JANOG56発表資料

現場の多様性に応える！
～ネットワーク作業自動化プラットフォームの浸透～

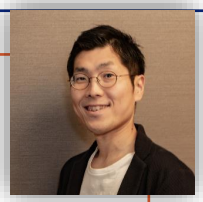
渡辺みどり (KDDI株式会社 コア技術統括本部 ネットワーク開発本部 ソフトウェア開発部 開発2G)
長野　知幸 (KDDI株式会社 コア技術統括本部 ネットワーク開発本部 ソフトウェア開発部 開発2G)
藪原　穣　(KDDI株式会社 コア技術統括本部 アクセス技術本部 ネットワークシステム設計部 フォトニクスネットワークG)

2025/07/30

Tomorrow, Together
KDDI

本日お話しすること

1. はじめに
2. 前回発表のおさらい
   - プロダクトのコンセプト
   - 自動化支援プラットフォーム概要
   - 部門横断チーム
   - 広めるためのモチベーション
3. プロセス面の課題と解決策
   - 広めるために実施したこと
   - 新規ユーザーの導入障壁
   - ユーザーニーズの理解不足



4. プロダクト面の課題と解決策
   - Jupyter手順書作成が難しい
   - ベンダー差分
   - GUIへの自動化が未対応

5. GUIの操作を自動化してみた
   - CLI文化とGUI文化の比較
   - GUIの自動化手法
   - デモ
6. 全体まとめ
7. ディスカッションしたいこと

登壇者紹介

渡辺　みどり
（Midori Watanabe）
ソフトウェア開発部
md-watanabe@kddi.com

長野　知幸
（Tomoyuki Nagano）
ソフトウェア開発部
to-nagano@kddi.com

藪原　穣
（Osamu Yabuhara）
ネットワークシステム設計部
os-yabuhara@kddi.com

はじめに

自己紹介

渡辺 みどり
（Midori Watanabe）

所属	ネットワーク開発本部 ソフトウェア開発部
経歴	ロ開発（2022～現在） 5GC NF開発 NW作業の自動化
JANOG参加歴	54@奈良 55@京都 **今回、初登壇**

前回発表（JANOG 55@京都）のおさらい

ネットワーク作業の自動化支援プラットフォームのコンセプト

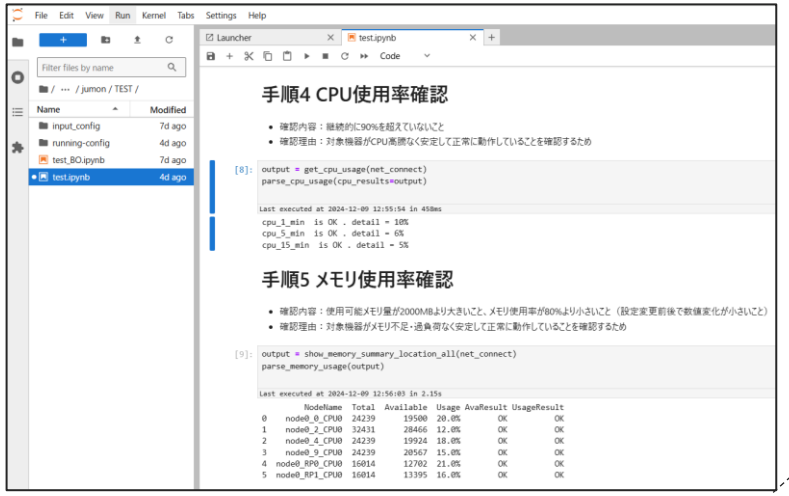
プロダクトのコンセプト

■ネットワークエンジニアが、**すぐに使えて、改善を継続的に実施できる自動化支援プラットフォーム**

＝開発チームが自動化を行う

ネットワーク作業の自動化支援プラットフォーム概要(前回発表サマリ)

■ 従来の手作業を簡単に自動化し、作業も楽に実行できる環境



Jupyter手順書(※)

