“自動化やってみた”のその先へ
～KDDIネットワーク検証の自動化、現場エンジニアの奮闘と学び～

KDDI

2025年8月1日

Tomorrow, Together
KDDI

自己紹介

氏名	松久　瑛洋
所属部署	ソリューション技術運用本部 ソリューションネットワーク技術部
担当業務	国際バックボーンNW開発 検証自動化
JANOG 参加歴	JANOG54, JANOG55 初登壇(緊張)

氏名	中川　亮平
所属部署	ソリューション技術運用本部 ソリューションネットワーク技術部
担当業務	国内法人NW開発 検証自動化
JANOG 参加歴	JANOG55 初登壇(すごい緊張)

アジェンダ

1. はじめに
2. 背景・プロジェクト発足
3. 実現手法について
4. 初期開発対象について
5. 検証自動化システム開発
6. 自動化してみて
7. 今後の展望

はじめに

Executive Summary

01 Background

◆ **なぜ**自動化が上手くいかないのか
自動化が進まない原因を色々な角度から分析

02 Solution

◆ 現場の開発エンジニアが自らの**あるべき姿**を徹底的に議論
◆ 内製開発によるシステム構築に挑戦

03 Difficulties

◆ NWエンジニアが自動化について**よくわからない**
◆ プログラマがNWについて**よくわからない**

04 Key to Success

◆ NWエンジニアに寄り添って開発を進め、**前のめり**で開発に参加してもらう
◆ NWエンジニアから**マルチスキルエンジニア**へ

【参考】検証自動化に関するKDDIの取り組み＆棲み分け

■ 短期的な課題解決後に中長期的にあるべき姿へと移行を図っていく。

※当社ソリューションネットワーク技術部の場合

	本発表 本発表での取り組み	JANOG54 OpenConfigを活用した NW検証業務自動化への取り組みと課題
仕組み	CLIベースの自動化	OpenConfig(共通データモデル) を活用した自動化
ベンダ依存性	各OS毎にシナリオ記述が必要	単一シナリオを複数OSで活用可能
対応OS	制限なし	OpenConfigへの対応にOS差分あり
時間軸	**短中期**	中長期

中長期的に移行

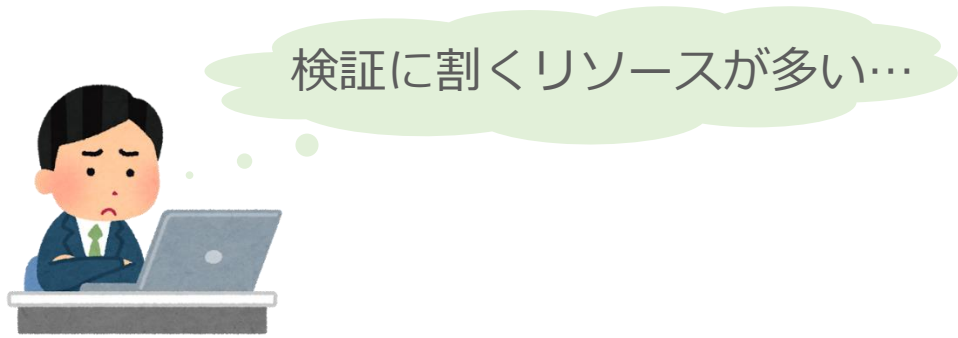
背景とプロジェクト発足まで

KDDIの検証業務について

■ KDDIは大規模ネットワークを運用しており、
マルチベンダ・異なるOSのネットワーク機器が多数存在している。
ソリューションネットワーク技術部では
法人向けの固定系ネットワークの設計開発
を主管している。

■ 近年は、脆弱性対応のOS更新が増加傾向。
検証実施～商用適用までの**リードタイムが直接セキュリティリスク**となるため
正確かつ迅速に検証を実施することが求められる

