Remote PHY Device Automation

From Theory to Deployment

A Technical Paper prepared for SCTE by

Rick Morrison
Manager, Network Automation Systems
Rogers Communication
2400 32nd Avenue, Calgary, Alberta, T2E 6T4
403 750 4500
rick.morrison@rci.rogers.com

Table of Contents

Title **Page Number**

List of Figures

1. Introduction

As functionality moves away from the core, hubs are shifting from housing specialized RF networks to commodity fiber-based IP networks.

Trading RF for IP based networks simplifies the physical work and time required by a technician to setup and maintain the physical aspect. As a tradeoff, additional complexity is added in configuring and maintaining software configuration.

In this paper, we will discuss how we reduced complexity and eliminates repetitive, manual tasks by embracing an approach that automates the delivery and management of infrastructure. We will also present some essential steps and best practices that were used in managing the rollout of automation.

1.1. Motivation

We had two major drivers for automation The first was the ongoing need for network capacity growth which was not sustainable with the status-quo. The second was the need to improve node activity reliability.

1.2. Objectives of DAA

To understand why we picked the automation options we did; it is important to review the overall objectives we had of distributed access architecture (DAA) and how those influenced the approach.

1. Network Automation. DAA significantly simplifies our hub architecture allowing us the leverage to increasingly automate provisioning, turn up and streamline our overall process.

2. Multi-gig symmetric broadband services. A DAA architecture is required to deploy full duplex DOCSIS (FDX) or Extended Spectrum DOCSIS (ESD) and deliver multi-gigabit symmetric speeds over our HFC Network.

3. Facilities Improvement. DAA does not eliminate the need for us to scale and upgrade our facilities, however it reduces the scope and cost of these upgrades significantly.

4. Leveraging DAA for our future. DAA extends an IP network deep into the plant. This network can support multiple services for residential and commercial use-cases. In addition, DAA is a steppingstone to virtual cable modem termination system (vCMTS) and the edge cloud.

In the context of automation and considering the above objectives it was important to choose what to automate first. We chose to automate the process that introduced new R-PHY fiber nodes to begin with.

2. Approach

2.1. Methodology

Creating a synergistic and effective team is a shared aspiration, yet achieving this level of collaboration is often challenging. Our objective was to establish a working framework that would foster cooperation among network, software, and operational teams.

Within this framework, we set forth the following objectives:

1. Adopt continuous improvement methodologies by releasing regular improvements to deployment and operational tool sets.

2. Implement feedback systems from our previous production deployments, and from industry knowledge sharing, to continuously optimize/enhance future deployments.

3. Development of an architecture that supports a transition to next generation technologies such as vCMTS and FDX/ESD.

4. Automation-first philosophy in the development of our deployment processes. Leverage existing operational support system (OSS) investments to provide high-value automation opportunities. Engage with network teams to support identification of high value automation opportunities leveraging lean methodologies.[22]

Creating a single project team comprised of both software and network teams was crucial to fostering collaboration. This approach allowed teams to collaboratively analyze all facets of solutions, provide mutual support, acquire new proficiencies, and align priorities effectively.

2.2. Deployment Milestones

We adopted a strategy of "continuous improvement" and "iterative process" as we worked through our objectives. To make things manageable, we followed a common approach of "crawl, walk, run" where work was divided into smaller, "bite-sized" steps to achieve our goals.

Our main goals were split into distinct stages:

1. **Network lab trial**: We started by evaluating the various components of the network. This process ensured the interop between network components from a variety of vendors. This was a very manual, but necessary to understand what it is we were building including:

    - Understand how we were going to support frequent firmware upgrades that address frequent bug fixes and new features associated with a new technology.

    - Understand how we were going to troubleshoot and monitor for issues by using command line interfaces (CLI), element managers (EMS) and consolidated logging.

    - Identify required components and gaps for future automation.

    - Create sequence diagrams and models for inventory and orchestration based on the lab trial workflows.

2. **Minimal viable product (MVP):** This is the basic version of our solution that we can use for testing and ensure functionality. We validated that it could manage video, voice, and data services to proceed into customer trials.

    - First automation use case was to add an R-PHY fiber node to the back office and network.

3. **High touch trial (10 nodes):** This stage included a highly knowledgeable cross-functional team to support the trial. This team was also tasked with identifying process and preparing for the next stage of ramp up.

    - Learn to iterate functionality of monitoring & activation tools and processes.
    - Heavy focus on integrating technology with operational processes.
    - Introduction of automation.

4. **Operational ramp up**: Here we aimed to share knowledge across a broader group, including additional regions. The focus was on quality vs quantity and velocity of rolling out new nodes.

    - New fiber node activations without the direct support of development teams
    - Introduce an easy-to-use service order (SOM/SO) user interface for an improved user experience.

5. **Business as usual - Scale Deployment**: This phase involved scaling up to thousands of R-PHY fiber nodes. The goal was to target critical markets first, and then start to prepare our transition to next generation technologies under the DAA umbrella.

    - Introduce R-PHY fiber nodes that supported N x N configurations for future considerations including the use of 2x10GE links back to the CIN for additional capacity.
    - Update R-PHY models to align to standards.[2][3][4][5][6]
    - Tune automation and inventory systems to align with forward looking processes and architecture.

3. Delivery Considerations

In the realm of confidently delivering automation products iteratively, our attention was directed towards three key areas:

1. **Automating the release pipeline**: Our efforts were concentrated on streamlining the release pipeline through automation. This not only expedited the delivery process but also improved its reliability.

2. **Utilizing low-code techniques:** We explored the use of low-code techniques as part of our strategy. This approach aimed to strike a balance between efficiency and customization, enhancing our overall development.

3. **Emphasizing Cohesion and Coupling:** Recognizing the significance of a cohesive and well-coupled system architecture, we integrated this principle into our strategy to ensure robustness and maintainability.

An invaluable lesson we grasped early in the process was the dual importance of automated testing and an automated release pipeline. This concept of "automating the automation" significantly contributed to the confidence of a successful release, and efficiency of our release lifecycle.

3.1. Release Pipeline Automation

A release pipeline serves as a framework through which software transitions from development through to production. This process incorporates continuous integration and delivery (CI/CD) along with automated testing supporting frequent and confident software releases.

In the absence of this automation, the process of releasing software took us around three months. However, with the implementation of an automated release pipeline, this timeline was drastically reduced to 15 minutes. The integration of an automated pipeline also facilitated the support of concurrent development efforts and automated testing activities.

This was accomplished by:

1. **Automating builds and testing:** We automated the process of building software and conducting unit & mock testing. This streamlined approach expedited the development cycle while maintaining quality.

2. **Automating multiple environment**: We introduced automation through deploying software solutions in containers using Kubernetes, enabling the rapid creation of multiple testing and production environments. This approach enhanced efficiency and consistency in deployments.

3. **Decoupling solutions and domains**: We made a conscious effort to separate different solutions and domains, considering both distributed and monolithic architectures. This separation allowed for greater flexibility, scalability, and adaptability in our software development. Most importantly, this separation helped keep cognitive loading manageable and contained with the knowledgeable domain.

4. **Adopting cloud native and 12 factor App principles:** By embracing cloud-native practices and adhering to the twelve-factor app principles, we ensured that our software was designed to fully leverage the capabilities of cloud native and promote scalability, resilience, and maintainability.[7][40]

These efforts collectively exemplify the immense impact an automated release pipeline has on the software development cycle, not only in terms of speed and quality, but also in terms of enabling modern development practices that align with industry.

3.1.1. Automating Builds and Testing

The automation of the build process occupies a leading role within the CI/CD framework. These automated procedures form the initial sequences of steps intentionally designed to swiftly identify any discrepancies within code arising from recent modifications.

The following procedure describes the steps encompassed:

1. **Detailed ticket documentation:** A meticulous description of the task's objectives and rationale is documented. This documentation allows future traceability, and any team member to comprehend and fulfill the task's requirements effectively.