Jupyter手順書：従来の作業手順書をJupyter Notebook形式にした作業手順書。
MarkdownによるメモとPythonコードが一体化したもの

特徴

- 誰でも同じ結果が得られる
- 理解が簡単で汎用性が高い
- 今の作業品質を維持できる
- 簡単に手順書を作れる

どうやって成し遂げたのか？

■ 部門横断チームを結成し、自動化を推進した

常識に捉われないクロスファンクショナルチーム

JANOG55 発表資料
11

■ ネットワーク（NW）エンジニアとソフトウェア（SW）エンジニアで構成された
部門横断のクロスファンクショナルな開発チームを**トップダウン**で結成

■ SWエンジニアによる客観的な視点により、NWエンジニアの
固定概念に捉われないソリューションを導くことができた

NWエンジニア
SWエンジニア

（これまで）NWエンジニアのみ	NWエンジニア×SWエンジニア

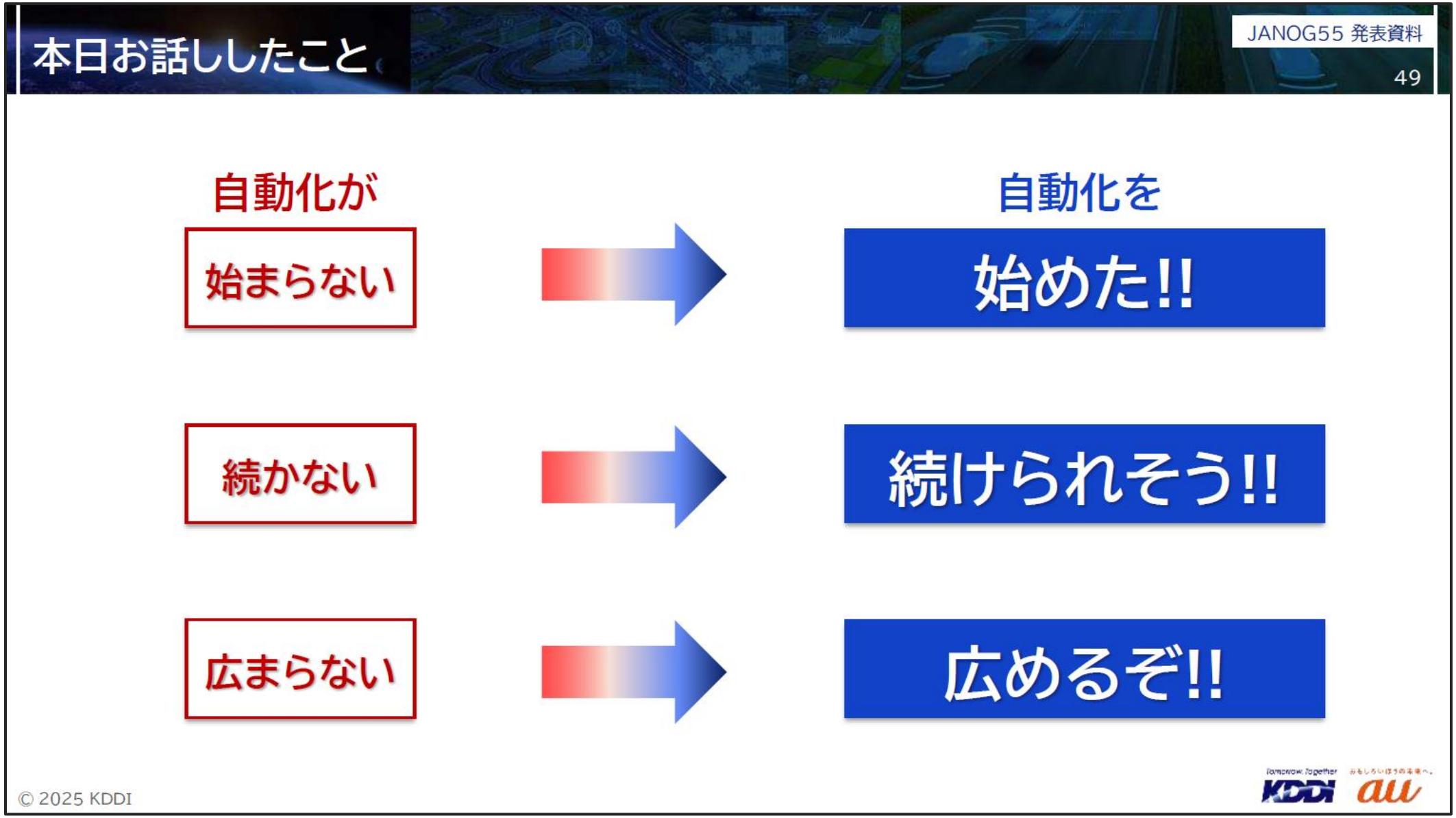
▲ すきま時間での開発
→自動化が**始まらない、完成しない**

▲ NWエンジニアの固定概念
→**特定作業単発**の自動化ツール

○ 体制を組んで開発
→自動化に**注力できる、完成する**

○ SWエンジニアによる客観的な視点
→**他作業でも利用可能**な自動化設計

その結果、自動化できた!

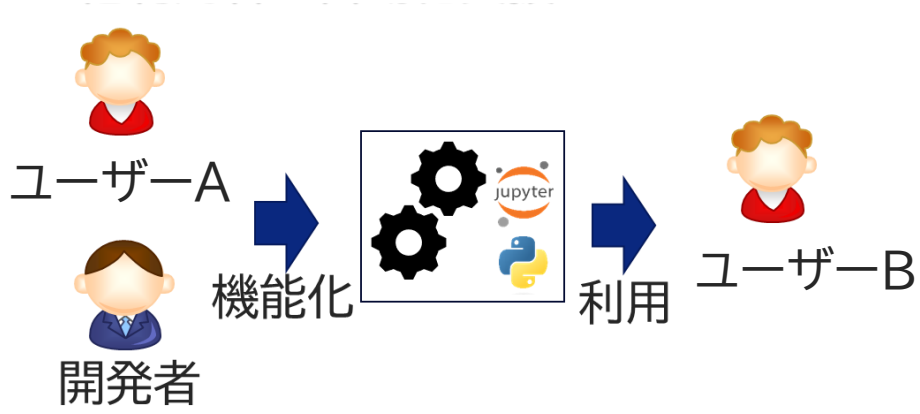


自動化支援プラットフォームを広めたいモチベーション 広めるぞ!!