当部では、
約**80件/年**の検証を実施
約**320人月/年**の工数をかけて検証を実施している
※外部委託している案件も有り



検証自動化の目的

■ 我々が自動化によって期待している効果

検証期間短縮

- 就業時間に縛られずに検証可。
- 手動の場合と比較して短納期でリリース可能

（外部委託）コスト削減
（自前実施）検証稼働削減

- 1clickで実施可能なので外部に委託する必要無
- 自前実施の場合も機器オペレーション不要

検証品質向上

- 性能試験の場合、より詳細な限界性能を把握できる
- 自動検証なので ヒューマンエラーも無い

自動化が浸透しない原因について

自動化が浸透しない orz…

多忙

手動で検証してるので余計時間がかかる
「自動化＝空き時間で行う」という認識
各々の担務に「自動化」が含まれていないため

→ **正式なプロジェクト化** によって、
「自動化」がメンバの担務になり、
優先度を高くして対応可能に

個人的に自動化しても広がらない

担当設備に特化した設計となってしまう。
他グループの業務を知らないので使えるか不明

→ **部内全グループからメンバが参加** することで、
担当設備の検証を見据えて開発に参加できる

やり方が分からない

ネットワークエンジニア主体の部署であり、
プログラミング等のスキルセットを
持ち合わせていないメンバもいる

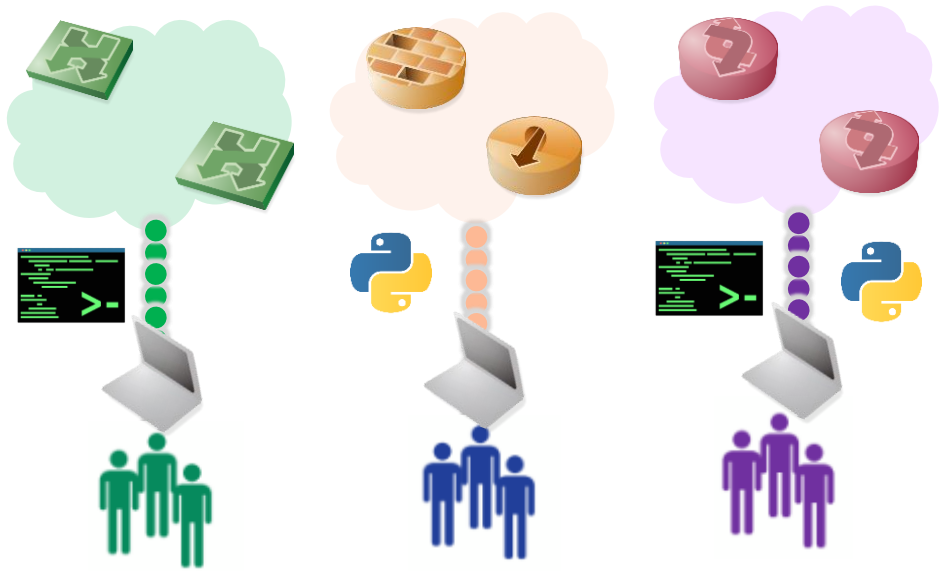
→ **有識者参加** によって
ナレッジを共有しあうことで、
お互いに高めあっていくことを目指す

目指す姿

■ 検証自動化PFを構築し、**24時間365日止まらない検証**を実現する。

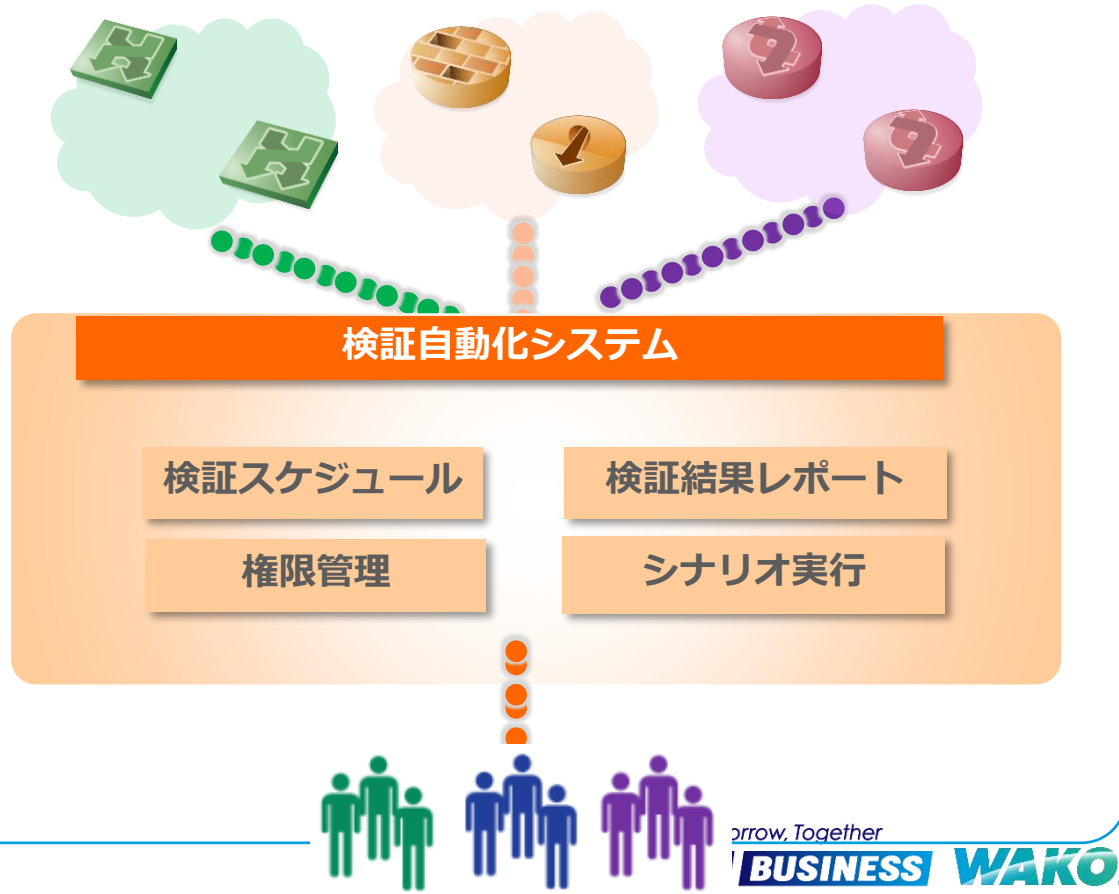
As-Is

・自動化度合いが各々異なる
・検証NWが各々分離している



To-Be

・共通PFによる検証自動化＋システムによる一括管理
・検証ナレッジの集約



自動化のターゲット

■ 自動化メリットを享受しやすいのは**単体試験・結合試験**
他領域についても現在絶賛挑戦中

環境構築

- 物理的な挿抜が必要
  →L1SWのオーケストレーションによる自動化 **検討中**
- 検証環境仮想化による自動化 **検討中**

単体試験

- 機能検証

特徴

- 案件によって実施内容がほぼ変わらない

結合試験

- 障害検証
- 負荷検証

特徴

障害発生→挙動確認→復旧　一連の流れを自動化可能
トラヒックジェネレータ操作を自動化可能

手順試験

- 案件によって実施内容（手順）が変わる
  ※定型作業であれば自動化可能

実現手法

実現手法について

■ 請負開発？　内製開発？メンバ間で真剣に議論実施

機能的には大きな差分無

リリース
スピード

操作性

請負開発　：コスト高そう
　　確実に実施できそう

内製開発　：コスト安そう
　　既存業務があるから稼働が大変そう..

