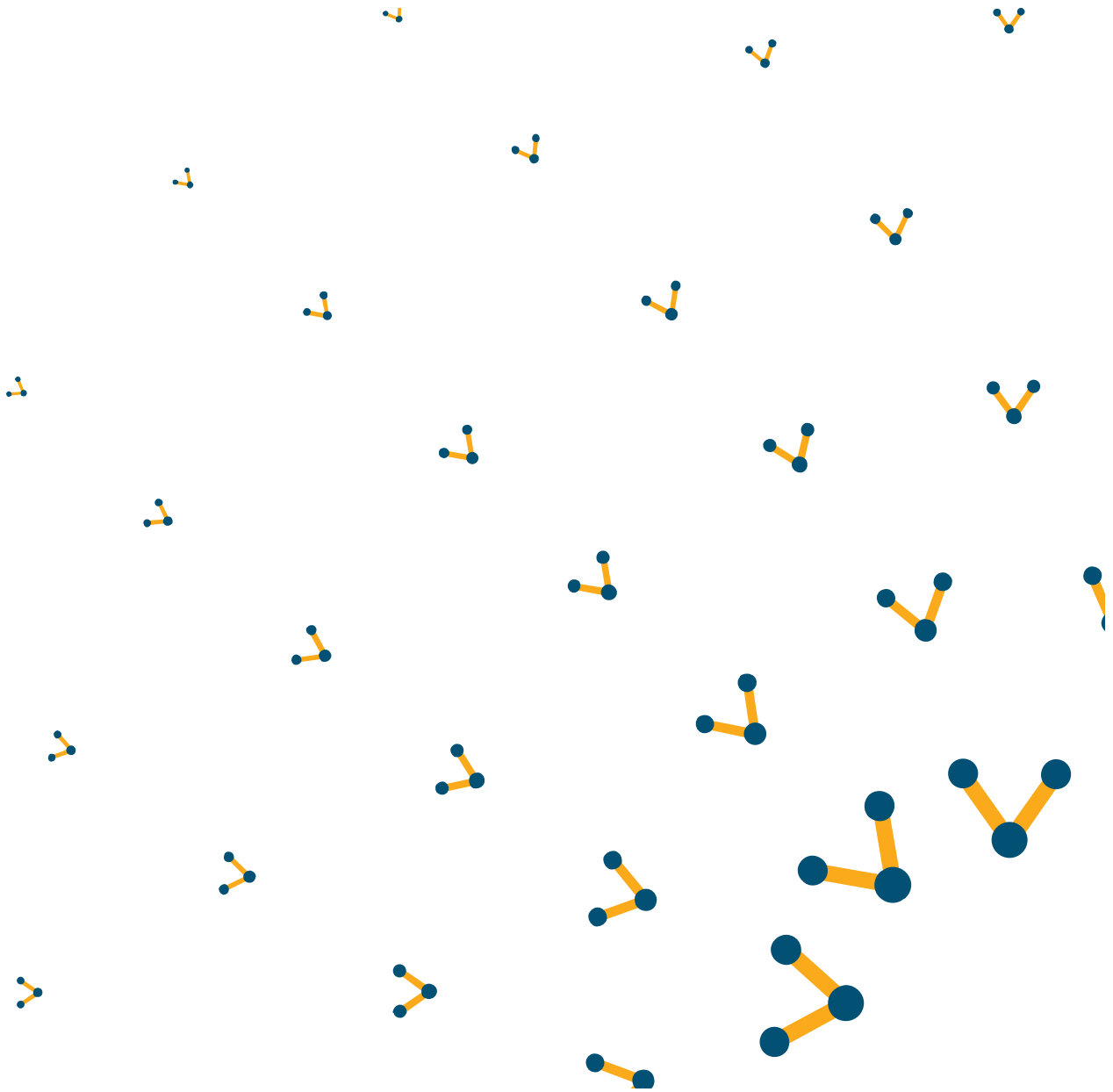


Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Automatic
Service creation	Explicit programming
Service modification	Automatic
Service deletion	Automatic
Brown-field	Integrity check, manual repair
Cross-device coherence	Automatic
Human approval	Explicit
Progress information	Transaction result