- 自動化したいけど多忙、スキル不足、自動化後の動作保証への懸念により、自動化できていない人を助けたい
- 多様な領域のノウハウを機能として反映して、他利用者の自動化支援に利用したい

自動化できていない人を助けたい

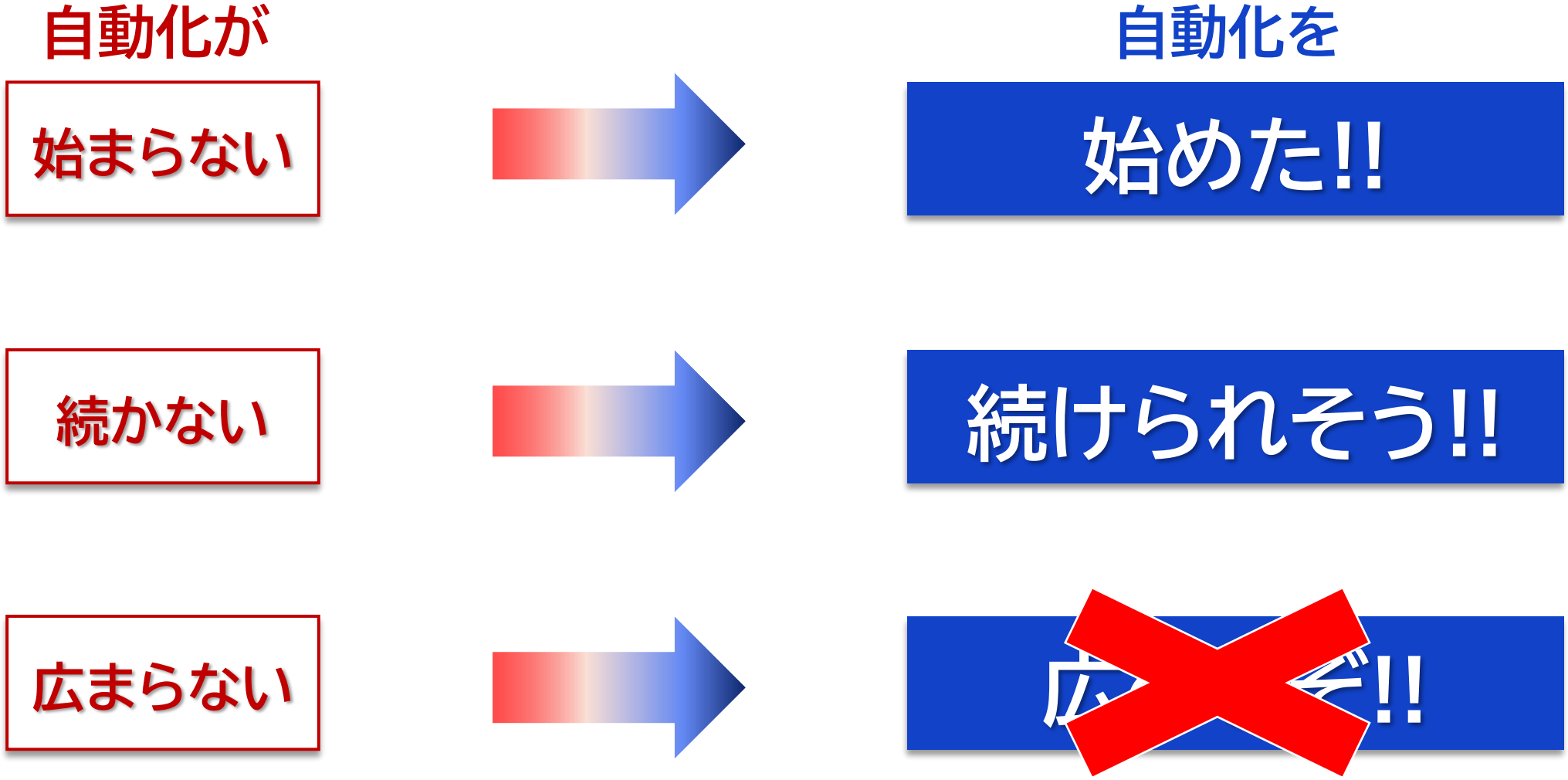
ノウハウの共有



ところが・・・

前回うまく行った流れで、広められると思った、、、

■ 0→1にはできたけど、1→nは難しかった



自動化を広めるにあたって立ちはだかる壁とその解決策

やりたいこと

1->nへの展開

立ちはだかる壁

- 新規ユーザーの導入障壁
- ユーザーニーズの理解不足

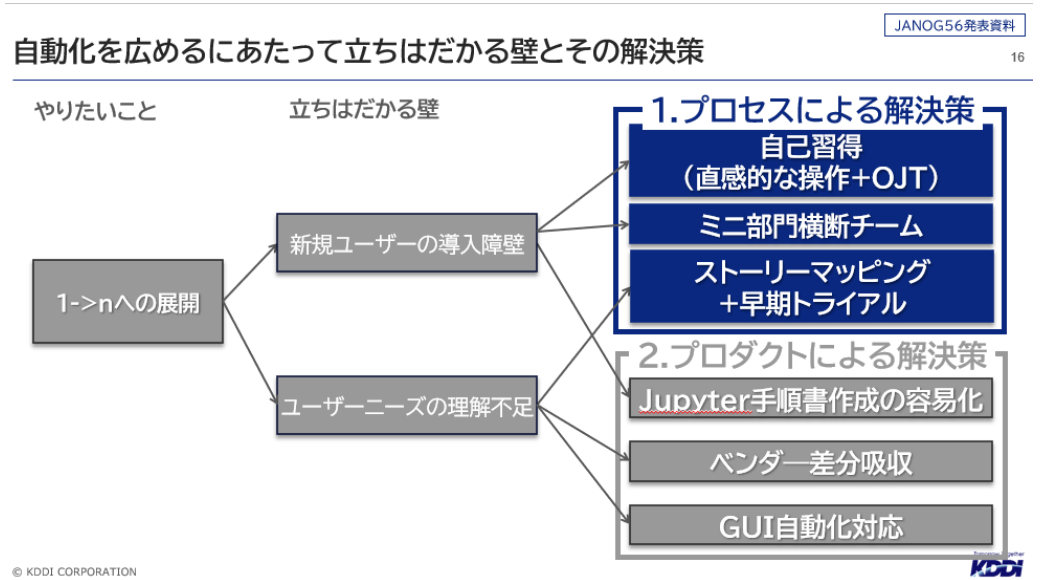
