DeveloperDays
Network Services Orchestrator

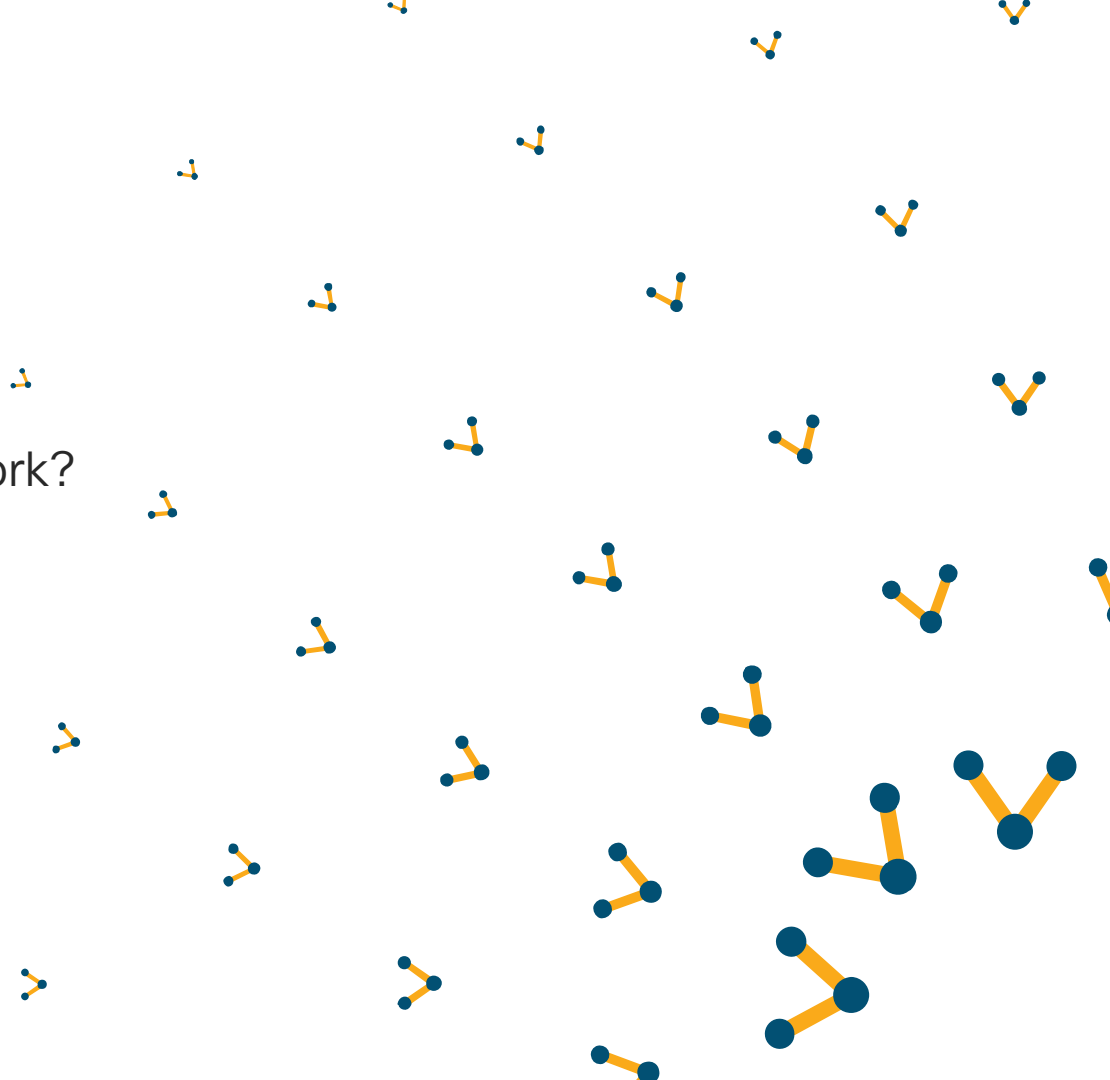
Workflow vs. NSO Service vs. Nano Service vs. Manual Configuration vs. Scripting vs. The World

Viktor Leijon – Technical Leader, Cisco

4th of December 2019

Agenda

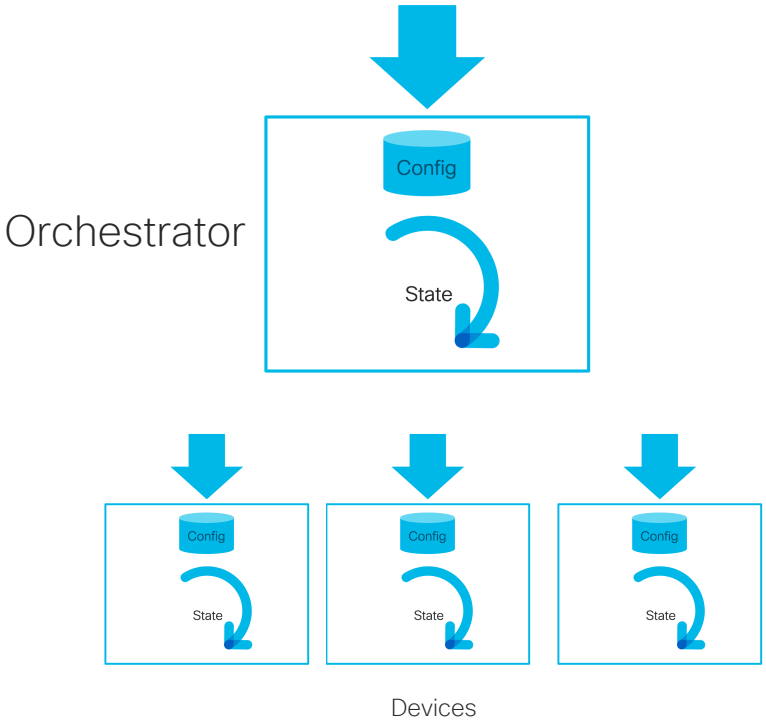
1. How can we affect a network?
2. Orchestration Options
3. Choosing Automation