2. **Environment setup for building and testing:** An environment is prepared to accommodate both the building and testing processes.

3. **Code is written in version control and merged:** Code, along with unit and other tests, are written and related to task tickets. Select code is merged to a branch for automated testing**.**

4. **Automated code reviews and static analysis:** These are implemented through rules based static analysis tools and custom build unit & logic tests. Manual peer reviews are still a critical component of development, but can be decoupled from the automated process, or reduced.

5. **Compilation and building**: The code is compiled and built, transforming it into a fully functional, containerized software product.

6. **Automated documentation generation and publication**: Documentation is automatically generated and published wherever feasible.

7. **Artifact tagging and registry storage**: All other binary artifacts are tagged and stored within a registry for systematic organization.

8. **Deployment and testing of artifacts and configuration**: The containerized artifacts and configuration are deployed and subjected to comprehensive, automated testing.

9. **Deactivation of build and test environments**: The build and test environment are deactivated and automatically torn down.

10. **Team notification of outcomes**: The outcomes of the entire process are communicated to the team through notification services.

We also implemented API contract testing of all systems integrated into the automation solution by constructing scenarios involving successful and erroneous use cases, utilizing tools like RESTAssured [32] and Mockito[25]. In consideration of the "practical test pyramid" [39], we ensure that we adhere to both upstream and downstream contracts where possible. This approach reduces the necessity for resource-intensive, end-to-end testing, thereby optimizing testing efforts, while improving release confidence.

3.1.2. Automating Environment Turn-Up

This shift negates the need for developers to endure infrastructure delays or depend on the development progress of other systems.

By embracing versioning practices with environments, adhering to cloud native principles, and adopting declarative definitions, the outcome is infrastructure that is predictable and reliable. Notably, this infrastructure can be swiftly deployed within seconds, fostering and environment conducive to experimentation and innovation.

Over the span of a year, we transitioned from a substantial monolithic structure to a hybrid distributed structure that profoundly elevated our velocity and confidence.

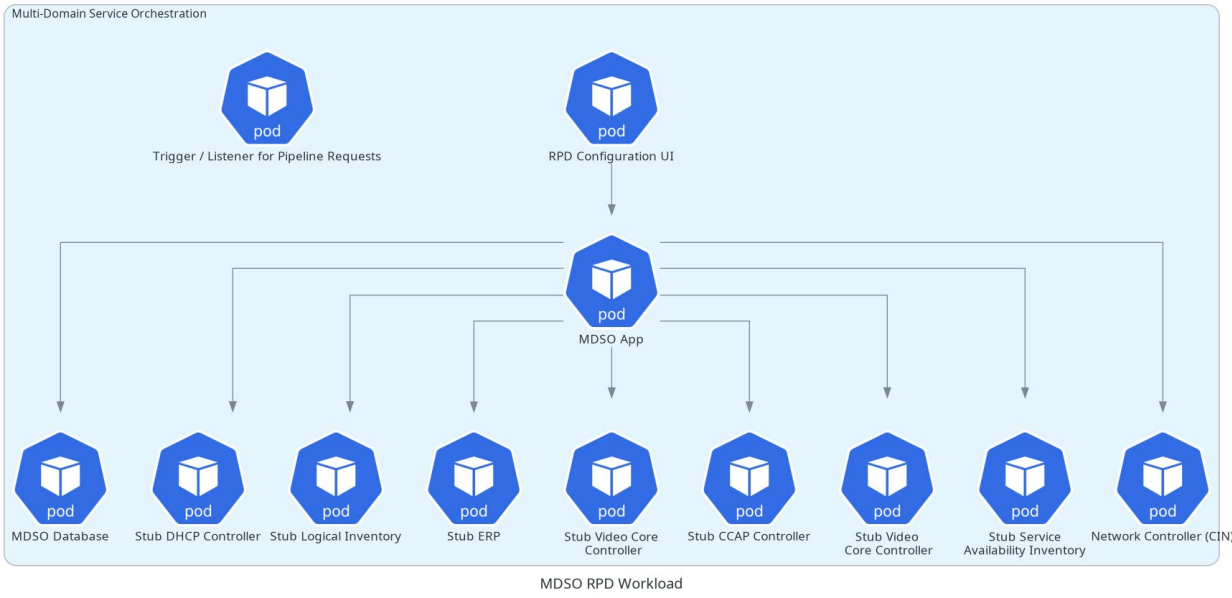

Figure 1: Multi Domain Service Orchestration (MDSO) Workload Instance

One or many instances (Figure 1) of a workload is meticulously brought into existence, tested, and systematically torn down each time a code merge or commit occurs. This parallel operational framework fosters the rapid simultaneous development of multiple features, ensuring a high rate of efficient development.

3.1. Low-Code Paradigm

In consideration of the lifecycle of an R-PHY fiber node, several low code strategies were adopted to accelerate development and simplify changes. [24]

Some of the low-code techniques we leveraged:

1. **Utilizing templates and models:** such as YAML (yet another markup language), YANG (yet another next generation) or HOT (heat orchestration templates) we were able to reduce the amount of code. [11][13][14][17]

2. **Auto-generated adapter code and tests:** derived from OpenAPI standards (e.g., REST swagger). [36]

3. **Industry standards:** benefited from constructs described from TMF638 including resource and customer facing services (RFS/CFS) and TOSCA.

4. **Declarative syntax:** The declarative approach defines the desired state of the system, including resources and properties infrastructure should have. In contrast an imperative approach defines a sequence of specific commands needed to achieve the desired configuration.

By adopting these techniques unnecessary and duplicate coding through layers is avoided.

3.2. Cohesion and Coupling

Cohesion and decoupling are two key concepts in software architecture, which serve to streamline intricate software components into more manageable work segments. This arrangement has several advantages:

- Reduction of cognitive load[9][37][35]
- Facilitation of parallel development through domain contracts and mock procedures
- Determination of optimal team sizes to maximize skill-based contributions.

To address the structing of tasks, work packages were subdivided into skill defined domains and explicit API (application interface) contracts were established between domains. This process defined two layers of architecture:

1. **The multi-domain orchestrator:** This orchestrator is responsible for the composition of all the services, coordinating resources declaratively defined in domains.

2. **The domain orchestrator(s):** Responsible for domain-specific orchestration, this layer operates through imperative sequences. It abstracts imperative actions into a declarative API which the multi-domain controller utilizes.

This arrangement (see Figure 2) ensures that domain-specific knowledge is concentrated within a domain orchestrator, alleviating cognitive burden within a specific domain. Qualifying cognitive load is intricate, underscoring the importance of garnering feedback from teams to calibrate the balance of complexity.

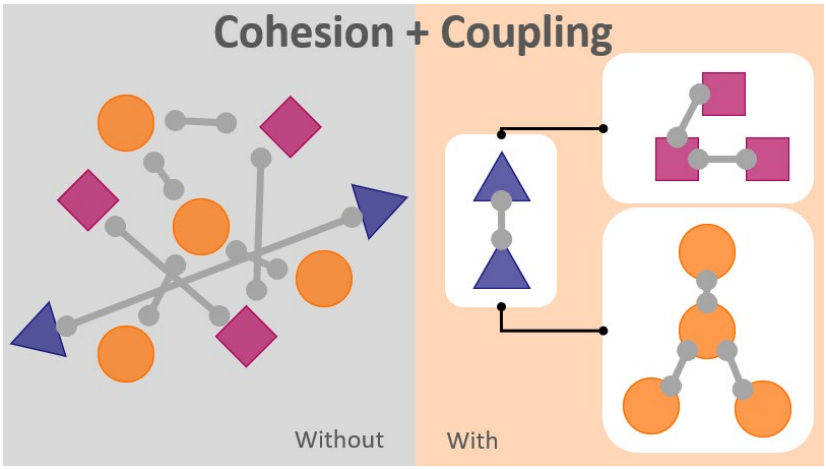

Figure 2: Cohesion and coupling

Decoupling is crucial because it simplifies the intricacies of the solution, rendering them easier to grasp and manage. On the other hand, coupling (or cohesion) serves to streamline the solution by aligning closely related elements together.