1.プロセスによる解決策

- 自己習得（直感的な操作+OJT）
- ミニ部門横断チーム
- ストーリーマッピング+早期トライアル

2.プロダクトによる解決策

- Jupyter手順書作成の容易化
- ベンダー差分吸収
- GUI自動化対応

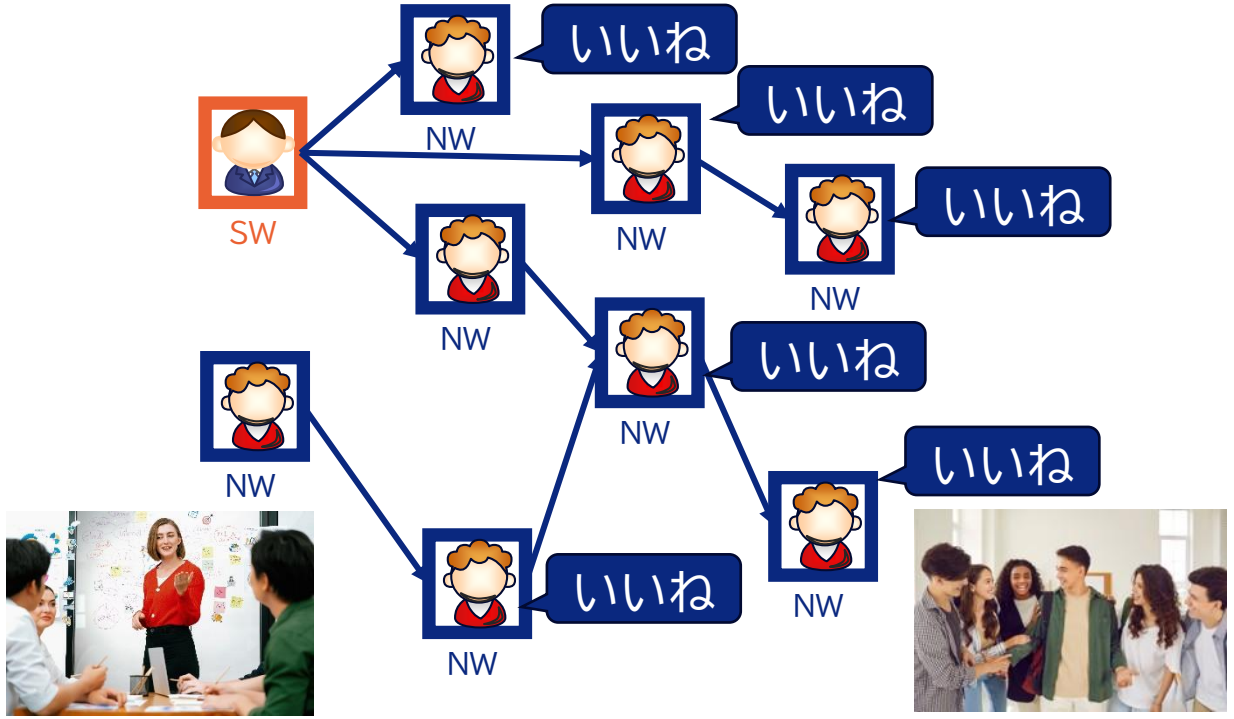
プロセス面の課題と解決策



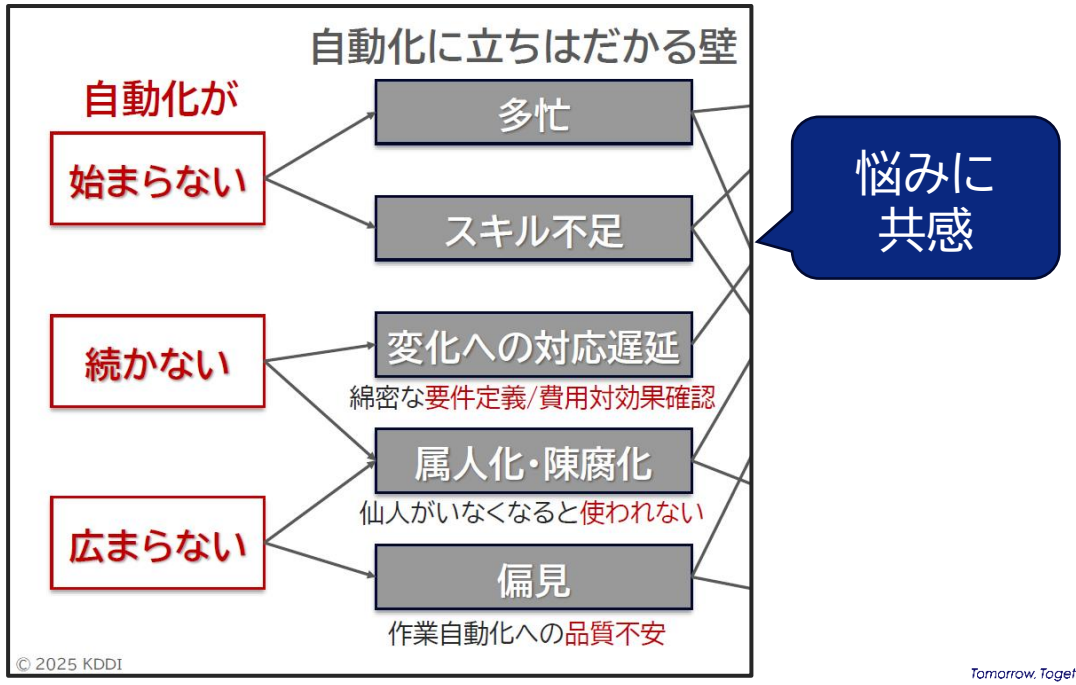
広めるために実施したこと

- 更にKDDI社内に広げるために、人脈を活用して営業活動を実施
- 社内でも作業自動化を進めたいけど進められない、同じ悩みを持っている人がおり、気づけば十数部門から利用希望が来る状況に