Database requirements

- **Support** for YANG schema/data model
  - Tree database
  - Validation according to the standards
- **Support** for transactions
  - Writeable snapshots that the orchestrator can manipulate
  - Atomicity / Roll-back
  - Global consistency on commit
- **Support** for subscribers and handlers for data
- **Support** for the calculation of diff-sets for each transaction

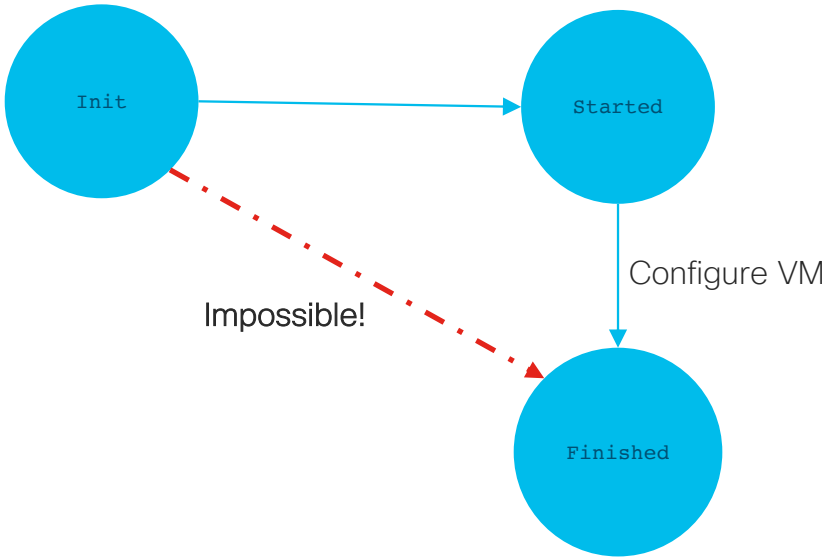
The next step:
Reactions



Problems

- What if it cannot be a single change?
- What if the intent depends on an operational state?
- What if a side-effect is needed?
- What if an underlying system is not purely intent based?

Virtual Machine Configuration



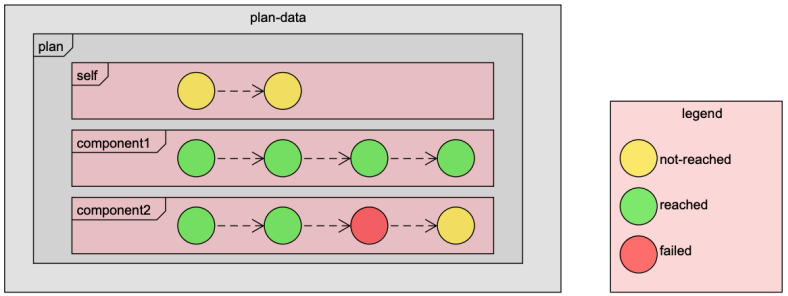
Solution: Reactive FASTMAP

- Putting dependencies on operational state into the service code.
- Adding a new primitive **`reactive-re-deploy`** to run FASTMAP again
- Adding kickers to trigger **`reactive-re-deploy`** when the operational state changes
- Allowing FASTMAP to react to progress
- Executing according to a plan
- Same FASTMAP, same database – new usage pattern

