The Three Stages of Automation
Stage #3: Developing Services

2018-03-08
Today’s Presenters

John Malzahn
Senior Manager, Cloud and Virtualization Solutions Marketing
Cisco Systems

Carl Moberg
Technology Director, Cloud Solutions and Platform Group
Cisco Systems
Todays Agenda

1. Introducing the Three Stages of Automation
2. Deep Dive on Stage #2 Developing Services
3. Using NSO as a Service Development Platform
4. Demo Time!
5. Wrap-up and Q&A
Key Market Trend Observations

Execution at the speed of software
- Networks provide well-known utility abstractions
- Agility, DevOps, NFV, SDN drives new expectations

Changing customer behavior and new expectations
- Everything on demand
- New services with a press of a button

Rapidly changing business models
- Cloud services, virtualization, programmable networks
- New value chains including OTT Co-opetition

All of this requires successful, flexible automation. But complexity has destroyed many automation initiatives.
# Departmental Pain Points

<table>
<thead>
<tr>
<th>Department</th>
<th>Role</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Network Engineer</td>
<td>• Error-prone manual tasks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Growing backlog</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Virtualization is coming</td>
</tr>
<tr>
<td></td>
<td>Day-to-day management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>of rapidly growing,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>complex networks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Provisions services</td>
<td>• No service insight</td>
</tr>
<tr>
<td></td>
<td>and manages service</td>
<td>• Lack of automation</td>
</tr>
<tr>
<td></td>
<td>quality in networks</td>
<td>• Quality issues in delivery</td>
</tr>
<tr>
<td></td>
<td>Ops and Provisioning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Service Developers</td>
<td>Develops new network services on demand</td>
</tr>
<tr>
<td></td>
<td>“Automation”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Customer Experience”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Time-to-Market”</td>
<td></td>
</tr>
</tbody>
</table>

© 2018 Cisco and/or its affiliates. All rights reserved.
## Transition Towards Automation

<table>
<thead>
<tr>
<th>Network Engineer</th>
<th>Ops and Provisioning Team</th>
<th>Service Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Automation”</td>
<td>“Customer Experience”</td>
<td>“Time-to-Market”</td>
</tr>
<tr>
<td><strong>Day-to-day management of rapidly growing, complex networks</strong></td>
<td><strong>Provisions services and manages service quality in networks</strong></td>
<td><strong>Develops new network services on demand</strong></td>
</tr>
<tr>
<td><strong>Challenges</strong></td>
<td><strong>Challenges</strong></td>
<td><strong>Challenges</strong></td>
</tr>
<tr>
<td>• Error-prone manual tasks</td>
<td>• No service insight</td>
<td>• Implementation time</td>
</tr>
<tr>
<td>• Growing backlog</td>
<td>• Lack of automation</td>
<td>• Cost of change</td>
</tr>
<tr>
<td>• Virtualization is coming</td>
<td>• Quality issues in delivery</td>
<td>• Lack of tooling</td>
</tr>
</tbody>
</table>

1. **Network API**
   - Utilize a single interface to all network devices

2. **Service Abstraction**
   - Leverage one central API for all services

3. **Transformation**
   - Develop your own services

© 2018 Cisco and/or its affiliates. All rights reserved.
<table>
<thead>
<tr>
<th>Current Situation</th>
<th>Target Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>People</strong></td>
<td><strong>People</strong></td>
</tr>
<tr>
<td>Walled-off departments, lacking software skills</td>
<td>Cross-functional teams with software skills</td>
</tr>
<tr>
<td><strong>Process</strong></td>
<td><strong>Process</strong></td>
</tr>
<tr>
<td>Waterfall operations for network operation and services</td>
<td>DevOps approach across tools development and operations</td>
</tr>
<tr>
<td><strong>Technology</strong></td>
<td><strong>Technology</strong></td>
</tr>
<tr>
<td>Rigid, built for single purpose, proprietary</td>
<td>Network abstractions built for DevOps with modern technologies</td>
</tr>
</tbody>
</table>
Devops Virtual Cycle