地道な営業活動

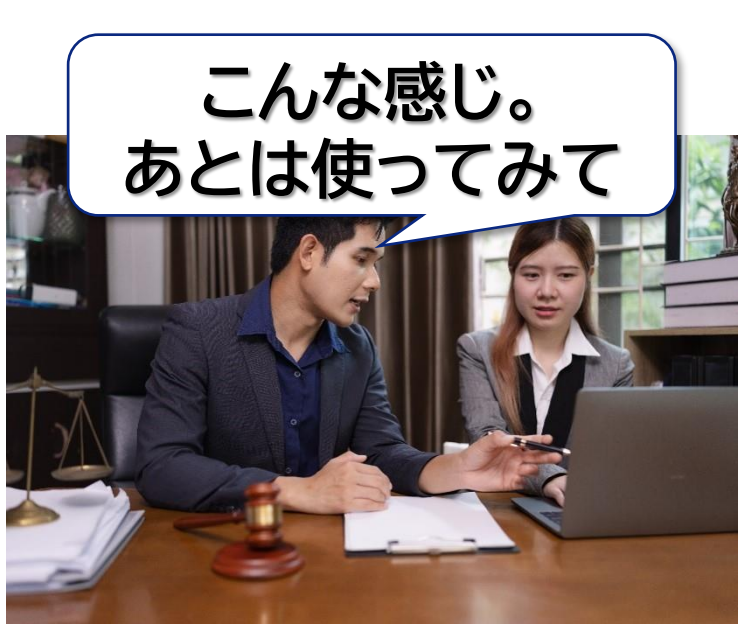


ユーザーが自動化を進められない理由



広めようとしてみた

- 既にシステムとマニュアルはあった。既存ユーザーと同じくNWに関する作業であれば使ってもらえると判断し、そのまま、使ってもらった
- 結果、使えないと言われた



開発者

新規ユーザー

開発者

使えなかった理由は？

- 展開を続けるには立て直しが必要だと感じ、どうすれば良いのか開発チーム全員で協議を重ねた。その結果2つの課題が明らかになった
- **ユーザーのやりたいこと、スキル、背景は1人1人異なる!**