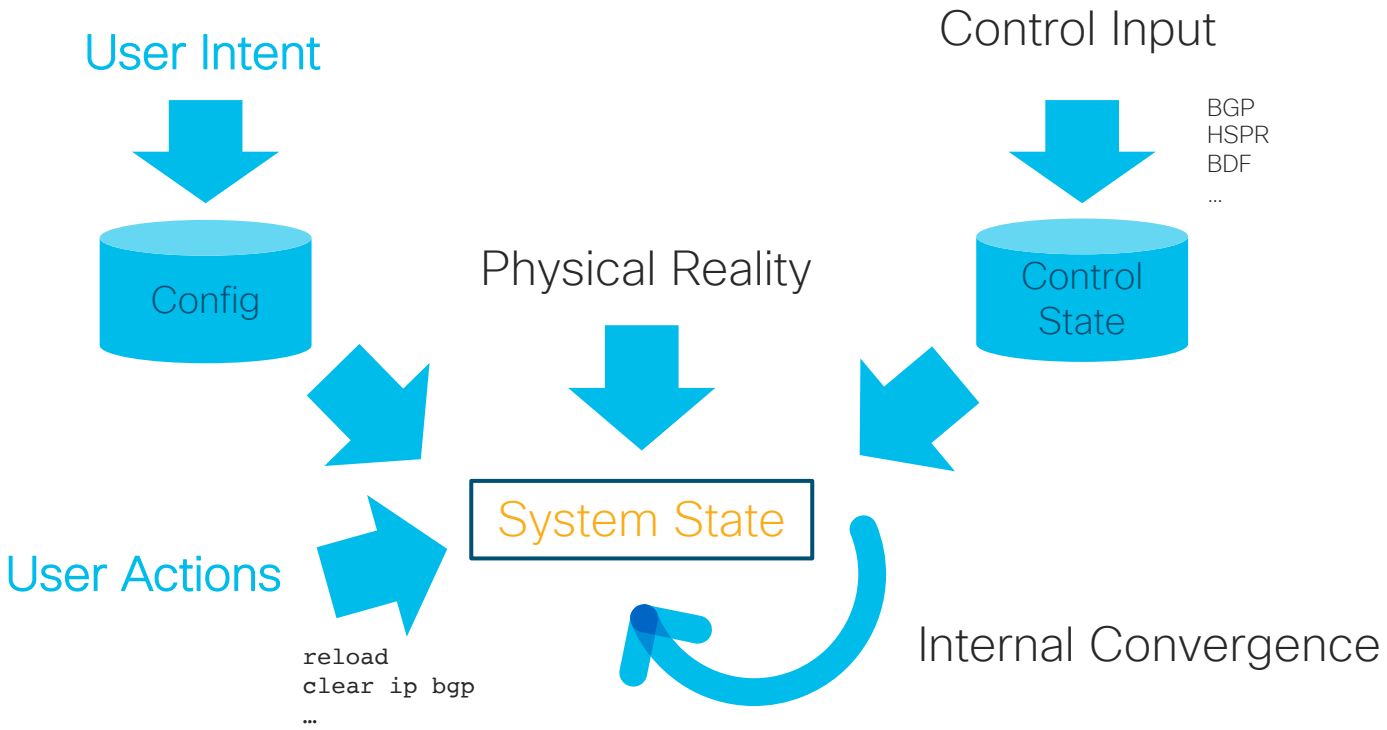
How can we affect a network?

Network Orchestration

- Moving the network to the right state
- Providing network level programmability
- Creates abstractions of the underlying technology



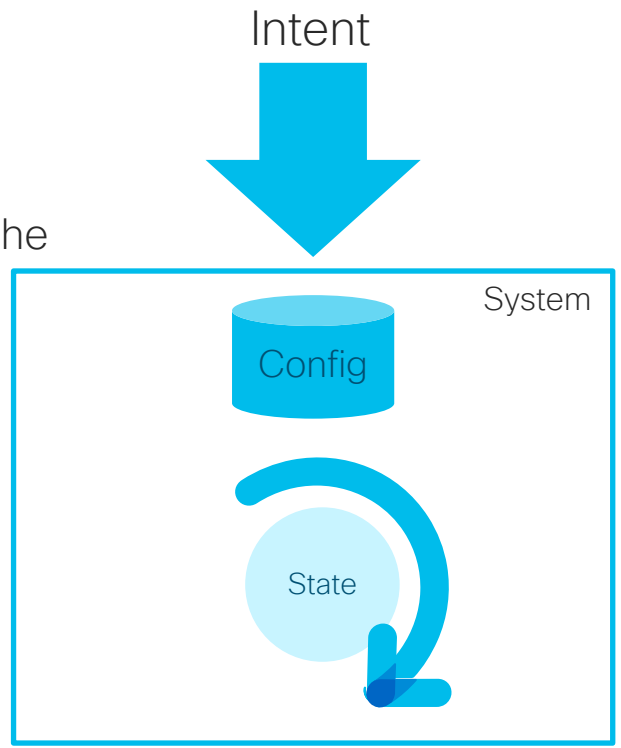
Causes of State



Intent based interface principles

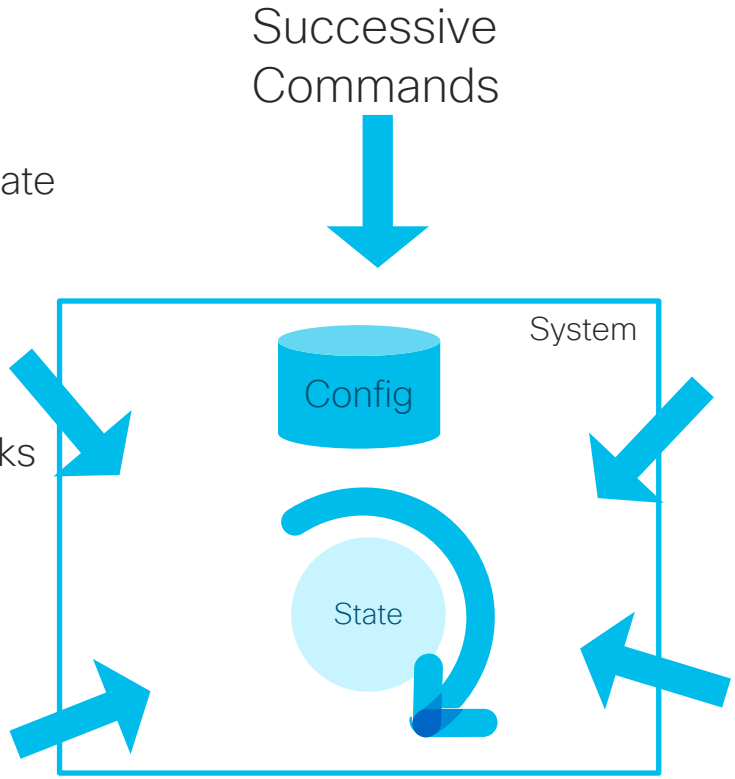
1. Writing your intent is enough
2. The system strives to execute on the intent
3. Intents are idempotent – multiple requests with the same intent has no additional effect
4. You can always write intent regardless of current state
5. The system never modifies received intent

Intent is configuration done right!