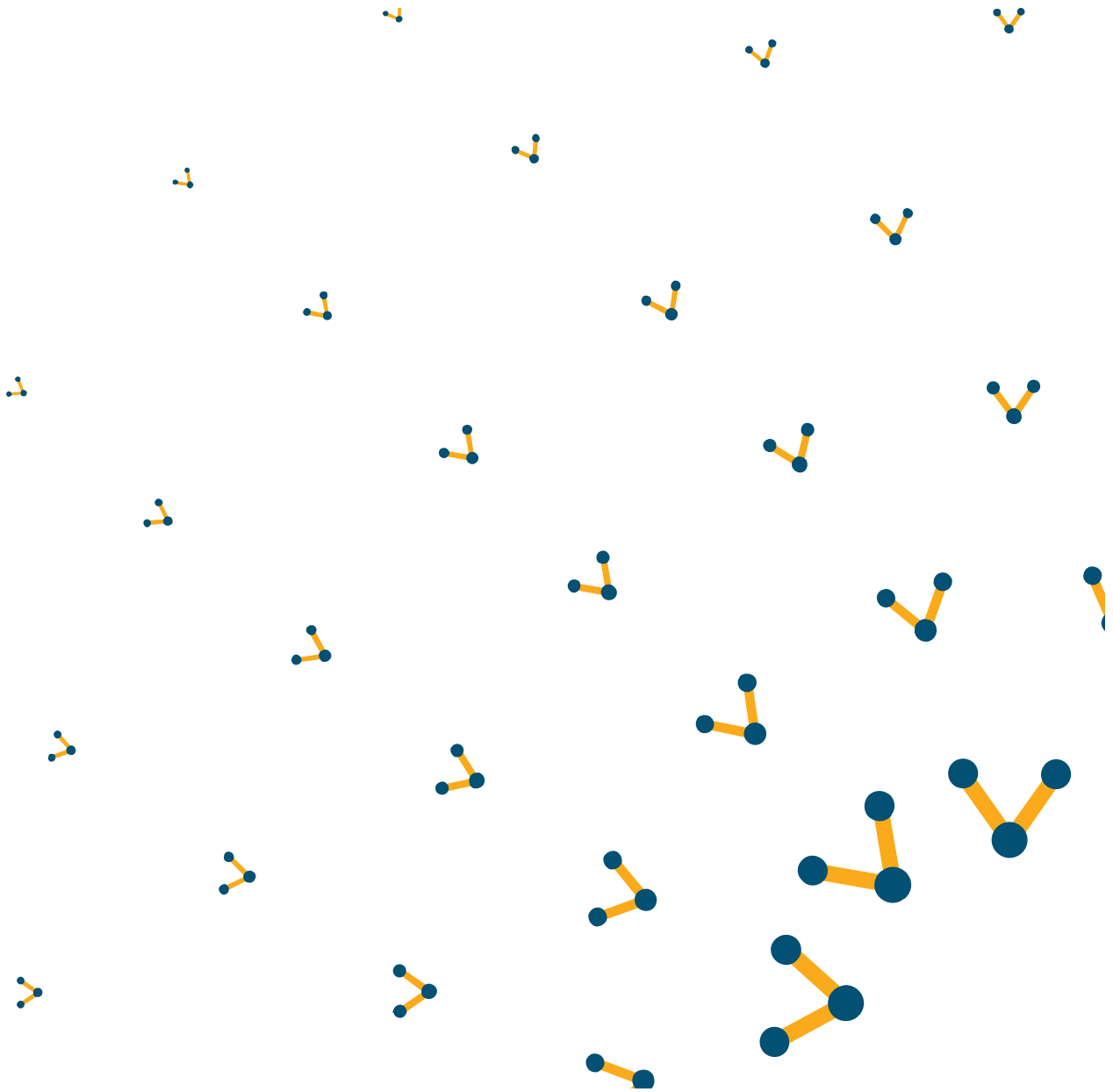
Nano Services: Life cycle for RFM

Tutorial tomorrow 9-11!

- With FASTMAP there is a single diffset
  - All reactions are merged together
  - Delete and update happens for all parts at once
- In some cases update/delete has to be gradual
- Nano-services allow for efficient life cycle implementation
- Model based plan evaluation