課題1:新規ユーザーの導入障壁

- 手順が複雑、理解する時間不足、疑問の多発

課題2:ユーザーニーズの理解不足

- 必要な機能不足、必要な改良点にも気づけず

”なぜ” このような課題が発生したのか？　～根本原因～

■ 部門展開を優先した結果、**稼働不足**と**思い込み**が生まれ、
導入障壁やミスマッチをもたらしていた

課題1：新規ユーザーの導入障壁

ユーザー

多忙で、新しいシステムの
使い方を学ぶこと、大きな
時間を割くことができない

開発

部門展開を優先し、開発チームの
負担を抑えた結果、
十分なサポートができなかった

原因 → **稼動不足（ユーザー・開発双方）**

課題2：ユーザーニーズの理解不足

ユーザー

このシステムでやりたいことを実現
できると思っていた

開発

部門展開を優先し、ユーザーが
やりたいことの調査が不足し、
既存機能でカバーできると思っていた

原因 → **思い込み**

解決策のコンセプト

■ **最小限の稼動**で誰でもすぐ使えるを実現し、**ユーザーの本音を知る仕組み＋早期ユーザートライアル**で稼動不足と思い込みを解消

課題1の原因：稼働不足


ユーザー

- 誰でもすぐに、使えるようになりたい


開発

- 部門展開時の稼動に開発チームが耐えられるような展開方法が必要

コンセプト **ユーザー・開発チーム双方とも、最小限の稼働**

課題2の原因：思い込み


ユーザー

- やりたいことが、本当にできるのか簡単に確認したい


開発

- ユーザーの本当の困りごとや要望を直接知りたい
- 部門展開と個別対応を両立できる方法が必要

コンセプト **ユーザーの本音を知る仕組み＋早期ユーザートライアル**

課題1: 稼働不足の深掘り

- 複雑な手順×現場の多忙 → 使い方が理解されず“自律利用”が進まない
- 今まで通りの進め方では、部門展開のスピードに限界

課題1-1:ユーザーの稼働不足

①手順が複雑でわかりにくい
②業務の忙しさで理解する
時間が取れず、後回しに
③簡単なレクチャーやマニュアルでは
疑問が多く、疑問解消までに
時間がかかる

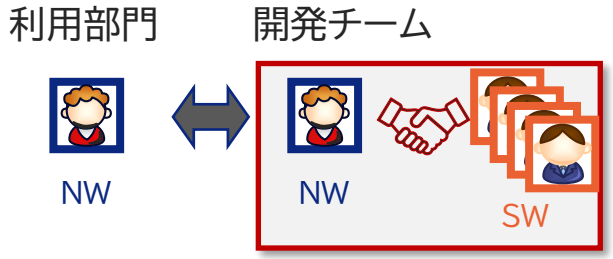
→自律的に利用を進めることが困難

課題1-2: 開発チームの稼働不足

部門横断チームを作ると“1部門＝3ヶ月”ずつとなり、
利用希望に即応できない

十数部門からの
利用希望

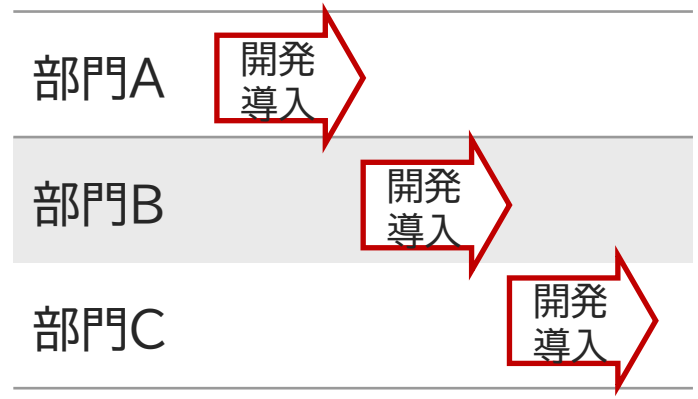
部門横断チーム（前回説明）



NW+SWエンジニア
による開発チーム

開発・導入まで3ヵ月/部門

部門横断チームで進める場合

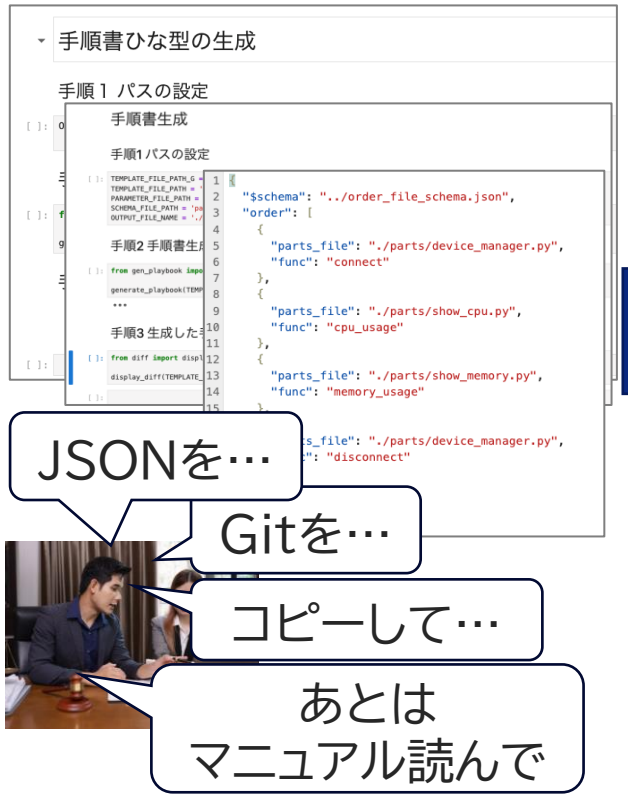


10部門×3ヶ月/部門=30ヶ月

【解決策】課題1-1: ユーザーの稼働不足

■ 直感的な操作 ＋ 厳選OJTカリキュラムで、多忙な現場でも最小限の稼働で使えるようになる仕組みを作りあげた

複雑な手順



直感的な操作+OJT

①手順が複雑

- 手順の単純化+GUI化で、直感的操作を実現

②多忙（理解する時間不足）