Key strategies for (de)coupling:

1. **Decoupling systems and domains for change management:** By disentangling systems and domains, the ability to enact changes without affecting external systems.

2. **Feature decoupling via API versioning:** API versioning allows for the separation of features, enabling modifications to specific components without impacting others until ready. This allows teams to develop at their own pace.

3. **Decoupling development through contracts and mock services:** Defining contracts and implementing "mock services" that simulate external systems or APIs enables teams to develop independently and at their own pace.

4. **System decoupling for resilience:** Decoupling systems ensures that if one system experiences an outage, the dependent system either generates an error or queues events.

5. **Selective coupling of closely related elements:** Closely related elements should be grouped together (ex: in a specific domain of orchestration), are strategically coupled to bolster confidence in real-time network provisioning.

Cohesion and coupling stand as fundamental attributes of our architecture that support velocity, confidence in delivery and reliability.

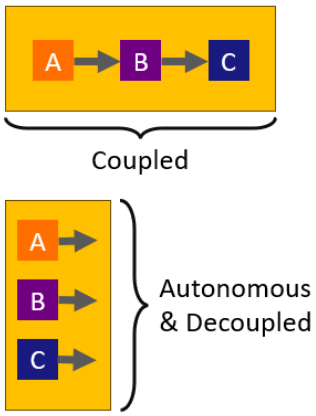

Figure 3: Decoupling teams' impact on schedule

In the realm of coupled or serial development, a sequential progression prevails (see Figure 3). This entails a situation where Team A's progress hinges on Team B's advancement, while simultaneously Team B is reliant on Team C's completion of tasks. This sequential reliance can result in notable delays in overall completion.

In contrast, the approach of autonomous development adopts a parallel trajectory. Here, Team A, Team B and Team C operate independently within their domain. This independence empowers teams to develop concurrently and reduces the aggregate time required to accomplish a project.

Autonomous development practices were instrumental in optimizing efficiency and expediting our success.

4. Process and Automation Architecture

The establishment of a process automation architecture is important to articulate a clear understanding of the business requirements and applications of the involved in automation efforts.

This section will describe these processes with focus on activations of R-PHY fiber nodes.

4.1. R-PHY Fiber node Processes

The journey of process improvement takes it first step by identifying existing business processes. Thought a detailed analysis of the operational landscape, inefficiencies and opportunities can be pinpointed. Armed with this insight, optimizing, and enhancing a business process unfolds.

Our goal was two-fold: Introduce a streamlined process alongside automation efforts.

At a high level we identified processes associated with introducing a new R-PHY node as follows:

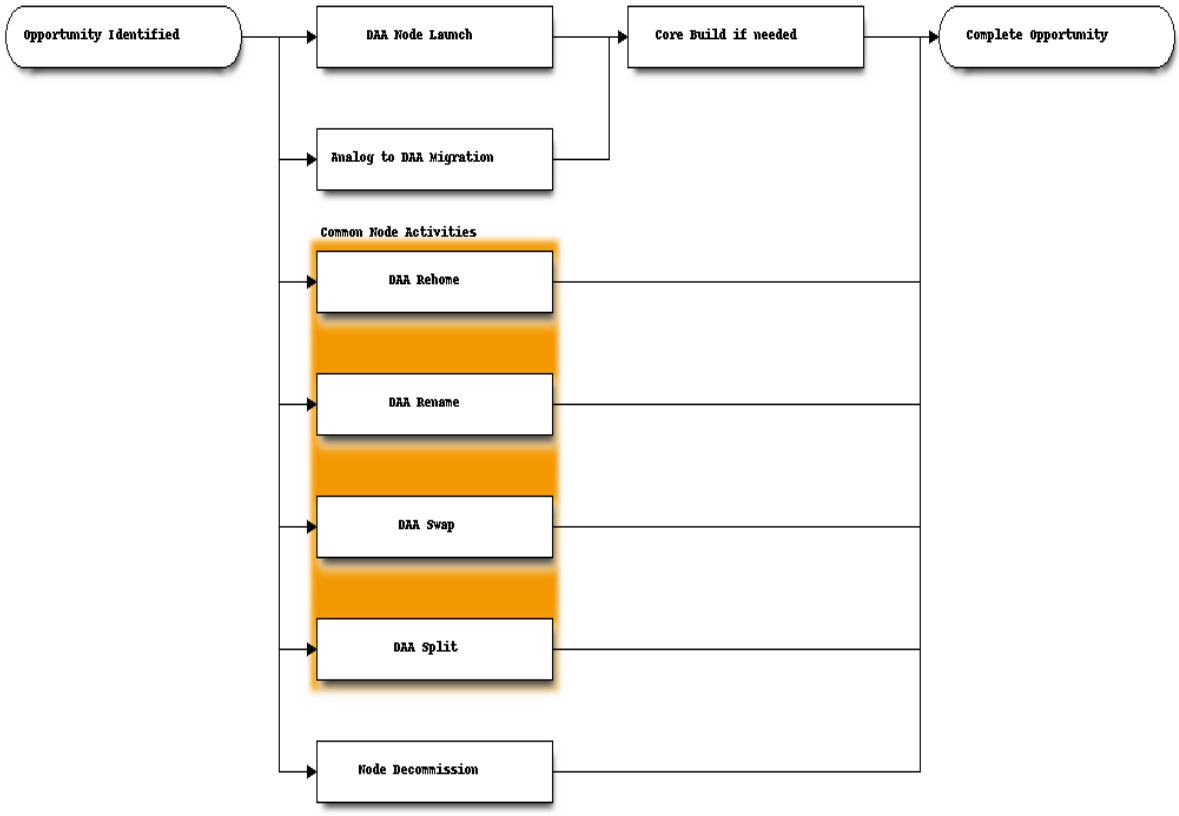

Figure 4:R-PHY fiber node processes

The crux of our automation endeavors was centered around the processes of adding & removing new R-PHY fiber nodes and the analog-to-DAA migration process.

This particular emphasis was attributed to our aspiration to scale up the deployment of R-PHY fiber noes within an established customer base, while upholding a sustainable pace.

The decommissioning aspect holds particular significance, as it supports our ability for regression testing and frequent updates, making it a pivotal use case to address.

4.1.1. Core Build Process

The focal point of the core build process centers on the deployment of the essential infrastructure (i.e., distributed converged cable access (D-CCAP) chassis and a video core) to accommodate DAA and R-PHY fiber nodes. [23]

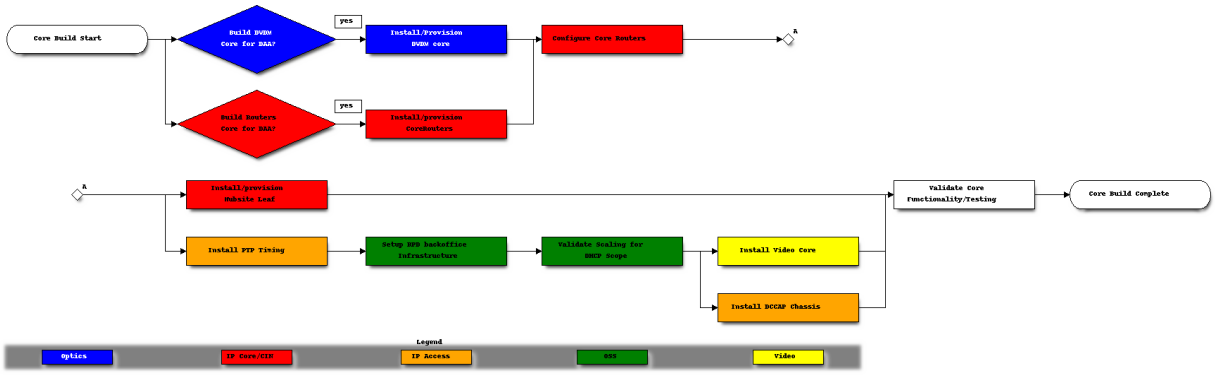

Figure 5: DAA Core Build Process

In our observation, the core build process (see Figure 5) was like the conventional build-out processes. The core build process is mainly physical cabling and physical documentation. As such this process was not a priority for automation.