Market Requirements

Operational Requirements

Service Requests

Development

Testing

Production

Network Operations

DEV

OPS

Create

Plan

Release

Configure

Verify

Package

Monitor

© 2018 Cisco and/or its affiliates. All rights reserved.
Devops Virtual Cycle (Today's Focus)

- Market Requirements
- Operational Requirements
- Service Requests
- Development
- Testing
- Production
- Network Operations

DEV
- Create
- Plan
- Package
- Verify
- Release
- Configure
- Monitor
- Network Operations

© 2018 Cisco and/or its affiliates. All rights reserved.
Transition Towards Automation

Network Engineer
“Automation”

Day-to-day management of rapidly growing, complex networks

Challenges
• Error-prone manual tasks
• Growing backlog
• Virtualization is coming

Service Developers
“Time-to-Market”

Develops new network services on demand

Challenges
• Implementation time
• Cost of change
• Lack of tooling

Ops and Provisioning Team
“Customer Experience”

Provisions services and manages service quality in networks

Challenges
• Lack of automation
• Quality issues in delivery
• No service insight

Service Abstraction
Leverage one central API for all services

Transformation
Develop your own services

Network API
Utilize a single interface to all network devices

© 2018 Cisco and/or its affiliates. All rights reserved.
Challenge Mapping

Implementation Time
• Informal specifications and manual steps
• Muddled boundaries between services and resources

Long time-to-market for new services
Challenge Mapping

**Implementation Time**
- Informal specifications and manual steps
- Muddled boundaries between services and resources

**Cost of change**
- Lack of in-flight changes and instance upgrades
- Mixed development and production environments

- Long time-to-market for new services
- Expensive lifecycle management
Challenge Mapping

Implementation Time
- Informal specifications and manual steps
- Muddled boundaries between services and resources

Cost of change
- Lack of in-flight changes and instance upgrades
- Mixed development and production environments

Lack of Tooling
- Mostly domain-specific technologies
- Centralized development setup

Long time-to-market for new services
Expensive lifecycle management
Excessive development costs
Quick System Overview

- Model-driven end-to-end service lifecycle and customer experience in focus
- Seamless integration with existing and future OSS/BSS environment
- Loosely-coupled and modular architecture leveraging open APIs and standard protocols
- Orchestration across multi-domain and multi-layer for centralized policy and services across entire network
Feature Mapping #1

Developing With Service Models
The Challenge

- We need:
  - A development approach focused on designing network services and implementing them on the network, automated end to end...
  - allowing for rapid service development meeting end-user needs and expectation

- On:
  - Brownfield networks
  - Across place in network, vendor, device type and protocol

Leverage the formality of a model-driven approach to significantly reduce the amount of coding needed
Model-based Architecture

- NSO assumes nothing about:
  - Network services
  - Network devices
- All data sets strictly defined by YANG models
- Tree-to-tree mapping reduces coding for lifecycle to absolute minimum
<table>
<thead>
<tr>
<th>Create</th>
<th>Update</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td>Challenging</td>
<td>Hard</td>
</tr>
<tr>
<td>Given a set of service-level inputs, provide a known and valid output to network</td>
<td>Allow arbitrary changes to the network service</td>
<td>Delete any given instance of a service and clean up the resources</td>
</tr>
<tr>
<td>May require some additional resource collection to fulfill the configuration set</td>
<td>May require collecting or handing back resources to fulfill configuration set</td>
<td>May require reference counting for shared resources</td>
</tr>
</tbody>
</table>

The ability to dry-run all operations is key for trust
High-level Development Process

**Model** the service structure and syntax using YANG and the related developer tools (validator, compiler) together with the users.

**Map** the relationship between the service model layer and the resource/device layer using templates and FASTMAP code (Python or Java).