- スケジュールを明確に設定することで、多忙な現場での時間を確保
- 短時間での習得に向け、OJTカリキュラムを厳選

③疑問の多発

- “すぐ”に疑問を解消できる場を提供

【解決策】課題1-2: 開発チームの稼働不足

■ ユーザースキルに応じた2つの展開方式で、多様な現場にフィットする自動化導入を効率的に実現

自己習得（スキル有）

直感操作＋厳選OJTで、翌日から自分一人での自動化が可能に

直感操作＋厳選OJT

開発チーム稼働98%削減

ミニ部門横断チーム（スキル無）

直感操作＋厳選OJTで、最小チームでも部門展開が可能に

開発チーム稼働75%削減

課題2:思い込みの深掘り

■ 部門展開を急ぐあまり、思い込みやヒアリング不足で本質的なニーズを見逃し、想定外のミスマッチ・追加開発が発生

思い込み

ユーザー

このシステムって何できるんだろう

Pythonって言ってるから、
きっとなんでもできるよね

開発

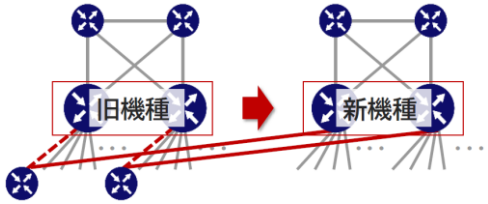
早く部門展開を進めたい！

軽く聞いた感じだと、
既存機能で対応できそう

結果、見逃されたニーズが顕在化

システムがサポート
してきたやり方

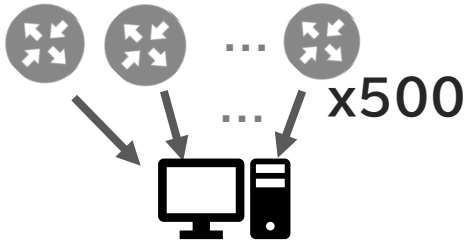
慎重なステップを
踏んだ手順で
1台ずつ収容変更



今回のニーズ

約500台のルーター
の情報を一気に取得

追加開発対応で、
スケジュール遅延



ユーザー

できる？

ユーザーからの質問をきっかけに判明

【解決策】課題2:思い込み

■ ユーザーストーリーマッピング＋早期トライアルで現場の本質ニーズをいち早く発見し、システムとのズレを最小化

ユーザーストーリーマッピング

最初に現状と理想の手順を照らし合わせ、ギャップを確認することで必要な機能が明確になる

早期ユーザートライアル

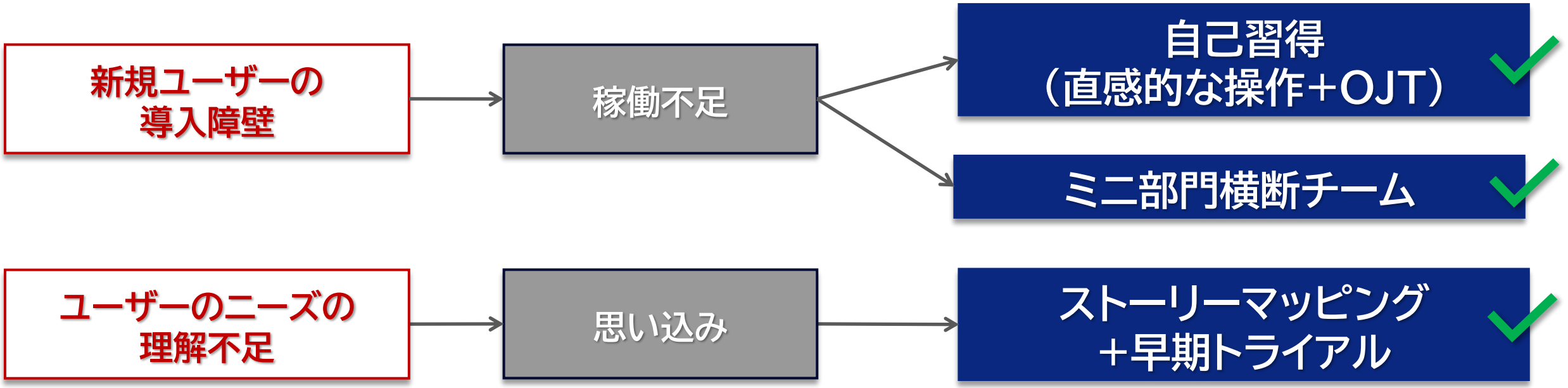
システムが利用できるか、やりたいことに必要な機能があるか、早い段階で確認

トライアル環境評価

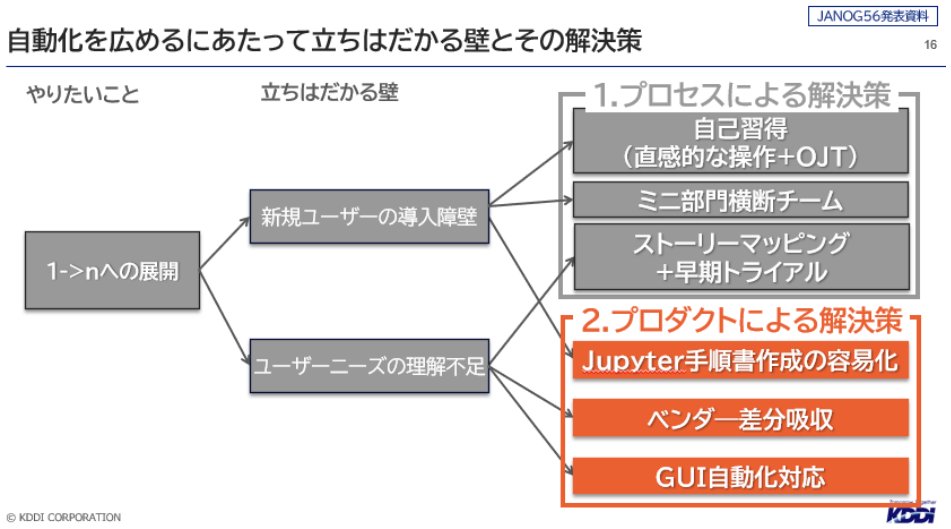
	共通トライアル環境	個別環境（採用）
概要	誰でも試せる環境	ユーザー環境と接続した個別環境
機器	開発チームが準備した機器	自動化対象の機器
評価	利用進まず △	思い込み解消 ○

まとめ

- 自動化を広げるには、現場に聴き、現場に合わせて、やり方を柔軟に変えることがカギだった
- スキルに応じて、自己習得（直感的な操作+OJT）とミニ部門横断チームの2パターンで展開。これにより、部門展開が効率的に対応可能となった



プロダクト面の課題と解決策



自己紹介

長野　知幸
(Tomoyuki Nagano)

所属	ネットワーク開発本部 ソフトウェア開発部
経歴	◘運用・保守(2002～2005) ◘開発(2005～2016) LTE向け電話システム開発、プロマネ ◘研究(2016～2022) モバイルコアNW ◘開発(2022～現在) 5GC NF開発、 NW作業の自動化(ディベロッパー)
JANOG参加歴	54@奈良 55@京都(登壇) **今回、2回目の登壇**