4.1.2. Network Migration Strategy

R-PHY deployments are one step in a comprehensive plan (Figure 6) to further the evolution of the network. It is important to highlight during the sub split and mid split phases we are "preparing" the physical aspects of the plant to support R-PHY (see Figure 6).

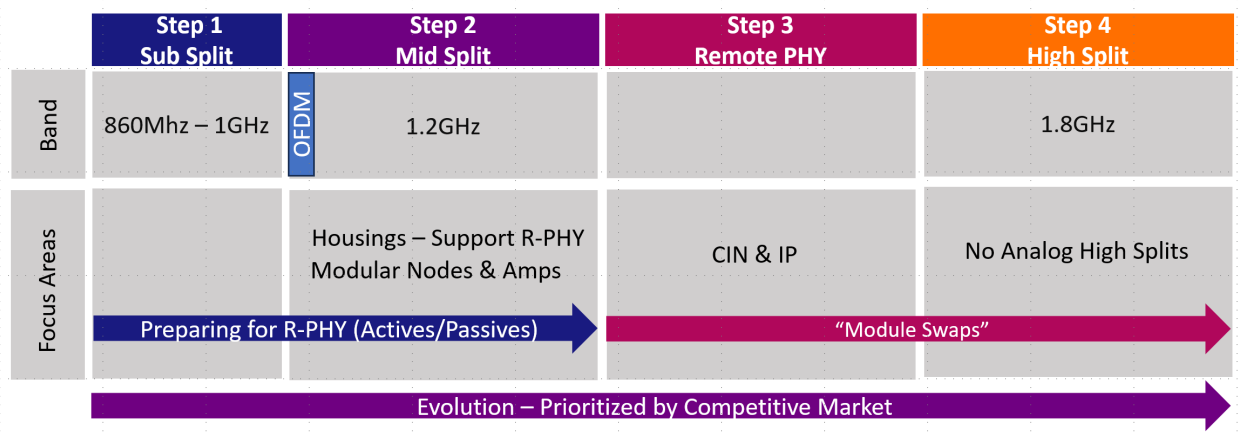

Figure 6:Network Migration Strategy

Our strategy has been to prepare housings, RF actives and passives to support bandwidth changes and modularize components where possible. This strategy has simplified deploying R-PHY by reducing the complexity when deploying a R-PHY node.

4.1.3. New R-PHY Fiber Node Launch Process

The "R-PHY Node Launch" and "Analog Conversion" processes (see Figure 7) emerged as the central contenders for automation priority. This choice was largely driven by the intricacies inherent to these processes and the desire to rollout DAA.

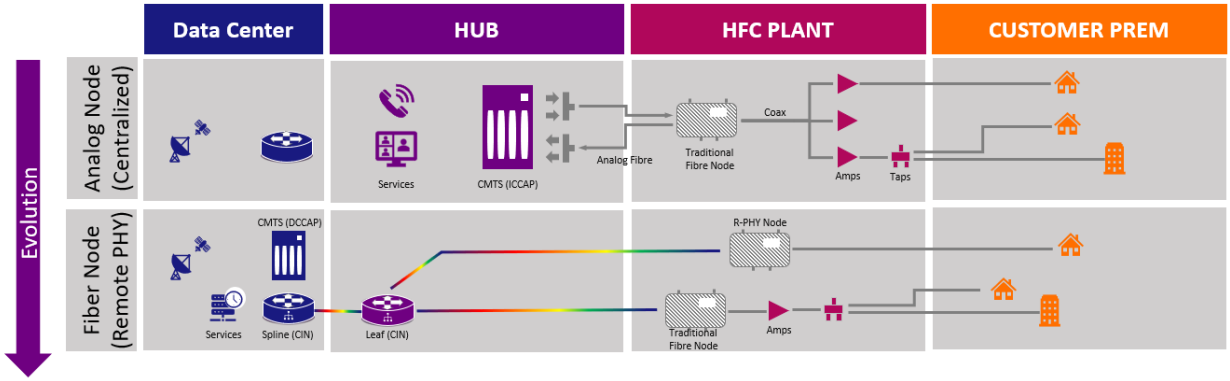

Figure 7: Analog & R-PHY network topology

A notable distinction between analog fiber nodes and R-PHY fiber nodes configuration is the substation of RF with IP technologies between the fiber node and CMTS. This entails replacing tangible physical construction with software-based configuration procedures.[2]

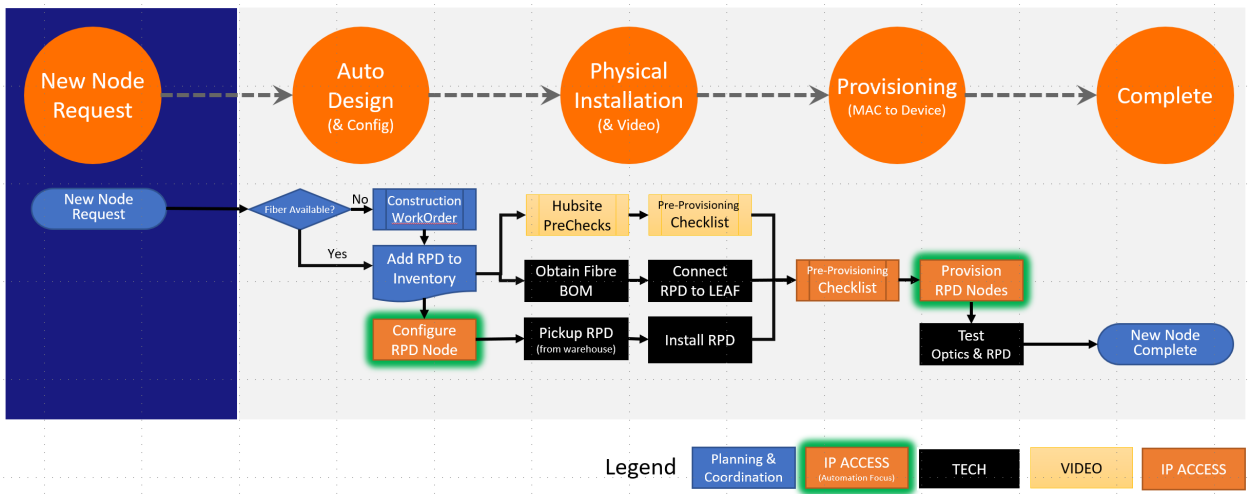

Figure 8: New R-PHY node process

At a broader perspective, a new R-PHY fiber node request (see Figure 8) entails an activity which triggers documentation of the fiber node within spatial inventory. This documentation, combined with current state inventory serves as the foundation for guiding network technicians in the installation of cables, and automating the design and configuration of the fiber node.

Upon the completion of the physical aspects of the installation, the activation process commences by associating the MAC (media access control) address of the R-PHY node with the video core and CMTS (cable modem termination system).

The entire process from beginning to end can span anywhere from a couple of days to a month depending on the complexity of the physical work.

The two automated tasks (configure and provision) replace approximately eight hundred lines of configuration with a handful of attributes per R-PHY device (device name, hub site & mac address or serial number of the R-PHY device).

Future automation will include pre-provisioning and the supplementary validation. Our judicious approach dictates that continue improvements be sequenced in accordance with diminishing returns of value, while simultaneously upholding the overarching architectural objectives.

4.2. Automation Architecture

Upon examination of the "new node process," the most substantial value derived from automation would be and was realized through the automation of the auto-configuration and auto-provisioning tasks. These two tasks would also be accelerated by leveraging prior OSS modernization investments.

As part of our strategy, we opted to continue utilizing our existing tools for monitoring and node segmentation tasks, as these required minimal or no alteration at this time.