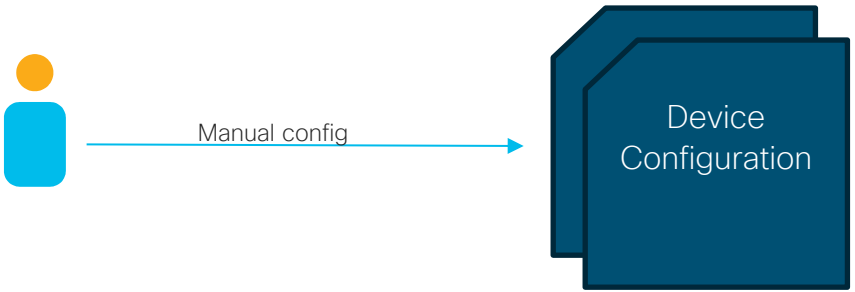
Command driven interfaces

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. Traditionally managed through workflows/runbooks



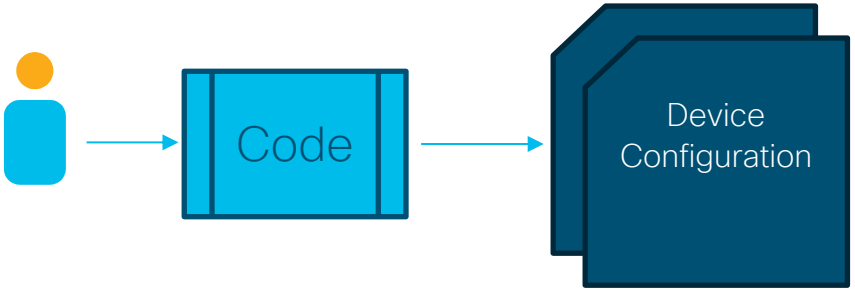
Orchestration Options

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
Multi-step configuration	Automatic
Progress information	Automatic

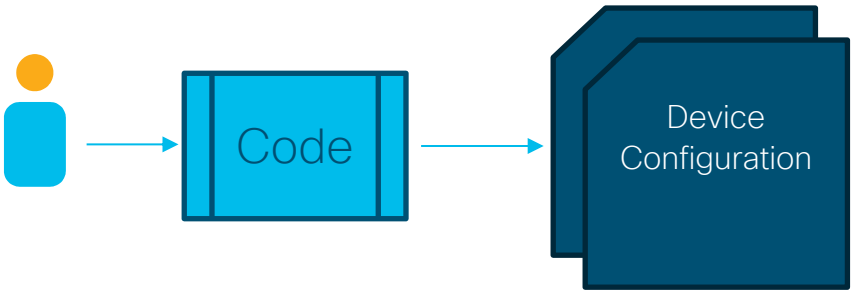
Scripting



Ansible, Salt, Nornir, NAPALM, …

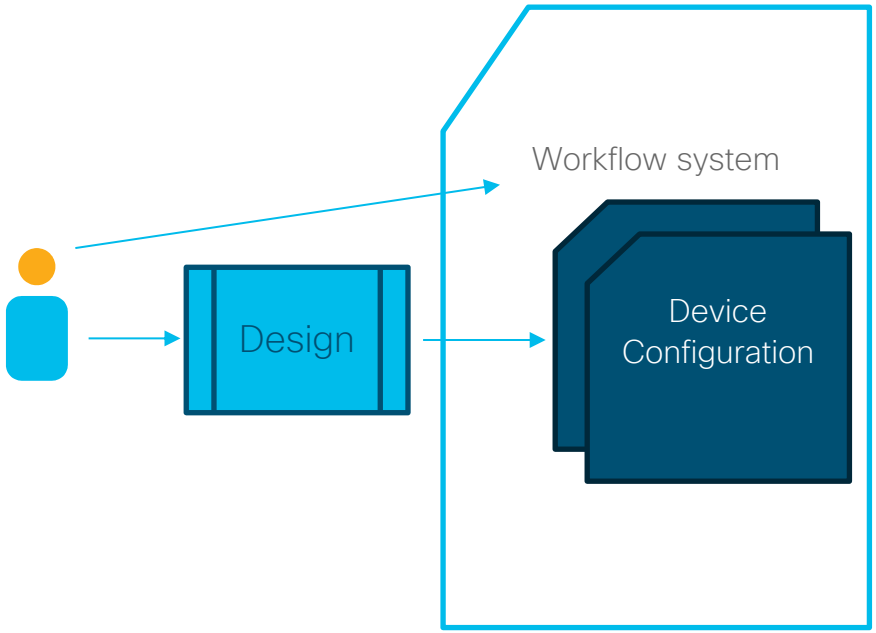
Feature	How?
Configuration	Explicit
Roll-back	Explicit programming
Compliance check	Explicit programming
Service creation	Explicit programming
Service modification	Explicit programming
Service deletion	Explicit programming
Brown-field	Manual
Cross-device coherence	Manual
Human approval	Explicit
Multi-step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

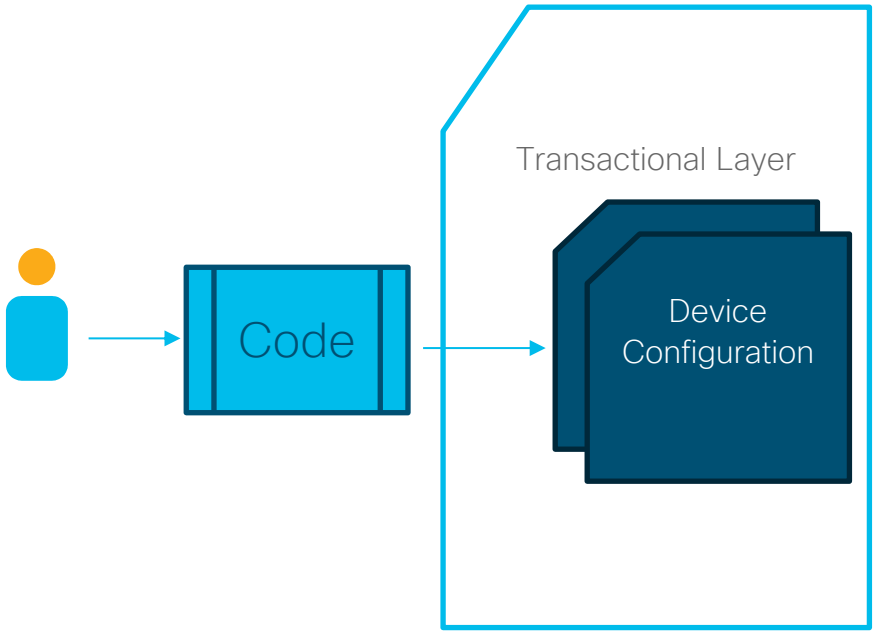
Scripting + “Service Designer” + Workflow



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
Multi-step configuration	By Workflow
Progress information	Dedicated UI