両者メリット/デメリットがあるので平行線

これからのネットワークエンジニアとして
あるべき姿って…

今の保持スキルって
5年先も有効？

生産性を向上させないと
今の人数ではそろそろ限界

既存業務に割く時間を削減し、
空き時間で新スキルを習得
して欲しい

実現手法について

メンバ間で真剣に議論を重ねた結果、

内製開発の懸念である
「稼働が大変そう。。」
これは
「我々でしっかり手を動かして開発する。
　すなわち**新しいスキル**を身につけられる」
ことの裏返しであることに気づく。

持続可能な組織へのレベルアップを目指すため、
内製開発で進めることを決断

忙しくなるけど、絶対良い経験になるはず！

持続可能な開発組織

■ AI時代到来により**持続可能な開発組織** へのレベルアップが求められており、
そのためにも
マルチスキル（ネットワーク＋ソフトウェア）人財
を目指す必要があった。

背景

・少子高齢化に伴う労働人口減
・AI時代の本格到来

当時の我々

Lv.0

・**手動検証**

Lv.1

- 検証自動化の概念を**理解**している
- VSCode・Gitを使える

Lv.2

- **簡単**な自動化シナリオを**記述**できる
APIを使ってNW機器や管理システムと連携できる

全員ココを目指す

Lv.3

- **複雑**な自動化シナリオを**記述**できる
- メンバーへの教育・**標準化**ができる
- **コードレビュー**ができる

過去事例の分析

■ 過去の失敗例
特定設備の検証自動化を目的としてOSSの自動化ソフトウェアを導入。
業務委託で社外エンジニアに構築いただいたが、上手く浸透せずに陳腐化。

■ 失敗した理由

❌現場視点不足
開発段階において、実際に検証を行う開発担当者との対話や
フィードバックの機会が少なく、現場ニーズに十分対応できていなかった。

❌メンテ負荷が高い
シナリオの修正時にプログラミングが多く発生するので
設備開発担当者の負担が大きかった。

■ 得られた教訓

✅設備開発担当者の巻き込み
現場エンジニアとの対話を重視し、**現場に寄り添いながら開発**を進める。

✅持続的な運用を見据えた設計
シナリオ自体の**メンテがなるべく楽になる**ようにモジュールを開発する

初期開発対象の選定

初期開発対象について

当部で主管する数ある設備の中、
SD-WAN設備の検証（単体試験）を初期開発対象として選定

＜選定理由＞

・セキュリティ対応で毎年OSバージョンアップ（に伴う検証）を実施
→自動化による効果が高い。
・自動化に対して一番前向きに見えた
（ヒアリングの際、アンケートを最も詳細に記載してくれた）

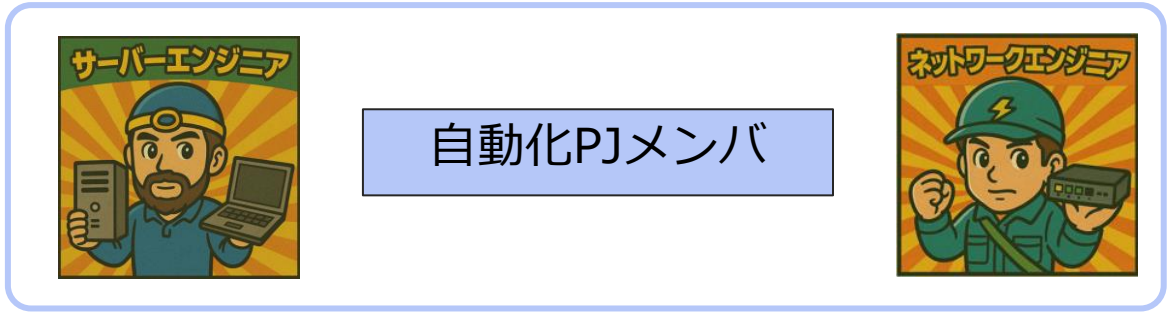
＋

・自動化プロジェクトメンバにとって未経験の設備・機器
→**設備開発担当者を巻き込んでいくムーブ**が必ず必要になる

今後、部内外に横展開していく上で
絶対求められる動き

開発チーム体制について

■ サーバエンジニア・設備開発担当者・プログラマにも参画してもらい、
開発チームを結成。様々な専門性を持つメンバで密に連携し開発を推進。



実現方法すり合わせ
追加機能要望

・Slack
・日次・週次MTG

シナリオ作成/レビュー

設備開発担当者

判定基準
質問/回答

シナリオ開発担当者

自動化システム開発について

開発キックオフ直後に噴出した課題

- 課題①：**NWエンジニアが自力でシナリオ修正・追加出来る実装とは?**
  - 通常の業務で忙しく自動化習熟に十分な稼働を割くことが難しい。
- 課題②：**拡張性の高いプラットフォーム構成をどのように実現するか？**
  - 設備によって検証場所が違う。自動化用環境構築の手間を最小にしたい
- 課題③：**NWエンジニアが自動化についてよくわからない**
  - 何が実現できて、何が実現できないのがわからない
- 課題④：**プログラマがNWについてよくわからない**
  - 試験手順書が何を書いてあるか分からない
  - 何をもって試験を合格とすればいいのかわからない