Domain orchestrators and the multi-domain service orchestrator (MDSO) are as stateless as possible, with inventory systems being entrusted with resource state information.

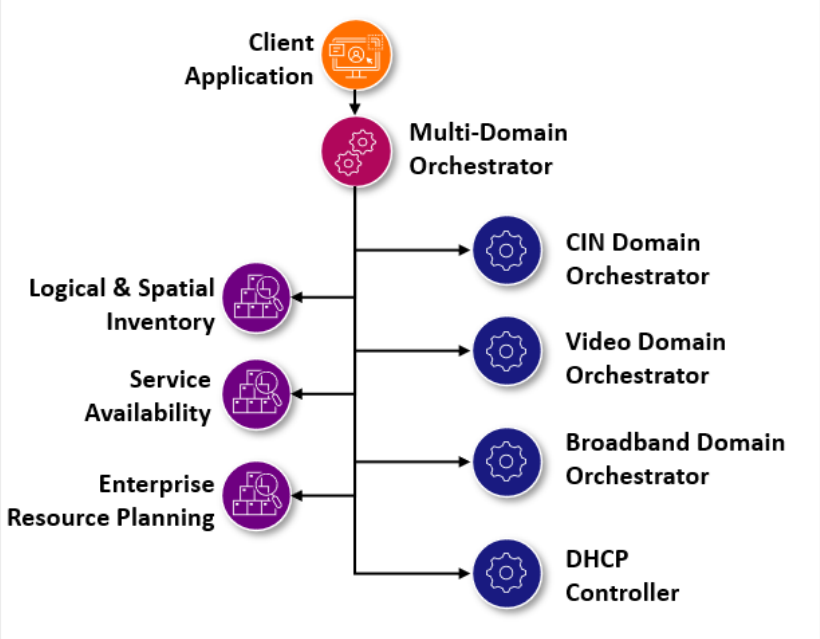

Figure 9: Orchestration System Interactions

This architecture (see Figure 9) is function based, and therefore tool agnostic. Some of the platforms used are:

1. Client Application - Vaadin, Quarkus running on Openshift.[41] [31]

2. Multi-Domain Orchestrator – HPE Service Director running on Openshift and deployed with ArgoCD and Tekton. [12][31][1][38]

3. CIN Domain Orchestrator – Cisco Network Service Orchestrator[28]

4. DHCP Controller – customer interface into Cisco CNR[8]

5. Broadband Domain Orchestrator – custom interface running on Kubernetes[21]

6. Logical & Spatial Inventory – Netcracker RI and SpatialNet[26][34]

7. End to End Testing – SOAPUI[33]

A minimalistic client application (see Figure 10) serves as a conduit for initiating either configuration or provisioning activities.

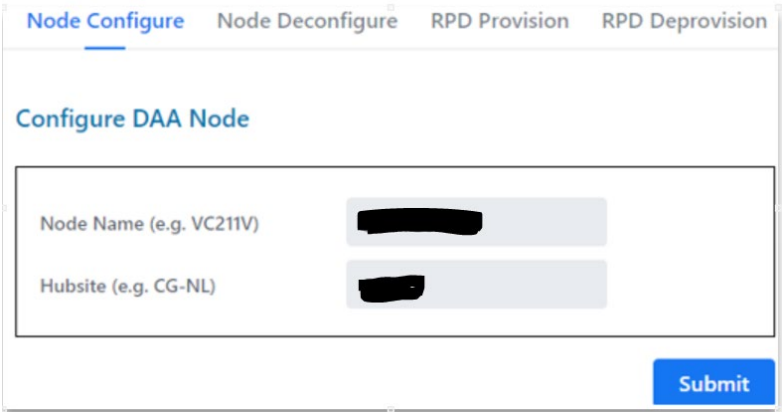

Figure 10: Auto configuration UI

The "configure node" function solicits for the new node name, and hub site location. These details facilitate the retrieval of spatial, RF spectrum and logical inventory data and automation of design and configuration tasks via the MDSO.

The "provision node" function seeks input of the new node name and either serial number or MAC address. These details activate the RPD by associating the MAC address to relevant equipment.

The multi-domain orchestrator functions as a vigilant overseer, tracking errors and the status of each domain function through the orchestration process.

4.2.1. Auto Configuration Orchestration

The R-PHY fiber node configuration will configure the node and make it available for the R-PHY fiber node device to provision. This involves creating the circuit between node and the CIN device followed by assigning the node to the CIN port and assigning the CCAP device to the R-PHY fiber node. After the end of this process, the CIN port assignment status will be set to 'Reserved' and the node status will be set to 'Planned' in inventory.

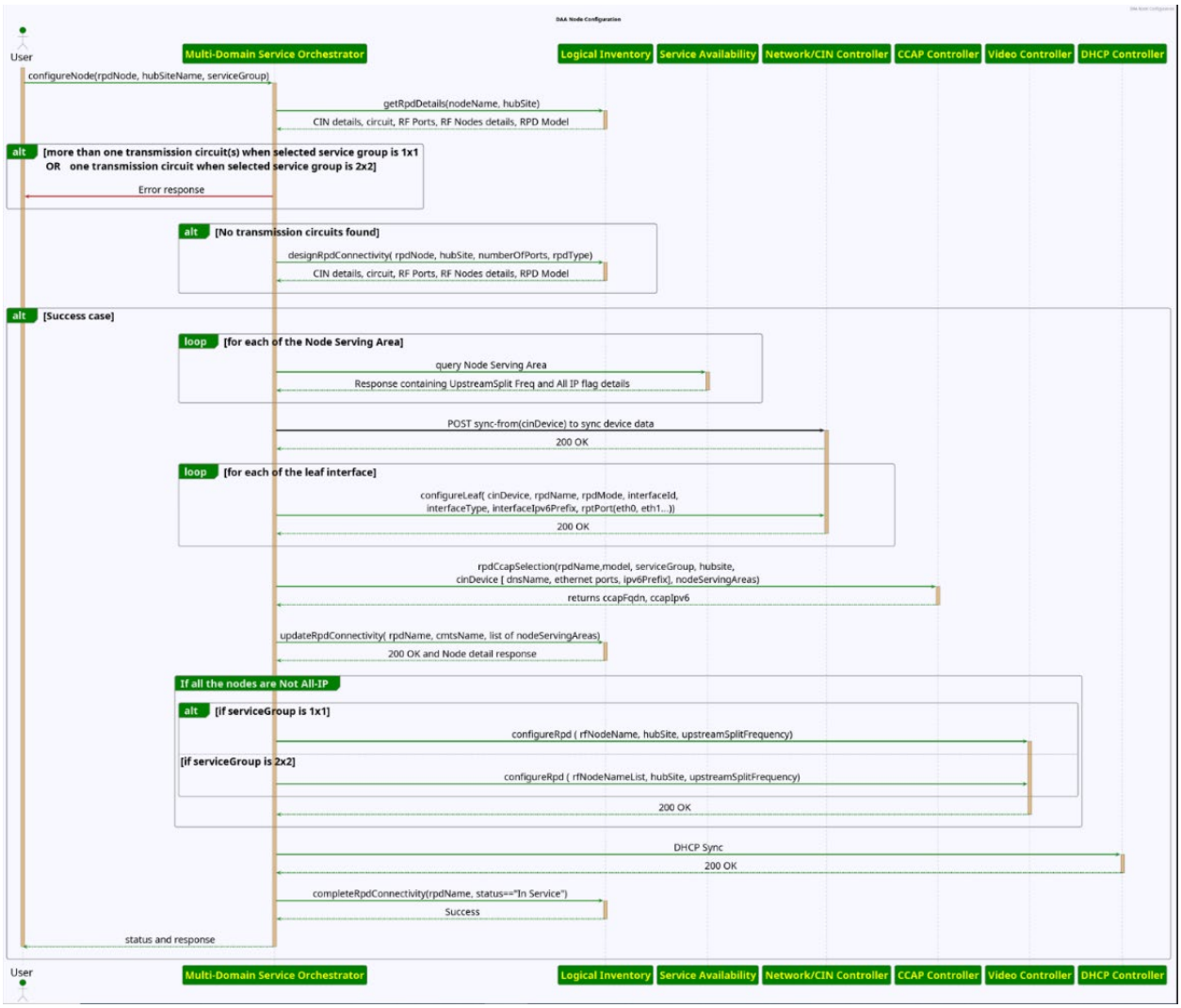

Figure 11: Autoconfiguration Sequence

This task replaces the manual effort (see Figure 11)required to:

1. Create the node in our logical inventory system.
2. Add the node to video on demand.

3. Configures the leaf port with ipv6, midv2, multicast and DHCP.

4. Reserves a cable-mac on the CMTS for the R-PHY fiber node.

5. Updates the DHCP server.

This task enables field technicians to start and complete their work to connect fiber cables to the CIN and install the physical R-PHY fiber node device.