The classic **service activators!**

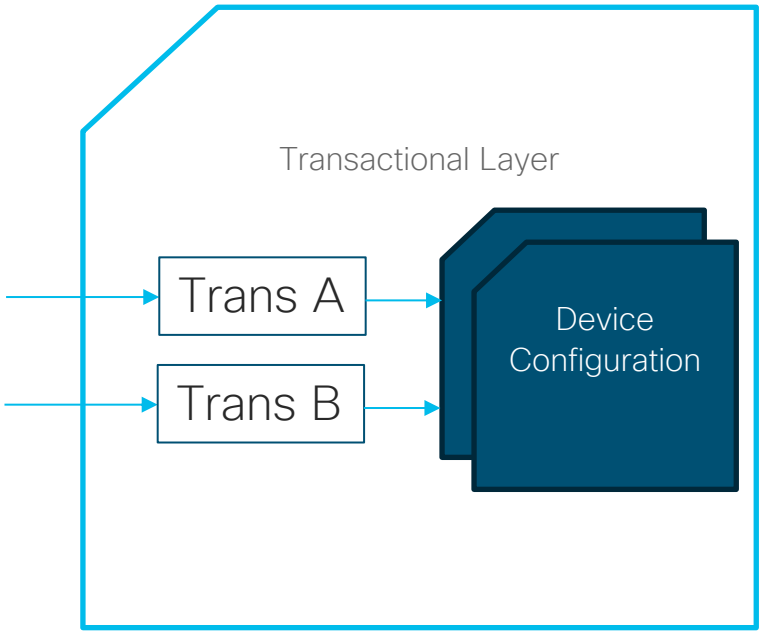
Scripting + Transactions



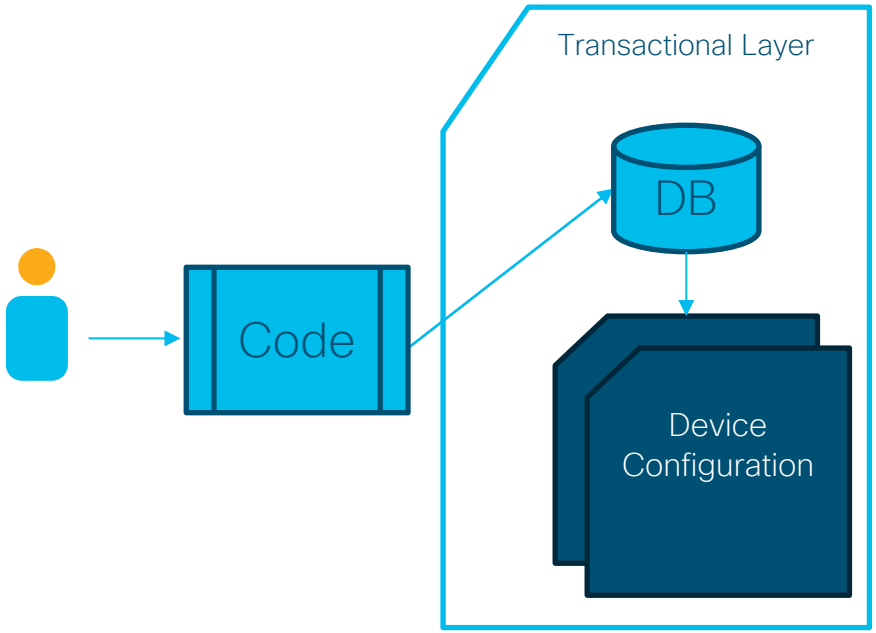
Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Explicit programming
Service creation	Explicit programming
Service modification	Explicit programming
Service deletion	Explicit programming
Brown-field	Manual
Cross-device coherence	Automatic
Human approval	Explicit
Multi-step configuration	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 ACID properties



Scripting + Transactions + Database



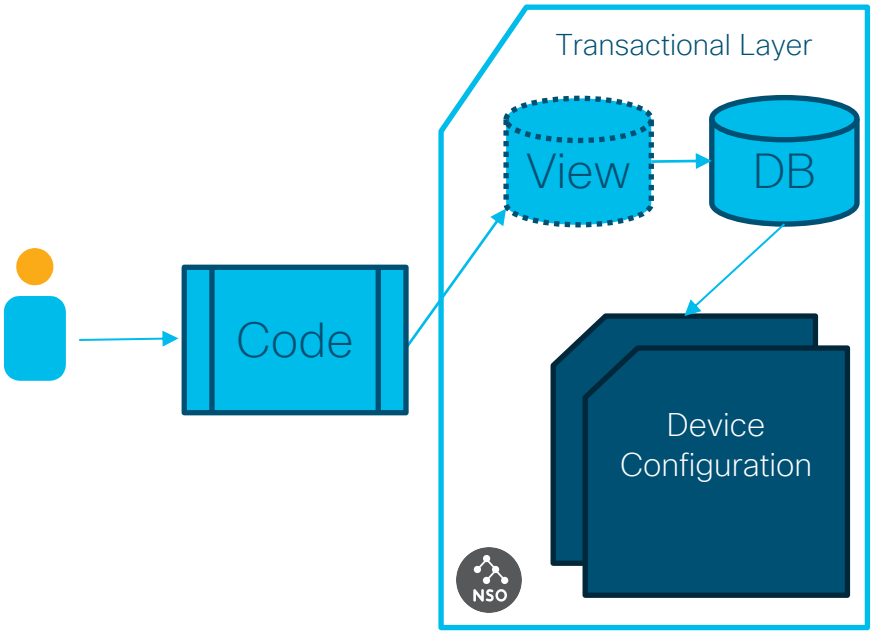
Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Automatic
Service creation	Explicit programming
Service modification	Explicit programming
Service deletion	Explicit programming
Brown-field	Explicit programming
Cross-device coherence	Automatic
Human approval	Explicit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

Enables subscription, analysis, et c.

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

Transactions + DB + FASTMAP view

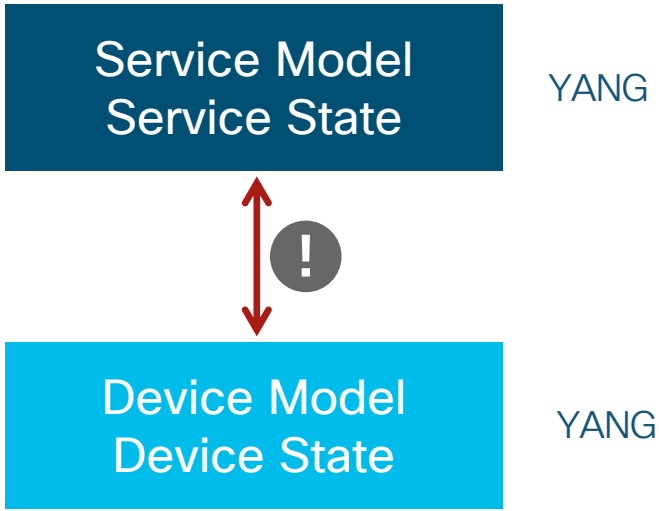


Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Automatic
Service creation	Explicit programming
Service modification	Automatic
Service deletion	Automatic
Brown-field	Integrity check, manual repair
Cross-device coherence	Automatic
Human approval	Explicit
Multi-step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