NWエンジニアのためのローコードな仕組みづくり

Robot Flamework

- 試験シナリオを記述。主要な処理はPython側で実装
- **数日程度のOJT**でNWエンジニアもシナリオを作成可能

呼び出し ↓ ↑ 結果出力

robotから直接呼出しを受ける機能部

Python

呼び出し ↓ ↑ 結果出力　　呼び出し ↓ ↑ 結果出力

NETMIKO　NTC Template　Genie Python Test Automation

自作判定モジュール群

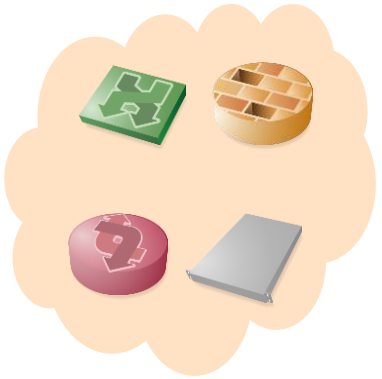
僕でもできる

本業で時間がないNW屋

縁の下の力持ち達（SV・PG）

telnet/ssh
コマンド実行 →

← 結果出力



シナリオ実装例

- コマンド実行で3パターン、条件判定として7つパターン用意し、**組み合わせにより自動化シナリオ作成可能な仕組みを作成**
- 属人的な記述ブレのリスクの最小化のために、実装規約についても詳細に規定

```
>> 2-1. ステータス確認
    [Documentation]    ${\n}対象ホスト: ${cedge_01}[hostname]
    ...                ${\n}実施コマンド: show ip ospf neighbor
    ...                ${\n}判定条件: Initではないこと
    ...                ${\n}判定方法: neighbor_id: 10.10.10.1, state: Init以外
    ${command_list}=    Create List    show ip ospf neighbor
    ${command_result}=    CF.Send Command    ${command_list}    ${cedge_01}[hostna
    ${show_router_log}=    CF.Show Router Log    ${cedge_01}[hostname]
    ${result_list}=    CF.Parse Template
    ...    cisco_xe
    ...    show ip ospf neighbor
    ...    ${command_result}
    Log    ${result_list}
    ${check_dict_01}=    Create Dictionary
    ...    neighbor_id=10\\.10\\.10\\.1
    ...    state=^(?!Init$).+$
    ${check_dict_02}=    Create Dictionary
    ...    keywords=${check_dict_01}
    ...    operation=eq
    ...    count=1
    ${check_list_01}=    Create List    ${check_dict_02}
    ${judge}=    CF.Check Result Values    ${result_list}    ${ch
    CK Judge And Log On Failure    ${judge}    ${err_msg}    ${s
```

[customer_edge01]: show ip ospf neighbor

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.10.10.1	1	FULL/DR	00:00:39	10.10.10.2	Gi0/0
10.20.20.1	1	FULL/BDR	00:00:39	10.20.20.2	Gi0/1
10.30.30.1	1	Init	00:00:39	10.30.30.1	Gi0/2

ntc-templateでパース

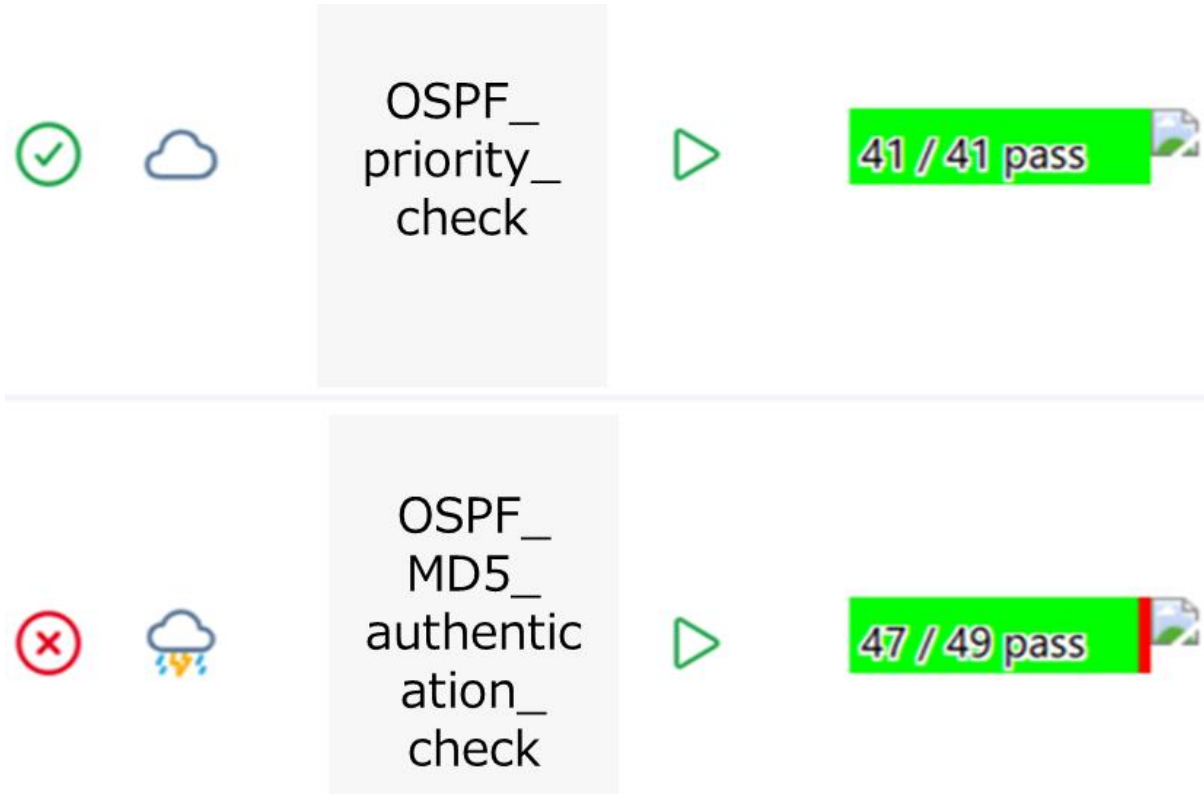
```
[
{neighbor_id:10.10.10.1, priority:1, state: 'FULL/DR', dead_time:
'00:00:39', address: '10.10.10.2', interface: 'Gi0/0'},
                              :
                              :
]
```