4.2.2. Provisioning Orchestration

Provisioning is the last task before the R-PHY fiber node can be activated. This automation associates the MAC address of the R-PHY fiber node with the network.

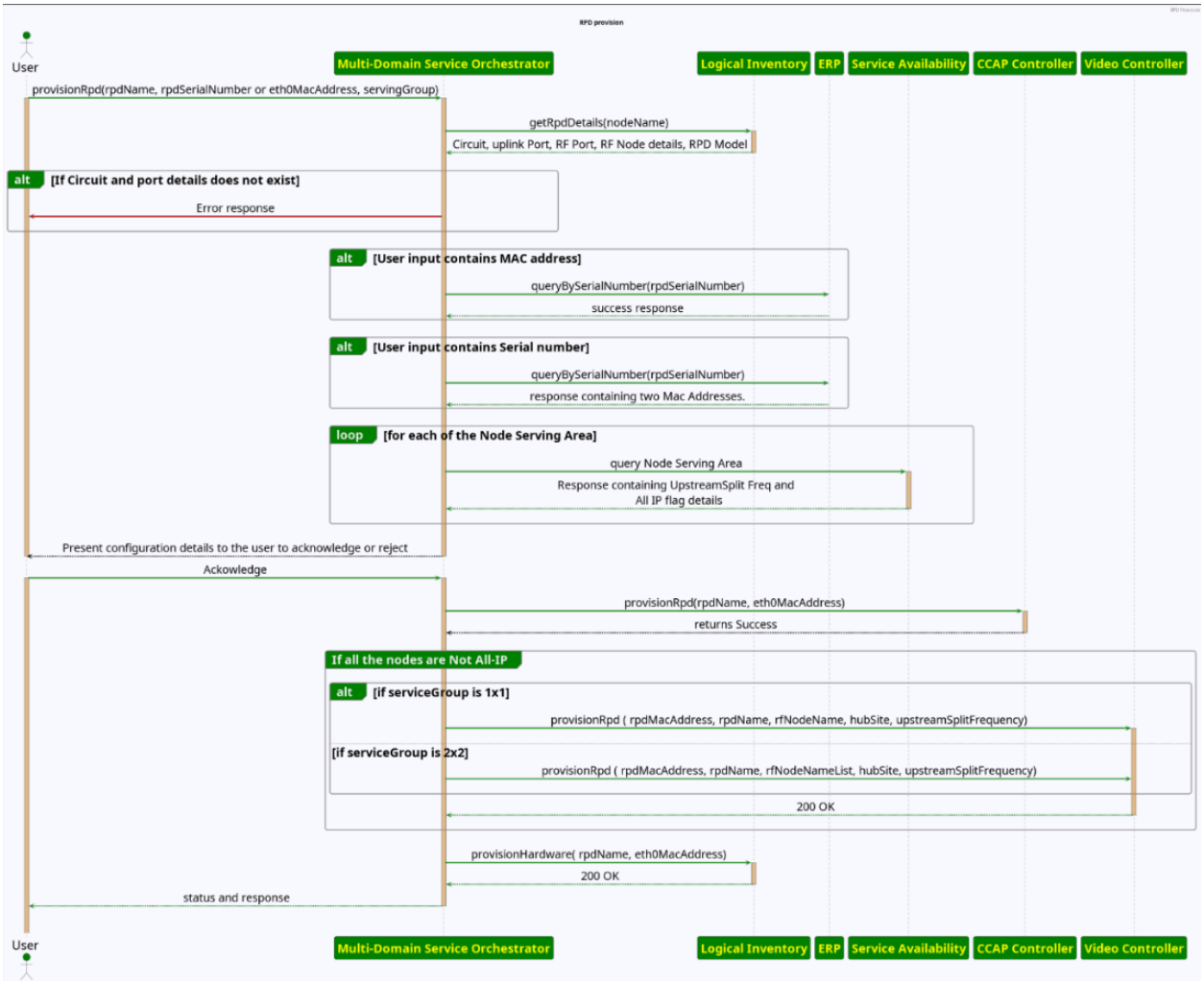

Figure 12: Provisioning sequence

This task (see Figure 12) replaces the manual workflow for the following:

1. Get MAC Address of R-PHY fiber node from serial number (SN) located in enterprise resource planning (ERP) (used for video core).
2. Get Circuit and R-PHY fiber node details from logical inventory.
3. Get video service availability details (all IP or legacy).
4. Provisions the R-PHY fiber node on the CCAP Chassis.
5. Provisions the video core.
6. Updates Logical Inventory with MAC to R-PHY fiber node relationship & label it as provisioned.

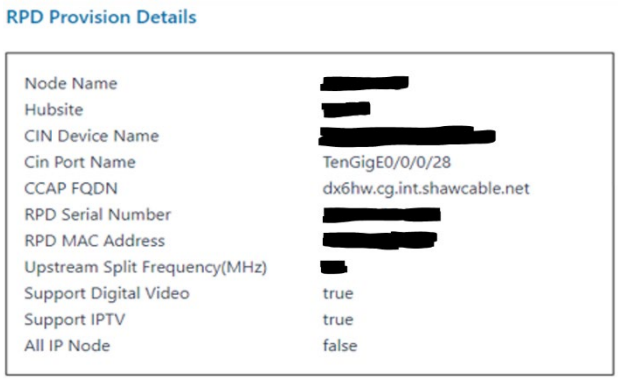

Figure 13:Output of Provision

The above output (see Figure 13) describes key variables and inventory information used to automate fiber node provisioning.

5. Results

5.1. Scaling R-PHY Fiber Node Additions Over Time

Across the past three years, a noteworthy milestone has been that the inclusion of 1000 new R-PHY devices was achieved. This progressive achievement has realized at a monthly rate spanning from 20 to 50 R-PHY nodes per month.