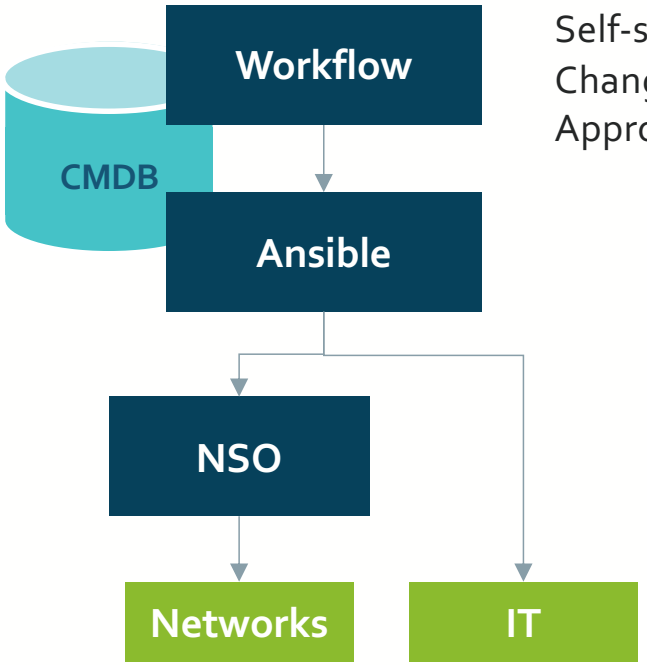


Conclusions



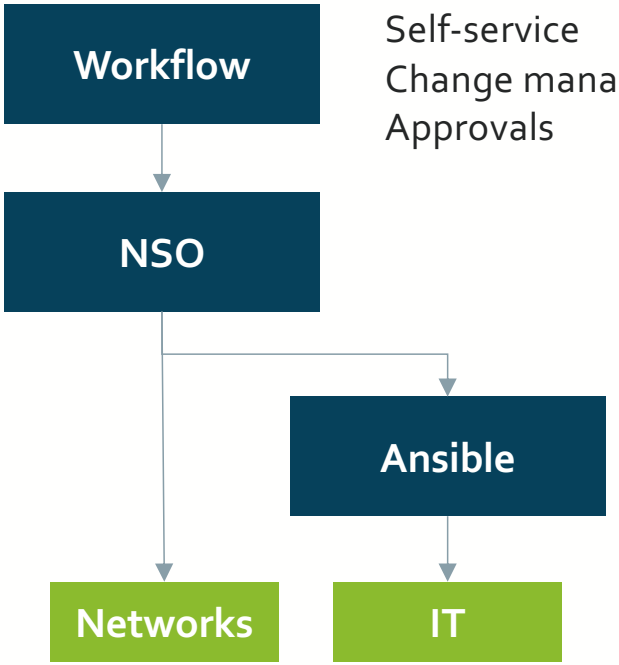
Can things go together - YES?

IT Application Centric organization



Self-service
Change management
Approvals

Network Service Centric



Self-service
Change management
Approvals

Lessons learned

Principles matter

Early design choices are the foundation to NSOs success

History repeats itself

Workflows are still there, and some people have forgotten what they are good at!

Conscious improvement

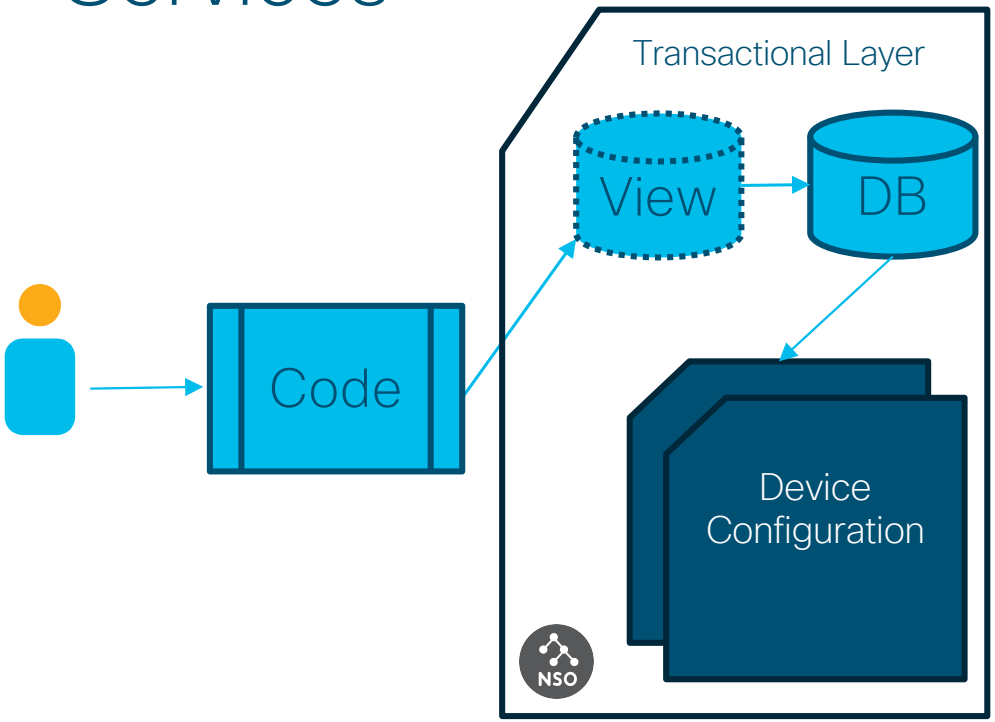
Changes to NSO must be compatible with the basic design philosophy