**Test** the mapping by running CRUD operations on the service layer using dry-runs and the netsim environment.

Repeat
State Convergence: Create Service Instance

Service Layer

Create operation on service

Resource Layer

Create method to produce resource instance

Store the instance

Datastore
State Convergence: Delete Service

Service Layer

Delete operation on service

Resource Layer

Apply the reverse delta of the instance

Datastore

© 2018 Cisco and/or its affiliates. All rights reserved.
State Convergence: Modify Service

Arbitrary Modify operation on service

Service Layer

Run Create method as if new service

Resource Layer

Apply the diff and store as new

Perform tree diff between original and new, resulting in:

Datastore
Feature Mapping #2
The Service Model Lifecycle
The Challenge

• We need:
  • To be able to comfortably support the whole lifecycle of service definitions (create, update, retire)...
  • allowing service developers to quickly bring up services and iterate over them as requirements change.

• On:
  • Brownfield networks
  • Across place in network, vendor, device type and protocol

Manage service implementations as software packages with versioning, upgrade features and lifecycle tooling
The Package Manager

Well-defined management of packaged applications, including:

- Install, upgrade, uninstall
- Strict versioning
- Dependencies resolution
- Isolation
- Bundle management
- Distribution across clusters
The Role of Service Packages

• All user-defined code that needs to run in NSO is delivered as a package

• A package is basically a directory of files with a fixed file structure consisting of:
  • Package metadata
  • YANG modules
  • FASTMAP code

• Packages are versioned and runtime loaded by NSO
# Service Package Content

<table>
<thead>
<tr>
<th>Path</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;package-name&gt;/</td>
<td>The name of the package as reflected by the top-level directory name</td>
</tr>
<tr>
<td>./package-meta-data.xml</td>
<td>Package-related metadata including version, NSO version requirements, other packages required,</td>
</tr>
<tr>
<td>./src.yang</td>
<td>The YANG modules for the package</td>
</tr>
<tr>
<td>./src/java ./python ./templates</td>
<td>Code to manage operations on service instances</td>
</tr>
</tbody>
</table>

- Packages can also be delivered as tar-files or gzip-compressed tar-files
- Packages are built for specific versions of NSO
Putting it all Together

Much more about this topic including best practices available at the NSO Developer Hub
Feature Mapping #3

Development Tools
The Challenge

• We need:
  • A low barrier of entry for service developers to get productive developing service packages...
  • using well-known and robust technology choices along well-known best practice processes.

• On:
  • Brownfield networks
  • Across place in network, vendor, device type and protocol

Allow developers to plan for, create, dev-test and ship service packages from their local development environment
Developer Tools and SDK Content

Create

- Dev-local multi-vendor network simulator
- Full production-grade installation in dev environments
- YANG tools including validator, compiler
- Project tooling for managing package sets

Verify

- Dev-local multi-vendor network simulator
- Build- and runtime validation of package content
- Offline-tools for validating version migration

Package

- Self-contained and versioned package format
- Hitless package installation and version migration
- Local or remote project and package locations
The netsim Network Simulator

- A lightweight, developer-local network simulation framework
- Uses a combination of confd and NEDs to bring up and expose simulated devices
- Allows developers to continuously develop against reasonably realistic network equipment

$ ncs-netsim create-network cisco-ios 6 ios
The ncs-project tool

• Create new projects using the ncs-project create command
• Define what packages to use in the project-meta-data.xml file.
• Fetch packages with the ncs-project update command
  • Local files, git
• Export the project using the ncs-project export command

$ ncs-project create myproject
$ ...edit the project metadata to pull in packages...
$ cd myproject && make all
$ ncs-project export
Demo Time
Demo Flow

- **Prepare** a development environment including
  - NSO runtime project and NEDs
  - Netsim setup with IOS-xE and IOS-xR devices
- **Model** a simple MPLS VPN service
- **Map** service and NEDs using template
- **Test** the create-update-delete loop
Wrap-up and Q&A
Operators Voted
Cisco Leads Industry In Lifecycle Service Orchestration