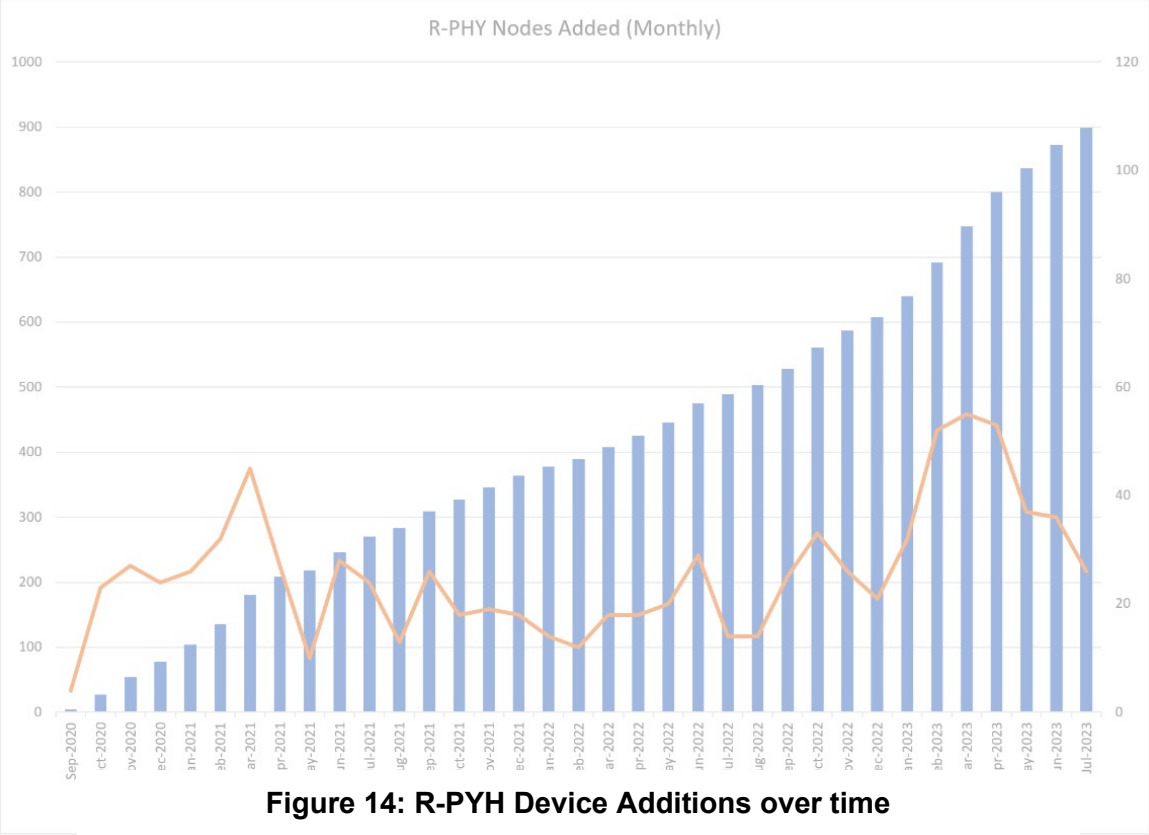

Figure 14: R-PYH Device Additions over time

5.2. Accelerated Node Activation Time

A remarkable transformation has unfolded in terms of the time taken for R-PHY node activation. The following chart describes the comparison of deploying a R-PHY node with automation, estimated R-PHY node without automation, and a traditional analog fiber node.

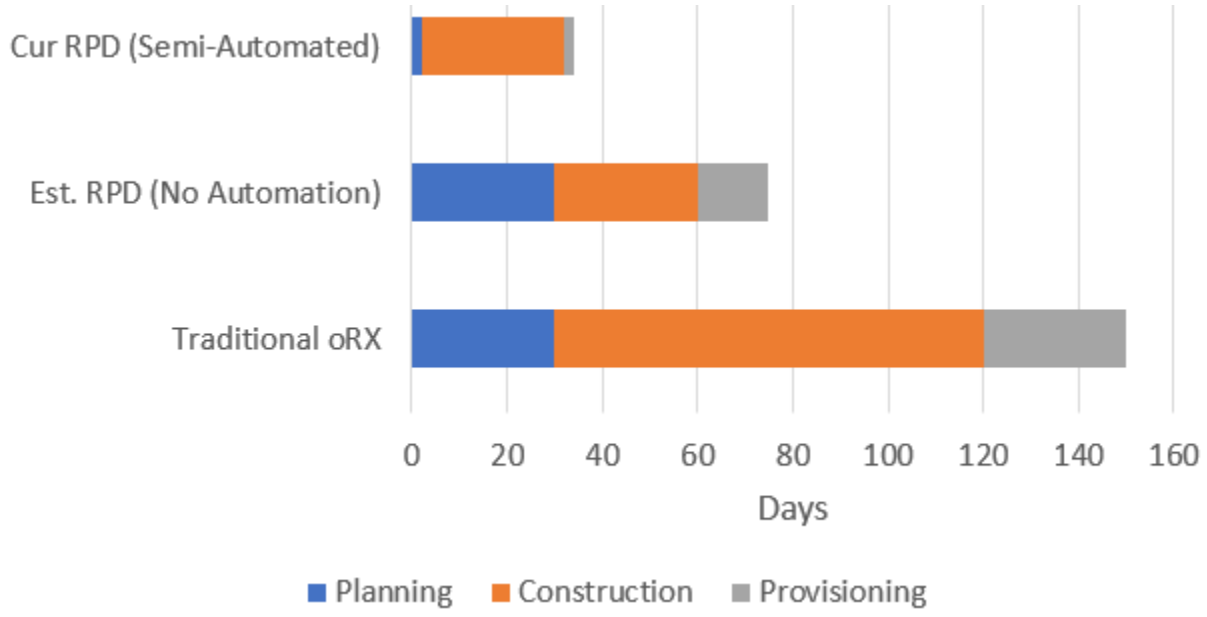

Figure 15:Time to Activate a Node

Presently, the automated "new R-PHY fiber node" can conclude within a matter of days, contingent on the intricacies of construction.

5.3. Errors in Auto-Configuration & Provisioning

Initial stages witnessed a notable frequency of transactional errors, surpassing the tally of successful instances. This trend primarily stemmed from delayed, absent, or inaccurate logical and spatial inventory. In response several critical lessons emerged:

1. Shifting from traditional analog fiber nodes, automation necessitates documentation ahead of installation. This transition ensures inventory data is preemptively available for automation.

2. Delays stemming from inventory systems, taking up to 24 hours for updates, prompted a realization that provisioning must transition to a near-real-time paradigm, warranting the updating of back-office systems.

3. A minor number of errors were attributed to unique use-cases. Continued focus on resolving these through continual improvement ensures stability.

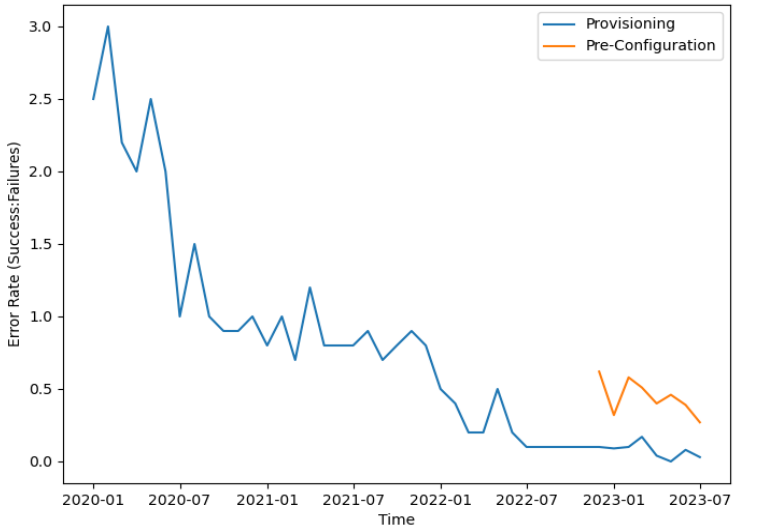

Figure 16:Error Rate

Subsequent efforts have led to a significant reduction in errors, a trajectory we endeavor to sustain while simultaneously introducing new features within the solution.

5.4. Complexity and Level of Effort

It is difficult to imagine the magnitude of effort required to maintain, support, and expand the DAA solution. To appreciate this, we undertake a comparative analysis between the magnitude of configuration lines on managed devices and the manual documentation and task-team requirements for deploying new R-PHY fiber nodes.

Through orchestration, dedication to inventory management, process mapping, and domain automation we have a highly sustainable process. In lieu of managing approximately 800+ lines of configuration within various systems, we have distilled the process down to a mere handful of parameters.

This optimization, achieved through streamlined processes and automation has resulted in a notable reduction in the effort required for operational continuity. This has been observed by reducing the amount of team required to launch a new node from approximately 25 to 7 and the number of manual tasks from approximately 70 to 20.