It works!

NSO is a mature product, and continuous improvements are being made

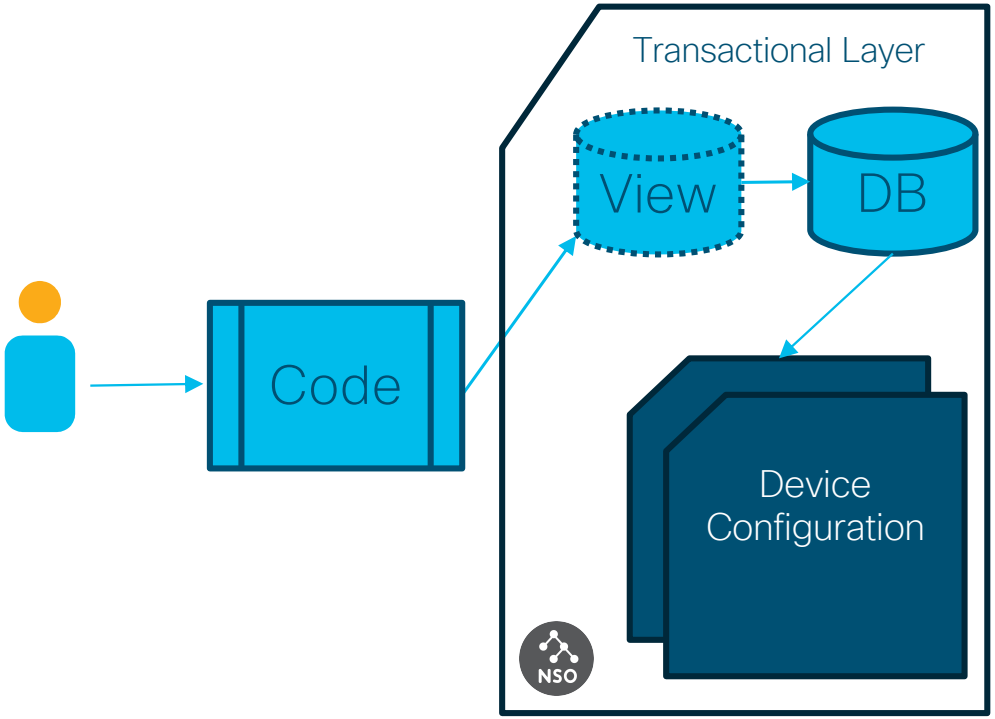

CISCO

Transactions + DB + FASTMAP view + Nano Services



Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Automatic
Service creation	Explicit programming
Service modification	Automatic
Service deletion	Automatic
Brown-field	Integrity check, manual repair
Cross-device coherence	Automatic
Human approval	Explicit
Progress information	Automatic

What is **next**?



Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Automatic
Service creation	Explicit programming
Service modification	Automatic
Service deletion	Automatic
Brown-field	Integrity check, manual repair
Cross-device coherence	Automatic
Human approval	Explicit
Progress information	Automatic