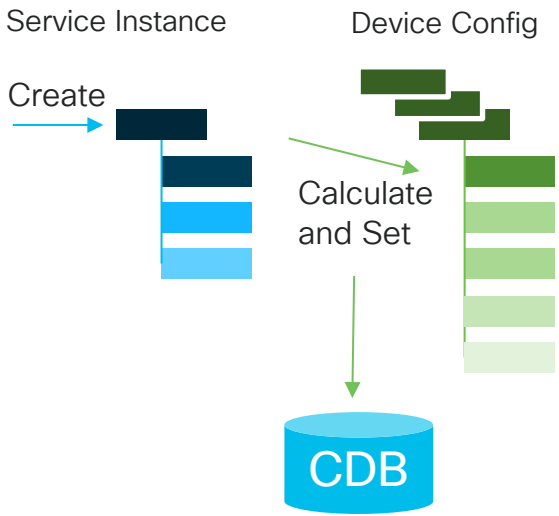
FASTMAP Explained

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/Delete is inferred at run-time from Create
- Relies on having a transactional database



FASTMAP Explained - Create

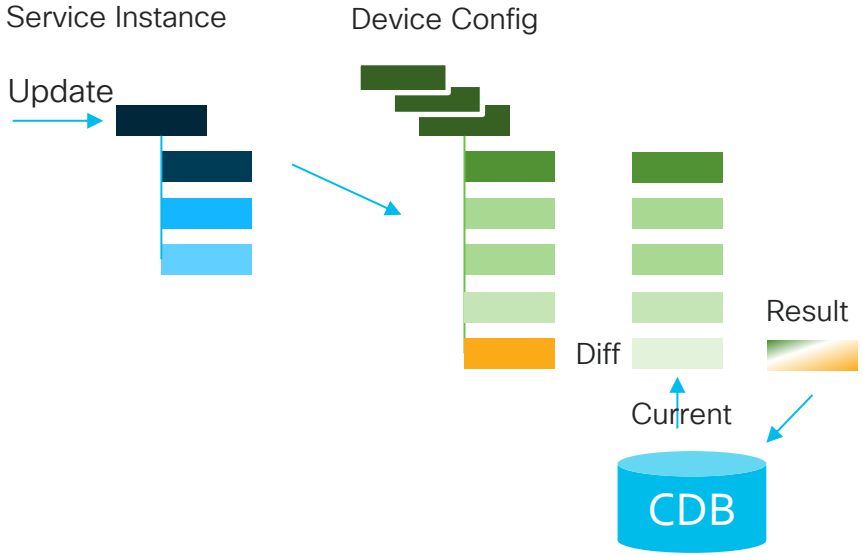



```
admin@ncs# vpn l3vpn volvo get-modifications reverse
cli {
    local-node {
        data  devices {
                  device ce1 {
                      config {
-                         ios:policy-map volvo {
-                             class class-default {
-                                 shape {
-                                     average {
-                                         bit-rate 6000000;
-                                     }
-                                 }
-                             }
-                         }
```

Reverse difference stored as operational data in service

"What should be done to the network to bring it back to the state before the service was instantiated?"

FASTMAP Explained - Update




```
admin@ncs(config)# vpn l3vpn volvo
admin@ncs(config-l3vpn-volvo)# qos qos-policy SILVER
admin@ncs(config-l3vpn-volvo)# !
admin@ncs(config-l3vpn-volvo)# top
admin@ncs(config)# commit dry-run
cli {
    local-node {
        data  devices {
                  device ce1 {
                      config {
                          ios:ip {
                              access-list {
                                  extended {
+                                     ext-named-acl GLOBAL-call-signaling {
+                                         ext-access-list-rule "permit tcp any any range 5060 5061";
+                                     }
+                                     ext-named-acl GLOBAL-ssh {
+                                         ext-access-list-rule "permit tcp any any range 22 22";
+                                     }
+                                     ext-named-acl GLOBAL-voice {
+                                         ext-access-list-rule "permit udp any any range 16348 32767
+                                     }
                                  }
                              }
                          }
+                         ios:class-map BUSINESS-CRITICAL {
+                             match-any;
+                             match {
+                                 access-group {
+                                     name GLOBAL-ssh;
+                                 }
+                             }
```

```
admin@ncs# vpn l3vpn volvo get-modifications reverse
cli {
    local-node {
        data  devices {
                  device ce1 {
                      config {
                          ios:ip {
                              access-list {
                                  extended {
-                                     ext-named-acl GLOBAL-call-signaling {
-                                         ext-access-list-rule "permit tcp any any range
-                                     }
-                                     ext-named-acl GLOBAL-ssh {
-                                         ext-access-list-rule "permit tcp any any range
-                                     }
-                                     ext-named-acl GLOBAL-voice {
-                                         ext-access-list-rule "permit udp any any range
-                                     }
                                  }
                              }
                          }
-                         ios:class-map BUSINESS-CRITICAL {
-                             match-any;
-                             match {
-                                 access-group {
```

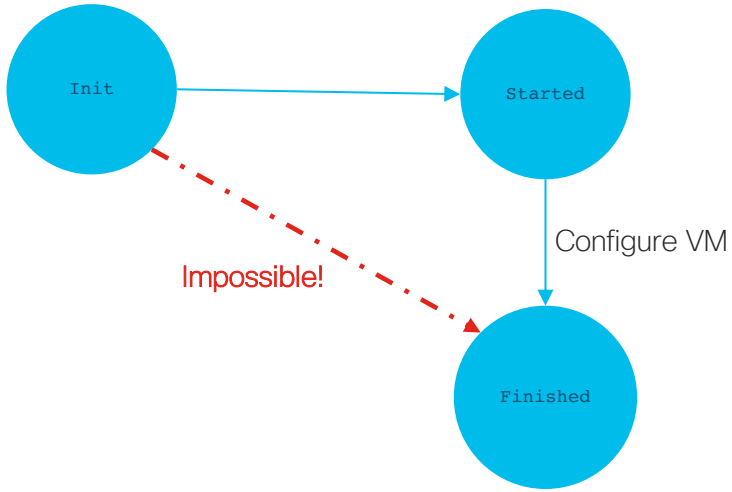
Reverse difference stored as operational data in service

"What should be done to the network to bring it back to the state before the service was instantiated?"