6. Reflections

During each iteration feedback was obtained from the program and documented in retrospectives & lessons learned. Retrospectives can be categorized under the following themes:

6.1. Positive Collaboration and Alignment Feedback

The orchestration of R-PHY devices engaged a multitude of teams: from planning, project and fiber coordination to IP teams, OSS, network technicians, video and more. This intricate ensemble necessitated a robust focus on collaboration.

Key elements encompassed fostering trust, defining roles and responsibilities, and cultivating an environment where the coalescence of efforts led to successful deployments, swift problem resolution, and a sense of meaningful contribution.

Vision and goal setting played a pivotal role by establishing the trajectory and minimizing tangential pursuits. Steering and working committees further facilitated efficient prioritization, enhancing alignment, and maintaining project trajectory.

6.2. Continuous Improvement and Feedback Culture

The culture of continuous improvement, underpinned by a keen responsiveness to feedback, resulted in the delivery of incremental successes, heightened agility, and augmented confidence in outcomes.

The iterative cycles and milestones offered scrutiny of each phase's insights and celebrations of collective accomplishments.

6.3. Adherence to Industry Standards and Models

Leveraging industry standards and models proved instrumental in expediting delivery and integration endeavors. However, lessons emerged from the choice of "custom models" due to the maturity of some models at the time resulted in additional effort down the road. In retrospect, embracing and maturing emerging standards may facilitated a smoother progression.

Additionally, the pursuit of transactional automation underscored the need to overcome the limitations of CLI or SNMP, promoting the evolution of the OSSI model [30] to enhance orchestration in the DOCSIS technology sphere.

6.4. Influence of Conway's Law

The resonance of Conway's Law, "the design of a system reflects the communication structure of the organization," often subtly unfolds in project dynamics.

Initial milestones emphasized technology and process, deferring user experience considerations to subsequent iterations which resulted in avoiding negative consequences of Conway's Law.[10]

6.5. Reliability of Automation Systems

As software systems encompass more systems and integrations, the probability of low availability increases. This concern held sway during the initial iterations, fueling proactive measures.

Optimizing start-up and recovery times, coupling & cohesion strategies and scrutinizing individual transactions errors emerged as successful strategies to mitigating this concern.

It is crucial to control the number of dependent systems in orchestration chains to maintain reliability expectations.

6.6. Foundations and Essentials

Initial stages underscored the significance of comprehending the "DAA Solution" before automating it. Having a working network solution in place before embarking on any automation proved invaluable. This approach furnished rapid support for DAA initiatives, magnifying the importance of an established groundwork.

Early architectural insights played a pivotal role by identifying gaps, API contracts, and sequence diagrams, culminating in expedited implementation.

7. Concluding Insights

In culmination, the collaborative efforts of diverse teams have positioned us well to support next generation technologies and services. Automation has emerged as our ally, propelling the swiftness and consistency of R-PHY fiber node additions to new heights while ensuring unwavering predictability and reliability.

However, as automation becomes more widespread, it is also becoming apparent that the technology that enables it is in many ways the easiest part.

An effective automation initiative is overwhelmingly based on people—including culture, process, capabilities, and skill sets.

Abbreviations

API	application programming interface
CCAP	converged cable access platform
CD	continuous delivery (not deployment)
CI	continuous integration
CIN	converged interconnect network
CLI	command line interface
CMTS	cable modem termination system
COTS	commercial off the shelf
DAA	distributed access architecture
D-CCAP	distributed converged cable access platform
DOCSIS	data over cable service interface specification
EMS	element management system
ESD	extended spectrum DOCSIS
FDX	full duplex
Heat	cloud formation declarative based template from OpenStack
HFC	hybrid fiber / coax network
HOT	heat orchestration templates
I-CCAP	integrated converged cable access platform
IP	internet protocol
MAC	media access control
MIB	management information base
MOP	method of procedure
NED	Network element driver
NETCONF	network configuration protocol
NOC	network operations center
OMS	order management system
ORX	optical receiver
OSS	operational support system
PHY	physical layer
PTP	precision time protocol
REST	representational state transfer
RESTCONF	restful network configuration protocol
RF	radio frequency
R-PHY fiber node	remote physical layer device
SNMP	simple network management protocol
SOAP	simple object access protocol
SOM	service order management
SSH	secure shell
SSL	secure sockets layer
TLS	transport layer security
TOSCA	topology and orchestration specification for cloud applications
VOD	video on demand
YAML	yet another markup language
YANG	yet another next generation

Bibliography & References

[1] Argo CD (Argo CD | Argo (argoproj.github.io))

[2] CableLabs DOCSIS DCA-MHAv2, Remote PHY Specification, CM-SP-R-PHY-I12-190307,

[3] CableLabs Flexible MAC Architecture System Specification, CM-SP-FMA-SYS-I03-220126.

[4] CableLabs FMA MAC Manager Interface Specification, CM-SP-FMA-MMI-I03-220126.

[5] CableLabs FMA OSS Interface Specification, CM-SP-FMA-OSSI-I02-220602

[6] CableLabs R-PHY Node Model (http://mibs.cablelabs.com/YANG/DOCSIS/R-PHY/)

[7] Cloud Native (www.cncf.io)

[8] Cisco CNR DHCP (Cisco Network Registrar - Cisco)

[9] Cognitive Load Theory (The Evolution of Cognitive Load Theory and the Measurement of Its Intrinsic, Extraneous and Germane Loads: A Review (tudublin.ie))

[10] Conway's Law on Organizational Structure (Conway's law - Wikipedia)

[11] Heat Orchestration Template (HOT) specification (Heat Orchestration Template (HOT))

[12] HPE Service Director (Telecom Operations, Network Automation, HPE)

[13] IETF RFC 6020, YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), October 2010.

[14] IETF RFC 7950, The YANG 1.1 Data Modeling Language, August 2016.

[15] IETF RFC 7951, JSON Encoding of Data Modeled with YANG, August 2016.

[16] IETF RFC 7952, Defining and Using Metadata with YANG, August 2016.

[17] IETF RFC 8040, RESTCONF Protocol, January 2017.

[18] IETF RFC 9195, A File Format for YANG Instance Data, February 2022.

[19] ITU-T, "Precision time protocol telecom profile for frequency synchronization", ITU-T Recommendation. 8265.1, July 2014.

[20] Jenkins Pipeline Automation (Jenkins)

[21] Kubernetes (Kubernetes)

[22] Lean Methodologies (Lean - Wikipedia)

[23] Legacy & Video Core Automation (Paper - Leveraging Legacy Video in Digital Access Architecture Networks - NCTA Technical Papers)

[24] Low Code Methods (What is Low-Code? | IBM)

[25] Mockito Framework (Mockito framework site)

[26] Netcracker Resource Inventory (Netcracker - Home)

[27] Node Provisioning and Management in DAA (Paper - Node Provisioning and Management in DAA - NCTA Technical Papers)

[28] NSO Network Service Orchestrator (Cisco Network Services Orchestrator - Cisco)

[29] OpenAPI Initiative (Home - OpenAPI Initiative (openapis.org))

[30] OSSI CableLabs (Specifications Search – CableLabs))

[31] RedHat Openshift (Red Hat OpenShift Container Platform)

[32] RESTAssured Test Services (REST Assured (rest-assured.io))

[33] SOAPUI Test Suite (ReadyAPI | SmartBear Software)

[34] SpatialNet (Synchronoss - Operator Branded Cloud and Messaging Experiences)

[35] Software Engineering, cognitive load (Dave Farley's Weblog | Thoughts on Continuous Delivery and Agile development.)

[36] Swagger Specification (API Documentation & Design Tools for Teams | Swagger)

[37] Team Topologies Regarding cognitive loading (Team Topologies and Team Cognitive Load Assessment — Team Topologies)

[38] Tekton CI/CD (Tekton)

[39] Test Pyramid: The Practical Test Pyramid (martinfowler.com))

[40] Twelve Factor App (The Twelve-Factor App (12factor.net))

[41] Vaadin UI (Vaadin | Discover the Web App Framework Built for Java)