パース後の結果にneighbor_id=10.10.10.1かつstateがInitになっていない要素が一つあること

プラットフォーム選定

- 各検証環境にagentを配備し、agentを束ねる管理サーバを構築する形であれば、環境構築稼働を最小化出来ると想定。
- CI/CDツールの中から**実行結果の確認が容易であったJenkinsを選定**

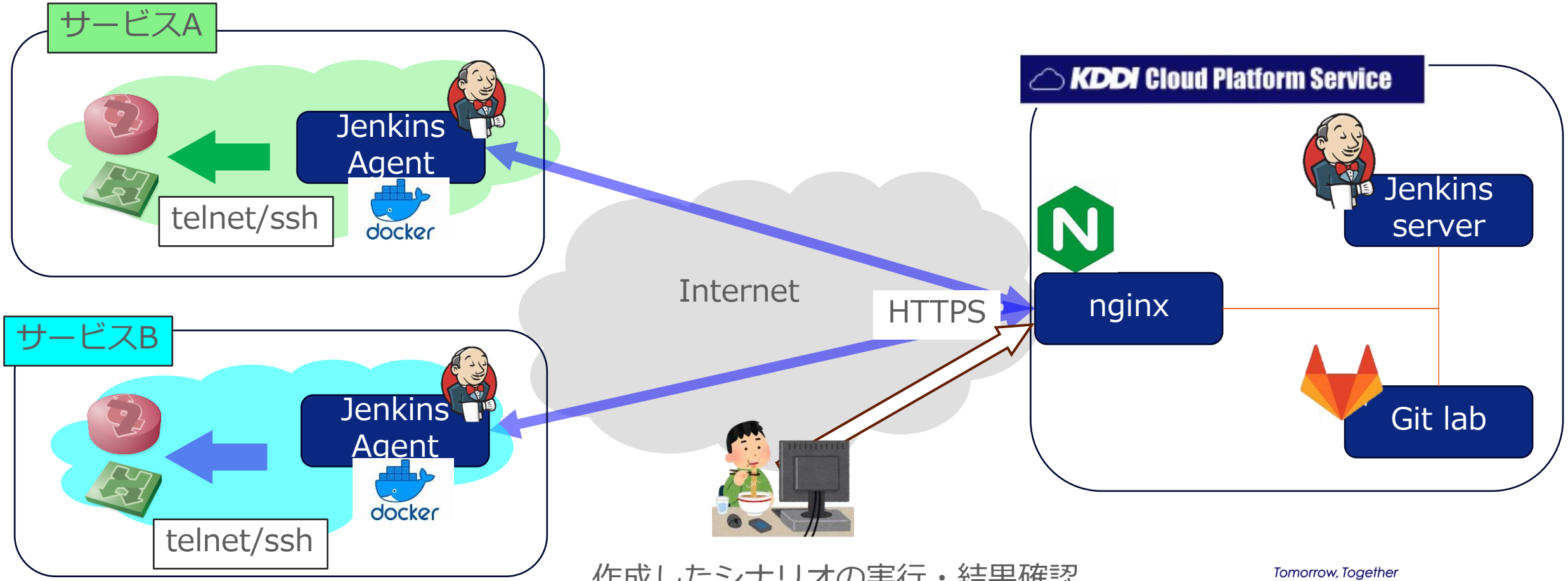
	Jenkins	GitLab CI
CI/CDツール		
プラグインの豊富さ	◎ Robot Flamework のプラグインも有	×
シナリオ管理の容易さ	◎ GUIで管理可能	△ 基本YAMLを編集
結果の視認性	◎ 結果の集計表示可能	△ 成否の一覧のみ