Problems

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

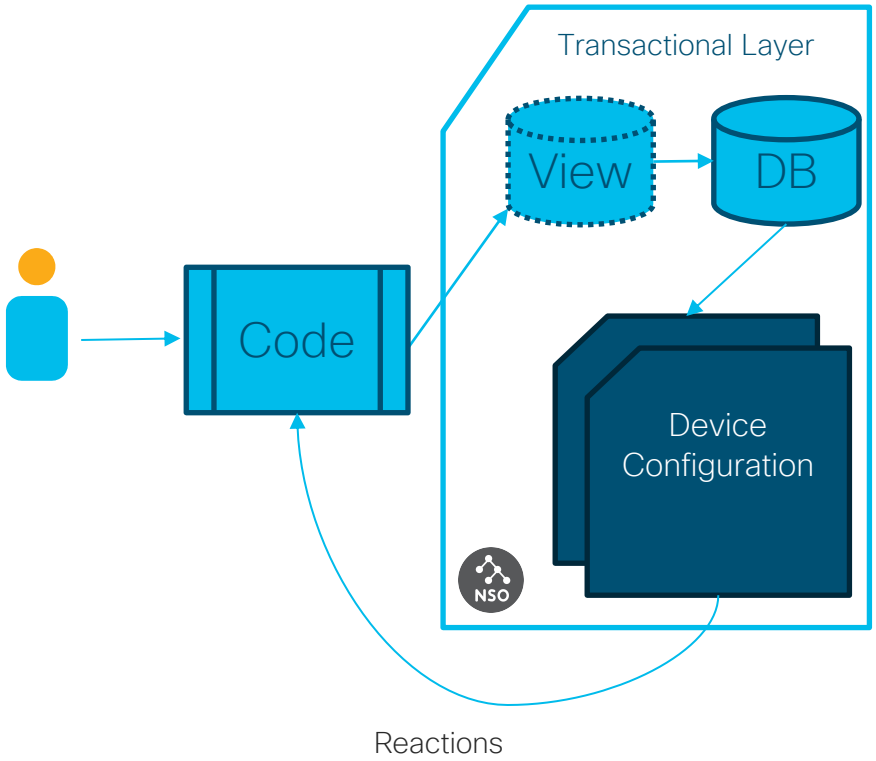
Virtual Machine Configuration



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

Transactions + DB + FASTMAP + ReactiveFASTMAP

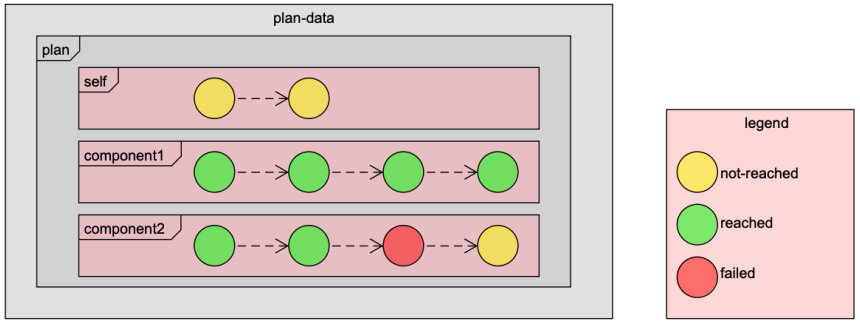


Feature	How?
Configuration	Explicit, optimized
Roll-back	Automatic
Compliance check	Automatic
Service creation	Explicit programming
Service modification	Automatic
Service deletion	Automatic
Brown-field	Integrity check, manual repair
Cross-device coherence	Automatic
Human approval	Explicit
Multi-step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

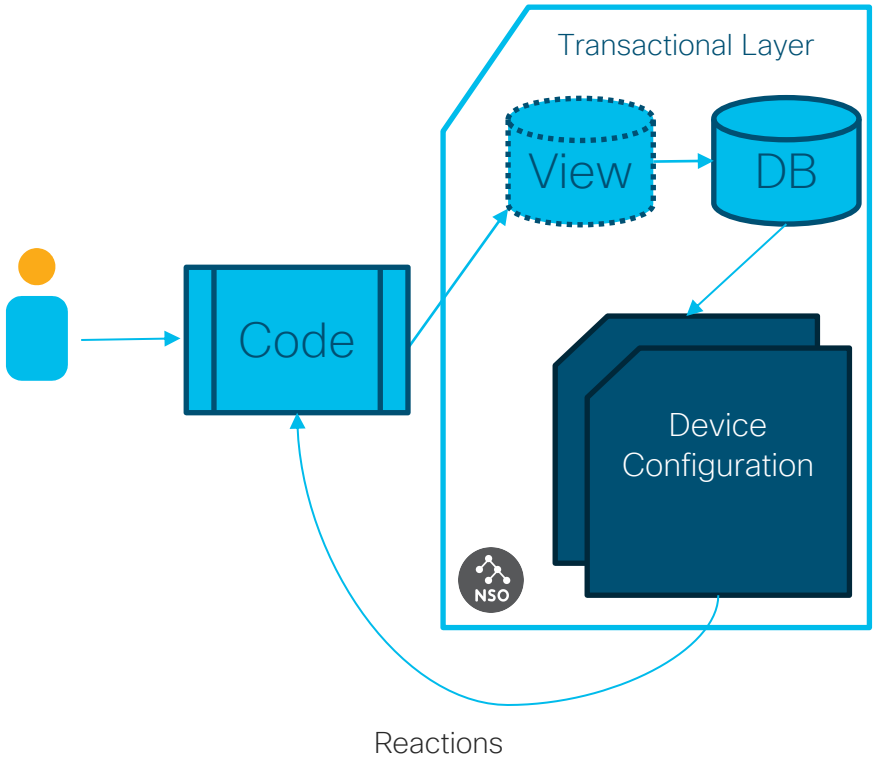
Nano Services: Life cycle for RFM

Tomorrow:
Lecture @10.45AM
Lab @1.15PM

- 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
- Declarative execution control