LSO and OSS Leaders

<table>
<thead>
<tr>
<th>Company</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco</td>
<td>52%</td>
</tr>
<tr>
<td>Ciena (Blue Planet)</td>
<td>30%</td>
</tr>
<tr>
<td>Ericsson</td>
<td>28%</td>
</tr>
<tr>
<td>NEC/Netcracker</td>
<td>24%</td>
</tr>
<tr>
<td>Nokia</td>
<td>24%</td>
</tr>
<tr>
<td>Amdocs</td>
<td>20%</td>
</tr>
<tr>
<td>Accenture</td>
<td>16%</td>
</tr>
<tr>
<td>IBM</td>
<td>16%</td>
</tr>
<tr>
<td>Oracle</td>
<td>16%</td>
</tr>
<tr>
<td>HPE</td>
<td>14%</td>
</tr>
<tr>
<td>CENX</td>
<td>8%</td>
</tr>
<tr>
<td>UBIqube</td>
<td>8%</td>
</tr>
<tr>
<td>Other</td>
<td>4%</td>
</tr>
<tr>
<td>ADVA (Ensemble)</td>
<td>2%</td>
</tr>
<tr>
<td>Comptel</td>
<td>2%</td>
</tr>
</tbody>
</table>

OSS Vendors Deployed Today

<table>
<thead>
<tr>
<th>Company</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco</td>
<td>38%</td>
</tr>
<tr>
<td>Amdocs</td>
<td>24%</td>
</tr>
<tr>
<td>Other</td>
<td>22%</td>
</tr>
<tr>
<td>Ericsson</td>
<td>20%</td>
</tr>
<tr>
<td>HPE</td>
<td>18%</td>
</tr>
<tr>
<td>Nokia</td>
<td>18%</td>
</tr>
<tr>
<td>NEC/Netcracker</td>
<td>14%</td>
</tr>
<tr>
<td>Oracle</td>
<td>12%</td>
</tr>
<tr>
<td>Accenture</td>
<td>10%</td>
</tr>
<tr>
<td>IBM</td>
<td>10%</td>
</tr>
<tr>
<td>Comptel</td>
<td>4%</td>
</tr>
</tbody>
</table>

Source: SDxCentral 2017 Next-Gen OSS and the Rise of LSO Report
What You Gain with Cisco Network Services Orchestrator, Enabled by Tail-f

• Agility throughout service lifecycle
  - Strict YANG model-driven solution
  - Auto-rendered business logic results in 90% less code
  - Effortlessly re-deployment of updated service and device models
  - DevOps for differentiation

• Full automation

• Robust and proven in tier-1 deployments

• Industry’s broadest multivendor support

• Relevant in today’s and tomorrow’s networks
NSO DevNet – Key Highlights

The one place to use for sharing, finding and collaborating on NSO public knowledge!

- Light start through DevNet content page and Learning-Labs
- Constant news and updates to help you keep up to date
- Large searchable content pool
- Cisco customers, partners and employees all have access
- Got a question, ask! We will help ensure a fast response
- Easy to share and find public content
- Code sharing through public GitHub

developer.cisco.com/site/ns0
<table>
<thead>
<tr>
<th>Network Engineer</th>
<th>Ops and Provisioning Team</th>
<th>Service Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Automation&quot;</td>
<td>&quot;Customer Experience&quot;</td>
<td>&quot;Time-to-Market&quot;</td>
</tr>
<tr>
<td>Day-to-day management of rapidly growing, complex networks</td>
<td>Provisions services and manages service quality in networks</td>
<td>Develops new network services on demand</td>
</tr>
<tr>
<td><strong>Challenges</strong></td>
<td><strong>Challenges</strong></td>
<td><strong>Challenges</strong></td>
</tr>
<tr>
<td>• Error-prone manual tasks</td>
<td>• No service insight</td>
<td>• Implementation time</td>
</tr>
<tr>
<td>• Growing backlog</td>
<td>• Lack of automation</td>
<td>• Cost of change</td>
</tr>
<tr>
<td>• Virtualization is coming</td>
<td>• Quality issues in delivery</td>
<td>• Lack of tooling</td>
</tr>
</tbody>
</table>

**Dates:**
- January 10
- February 7
- March 7
Questions?