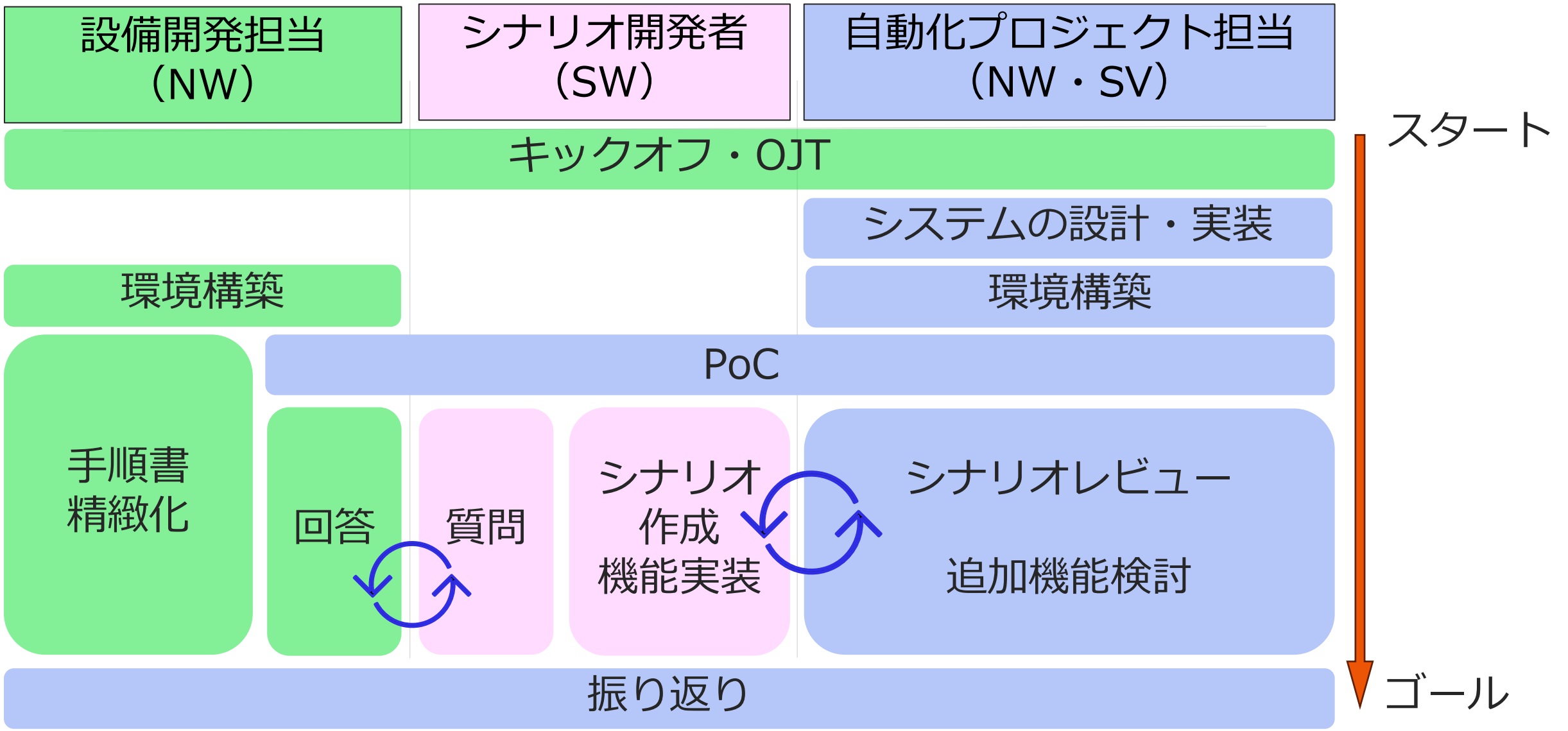
jenkinsの結果画面（※発表用に若干加工）

拡張性の高いプラットフォームの構築

- クラウド基盤に構築したJenkins serverを管理コンソールとして活用
- server-agent構成により**検証環境が物理的に異なる場合でも容易に環境構築が可能**
- Jenkinsのプラグインを活用し、シナリオの一括実行の様な必須機能も実装