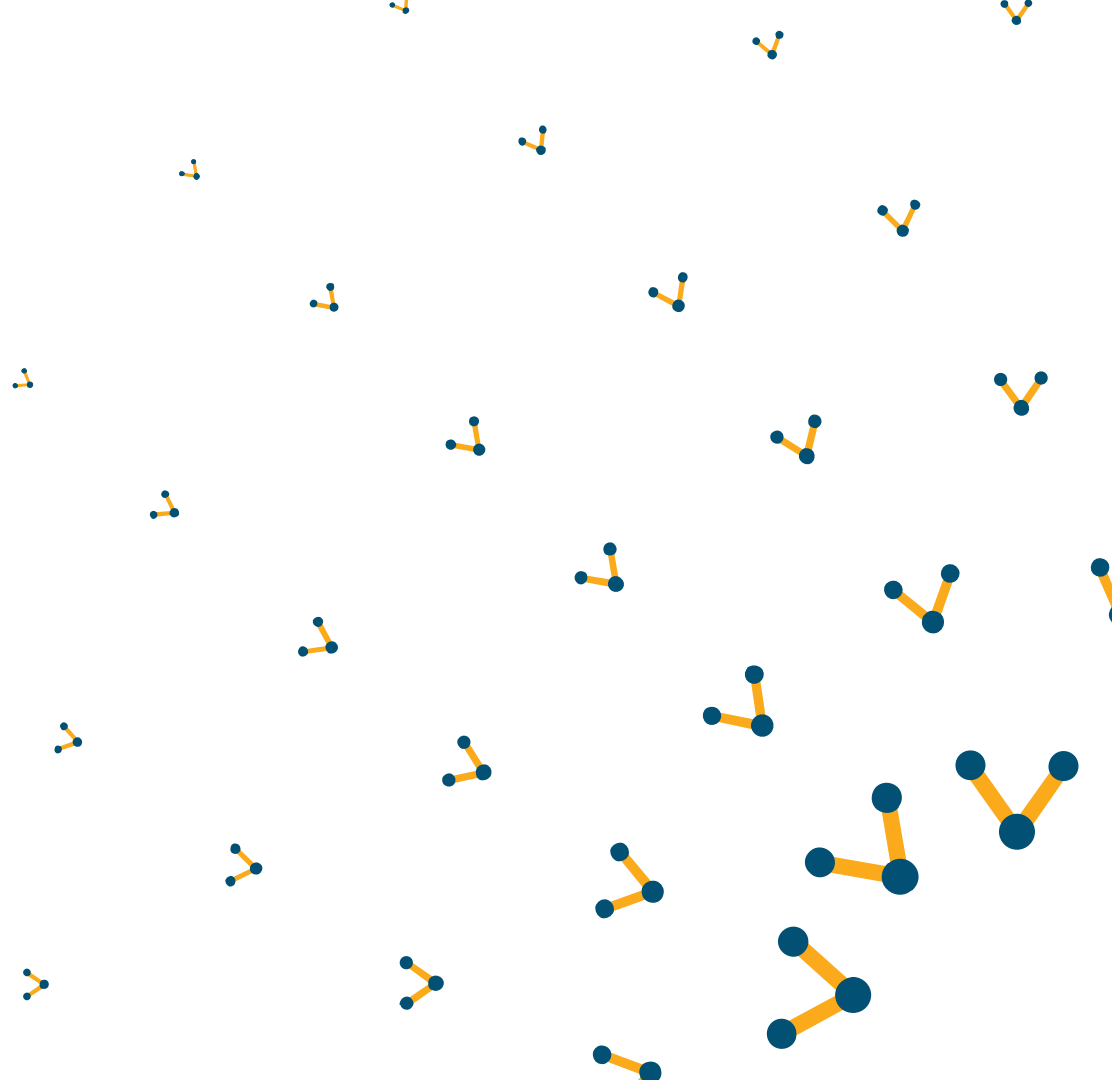


Transactions + DB + FASTMAP + NanoServices



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

Choosing Automation



Choice: Order or Outcome?

Order

- Traced back to a single user requests
- Flow-centered

+ Easy to trace completion of commercial orders

- In flight changes are a challenge
- Error attribution is hard

Outcome

- The sum of all requests
- Goal-oriented

+ Easy to understand the complete picture

+ Can integrate many different sources of intent

- Hard to trace individual orders and changes

Choice: Stateful or Stateless?

Stateful

- Keeps the state of all requests
- Uses a request and the state to decide what to do to a device

+ Can handle the most complex scenarios
+ Can provide advanced tools

- More complex model

Stateless

- No memory of previous requests
- Applies a request directly to a device

+ Simple model

- Difficult as complexity grows

Choice: Intent-based or Sequence of Actions?

Intent-based

- Transfer of responsibility to the system
- Aligned with an outcome-centric view

\+ Ideal as an interface to a closed-loop system
\+ Allows for modularity and composability

\- More difficult to interface with humans

Sequence of Actions

- Good for things that are one-off
- Aligned with an order-centric view

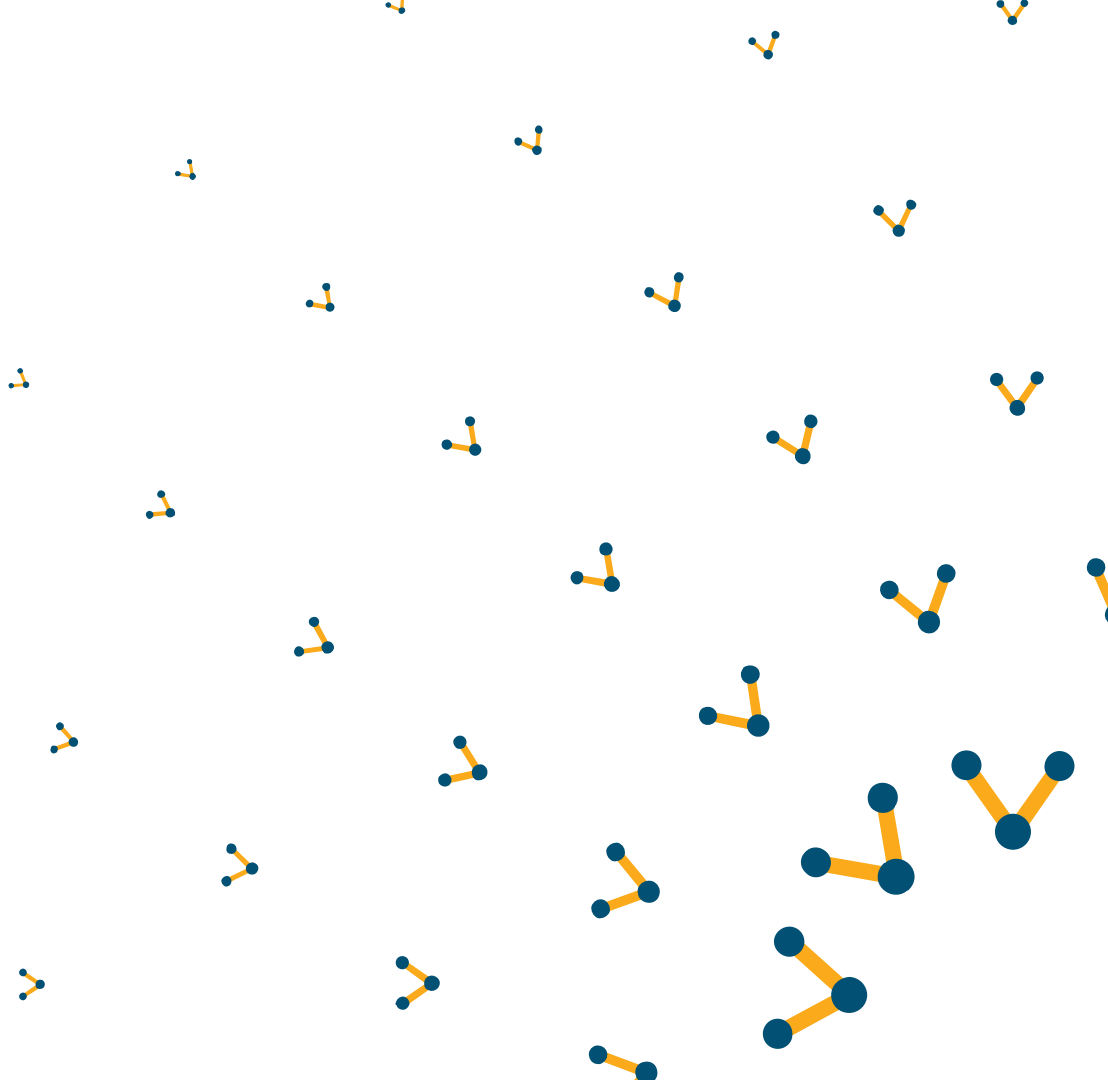
\+ Good interface for approval
\+ Can be combined with workforce management
\+ Easy to give a graphical interface