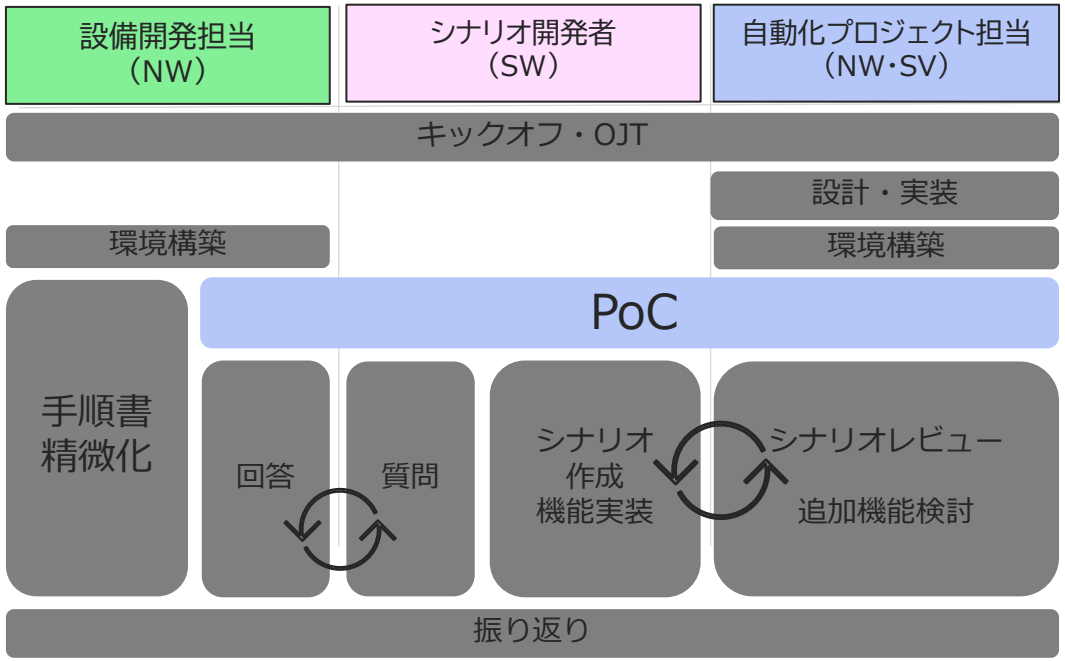
開発の流れ



NWエンジニアが自動化を理解するためのPoCの実施

- 設備開発担当が早期に完成形をイメージできる
  - 想定より自動化が簡単そうで**単純にモチベーションが上昇**

- 自動化用の手順書をどのように作成すべきか？イメージできる
  - 判定条件はどのように設定すべきかについて理解できた

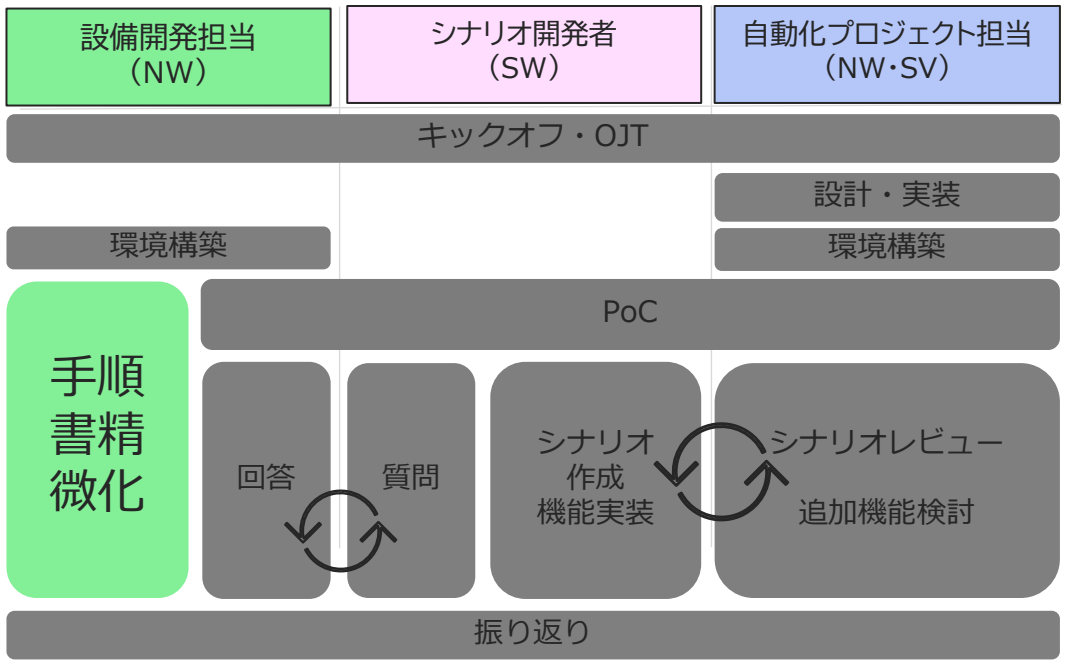


本音を隠して上司に報告したPL

プログラマがNWについてよくわからないことは許容

- 短い期間で構成・OS・プロトコルの異なるNWの検証の自動化を実施していくため、**プログラマ側で各NWを習熟することは困難と判断**

- 手順書を詳細に記載することで、プログラマでも試験内容を判別できるようにすることで対処



大項番2：検証

項番名： **ステータス確認**
対象装置：**custeomer_edge_router01**
実施内容：**sh ip bgp vpnv4 vrf 1000 neighbors 172.16.51.2 routes**
判定基準：**LANrouterからの受信ルートにMED200が付与されていること**

```
custeomer_edge_router01#sh ip bgp vpnv4 vrf 1000 neighbors 172.16.51.2 routes
BGP table version is 23, local router ID is 172.16.255.111
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:1000 (default for vrf 1000)
 *   172.16.3.0/24    172.16.51.2              200             0 64530 ?
 *   172.16.51.0/30   172.16.51.2              200             0 64530 ?
 *   172.16.51.4/30   172.16.51.2              200             0 64530 ?
```

正常性確認で参照すべき値を赤字で指定

実際の手順書から抜粋

開発の振り返り

- 開発期間：2024/11~2025/3
  - 設計・実装　　：2024/11~12
  - 環境構築・PoC　：2025/1上旬
  - シナリオ作成　　：2025/1下旬~3

- 作成シナリオ数:130個
  約20シナリオは初めて自動化に挑戦するNWエンジニアが作成を担当

- 自動化による新OS検証：2025/5~
  バージョンアップに伴うシナリオ修正も**設備開発担当にて対応**し検証継続中

- 同一のシステムを用いて、別設備の検証自動化も推進中

自動化してみて

検証業務プロセスの変化（Before→After）

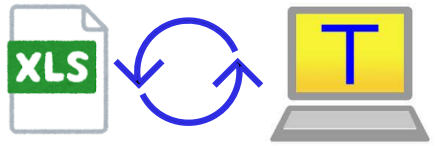
- ワンクリックで検証を実施できるようになることで、
  検証に要する期間を約85%短縮できた。

- 空いた時間で、新サービスの技術検討を行うなど、
  中長期先を見据えた活動に時間を割くことができるように

検証を外部委託してた場合、
不要になるのでコスト削減になる

Before

100項目

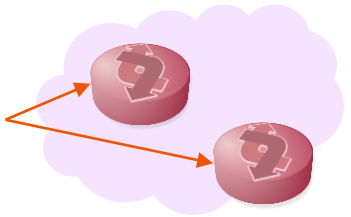


エクセルの検証手順書を使って手動実施
商用の構成パターン分繰り返し実施

After

100項目
試験実施

1 click



試験実行後の結果確認のみ実施

※イメージです

Tomorrow, Together
KDDI BUSINESS WAKONX

自動化による付随効果

■ **検証項目・手順の刷新**

検証手順書修正を通じて、
改めて見直すと意外と無駄が多いことに気づく。
重複した項目の削除・確認観点の追加など**実情に合わせて刷新**できた。

■ **前向きな雰囲気の醸成**

組織として**自動化に本気で取り組もう**という雰囲気を醸成できた。
自動化対象案件もプロジェクト開始当初より集まりやすくなった。

見えてきた課題

■ **やはり瞬間的に多くの稼働がかかる**

稼働がかかる項目：手順書修正、テストシナリオレビュー、QA対応

平均して**40%の稼働**を割いていた
プロジェクトにかける稼働量に対する認識を
組織内で上司含め事前に認識合わせる必要有り

■ **地道な草の根活動の重要性**

トップダウンだけでは不十分。
確実に浸透させるためには、
「設備開発担当者が前のめり」で参画することが不可欠
横展開を見据えると、組織内の全員が自動化を身近に感じることが必要。
地道な広報活動（ハンズオンなど）を継続して行っていきたい。

今後の展望

検証チームの目指す姿

■ 設備開発担当者が**マルチスキルエンジニアへと進化**し、
シナリオのメンテ・開発をリードできる姿を目指していく。

現在

自動化PJメンバ

自動化システム構築、
シナリオ開発の議論

設備開発担当者

サービスに
関する質問

シナリオ開発担当者

目指す姿

設備開発担当者

検証自動化システム

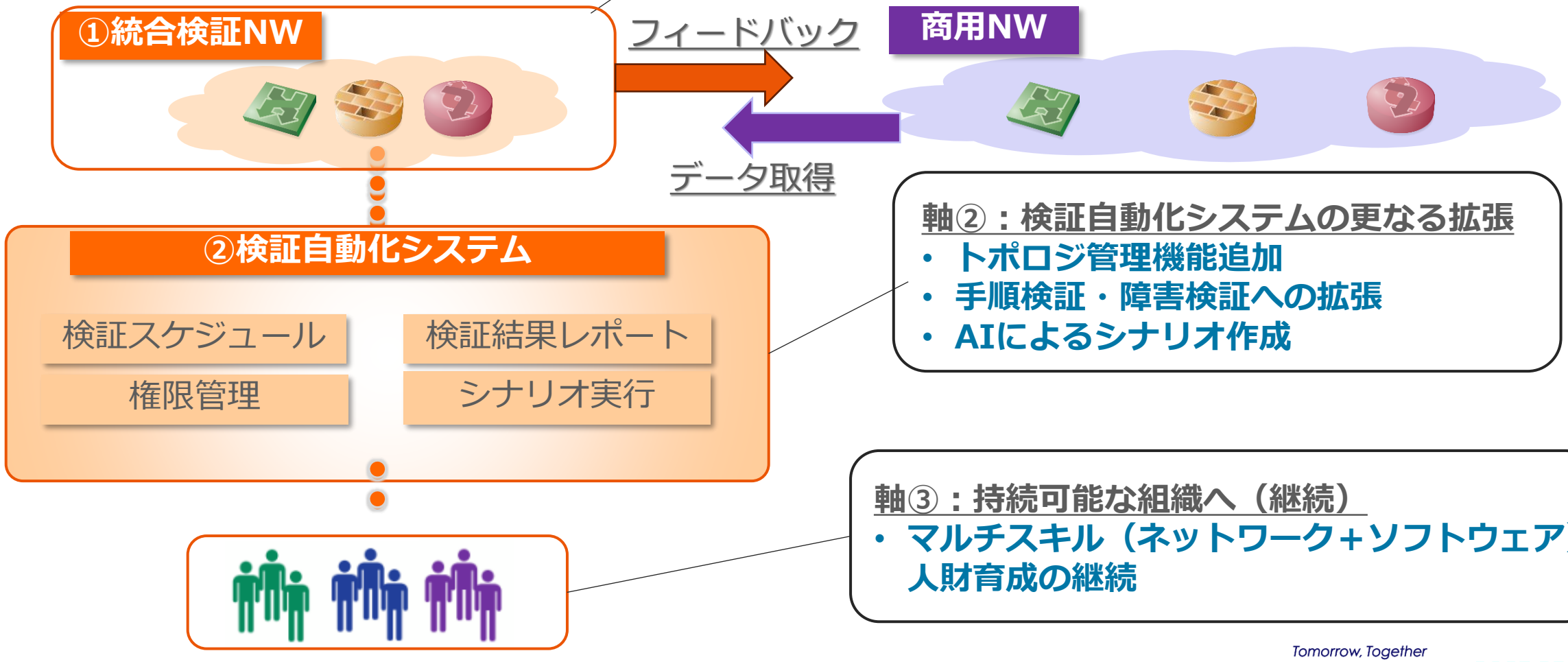
シナリオ開発担当者

自動化の議論
サービス仕様の議論

- 設備開発担当者が主となりシナリオメンテを実施。
- (大規模開発の場合)プログラマの成果物レビューを実施。

検証業務の目指す姿

今後の展望



軸①：次世代ネットワークラボ
- L1スイッチ導入による
  NWラボの完全リモートオーケストレーション
- 仮想ネットワークラボ（デジタルツイン）

軸②：検証自動化システムの更なる拡張
- トポロジ管理機能追加
- 手順検証・障害検証への拡張
- AIによるシナリオ作成

軸③：持続可能な組織へ（継続）
- マルチスキル（ネットワーク＋ソフトウェア）
  人財育成の継続

さいごに

議論ポイント①

現場に自動化を導入する中で、思い通りにいかなかった経験はありますか？
また、自動化を現場に”浸透”させるために意識している点はありますか？

議論ポイント②

ぶっちゃけ自動化しなくてもいいケースってありますか？

議論ポイント③

ネットワークエンジニアに今後求められるスキルセットや役割は
どのように変化すると思いますか？

Tomorrow, Together

KDDI

「つなぐチカラ」を進化させ、
誰もが思いを実現できる社会をつくる。

KDDI VISION 2030