\- Hard to get error-handling right
\- Enforces sequentiality

Combining workflows and intent based orchestration?

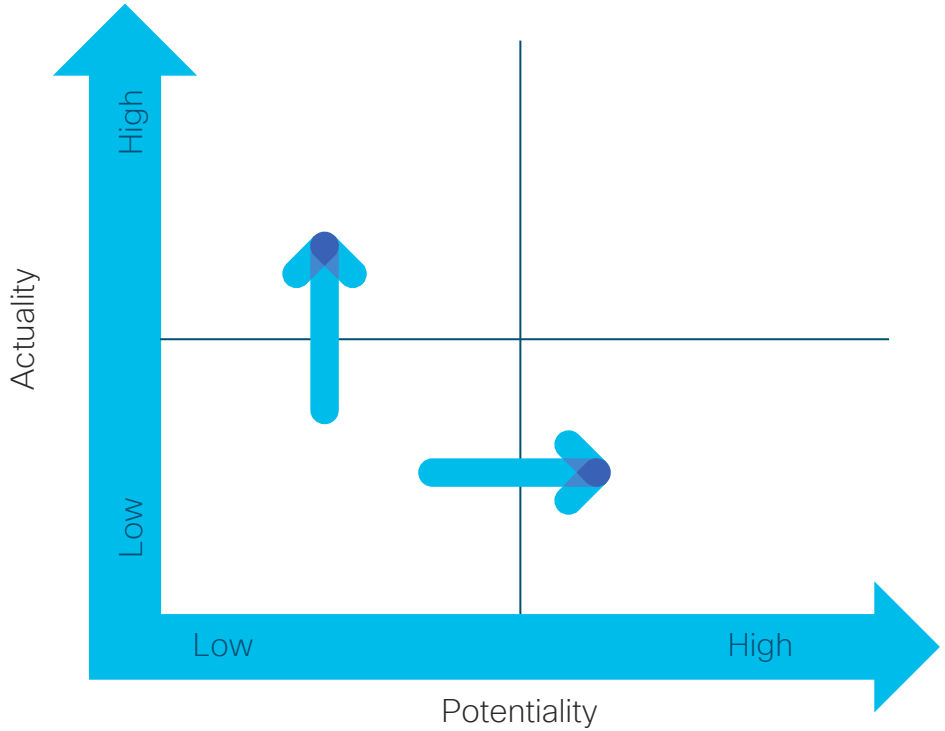
- Layered architecture
- Strict separation of concerns
  - Workflow system: Exclusive control over human interaction
  - Intent based system: Exclusive control over the network
- Workflow freedom
  - No need to understand technical details
  - No pass-through to the network provisioning system
- Ensure intent autonomy
  - All dependencies should be encapsulated in the intent
  - The intent based system never correlates or coordinates separate intents

Conclusions

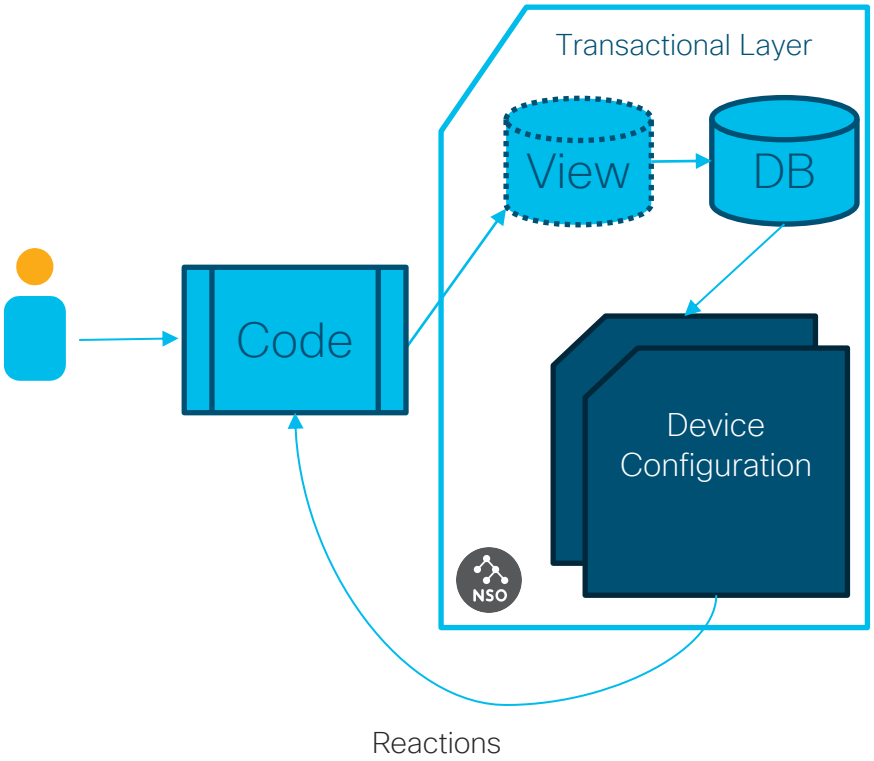


Factors

- Cost
- Complexity
- Ambition
- Human interactions
- System interactions
- Available skills
- Available components
- Future needs



Where does NSO go from here?



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

Lessons learned

Principles matter

Early design choices are the foundation to automation success

Choice is hard

It is important to carefully consider both technology and strategy

Different Tools

A complete solution probably integrates several different tools at different layers

It works!

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


CISCO