利用現場の展開で浮き彫りになった、「3つの課題」

■ 「使えない」と言われ、分析やヒアリングを行った結果、プロダクト面として3つの課題が明らかになった

課題1：新規ユーザー導入障壁

- Jupyter手順書を作成する手順が複雑で挫折
- Pythonを覚えるのが大変、作成に時間がかかる

課題2：ベンダー差分

- ベンダーごとにコマンド体系が異なる
- 機器ごとに担当者が分かれ、属人化が進む

課題3：GUI操作自動化

- GUI機器は自動化しにくい
- CLIが使えず自動化の対象外に
- GUI操作が手作業で残る

”なぜ” このような課題が発生したのか？　～根本原因～

■ **学習コストの高さ**、**標準化の不在**、**技術的制約** によって、自動化が「実験止まり」や「一部の人専用」になっていた

課題1：新規ユーザー導入障壁	課題2：ベンダー差分	課題3：GUI操作自動化
スキルや知識にばらつき（Python・Jupyter・プラットフォームの習熟度）	ベンダーごとにコマンドが異なる 統一されたアプローチがない	GUIしか使えない機器は自動化しにくい
▼	▼	▼
自動化できるのは一部の人	個別での自動化となる	一部の操作で手作業が残る
原因 **学習コストの高さ**	原因 **標準化の不在**	原因 **技術的制約**

既存手法の実績と限界

■ 既存手法で成果は出たが、1→nへの展開には、現状の限界を突破する新たなアプローチが不可欠

達成できたこと

- クラウド上の共通環境・Jupyter手順書・ノウハウのパーツ化といった機能により、自動化が難しかった現場での自動化を実現

自動化できるようになったぞー

もっと自動化してやる！

- 数字で見る成果
  - 手順書作成　約4日　→ 約0.5日
  - 作業時間　約6時間 → 約3時間

現状の限界

新規ユーザー導入障壁	Python・Jupyterの利用には学習が必要で、“できる人”に依存
ベンダー差分	ベンダーや現場ごとに手順が分かれ、標準化が困難
GUI操作自動化	GUI機器や非手順部分は自動化対象外のまま

新たなプロダクトコンセプト

旧　ネットワークエンジニアが、すぐに使えて、改善を継続的に実施できる

新　ネットワークエンジニアが、**誰でも**すぐに使えて、改善を継続的に実施できる

誰でも

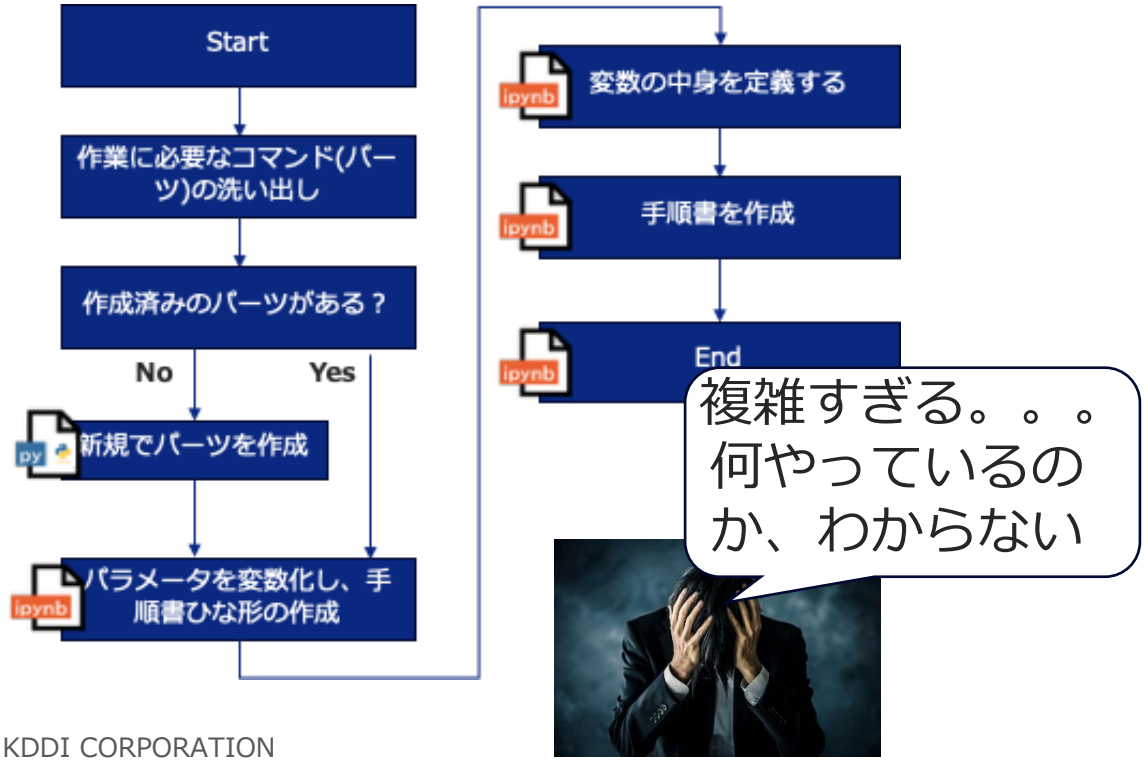
		必要な取り組み
新規ユーザ導入障壁	→	学習コストをさげる
ベンダー差分	→	標準化への取り組み
GUI操作自動化	→	技術的制約を減らす

「できる人だけのもの」から
「誰でもできる」へ

課題1. 新規ユーザー導入障壁

■ 新規ユーザーが、Jupyter手順書を作るにあたり、2つの導入障壁があった

課題1-1. 手順書作成の手順が多く複雑



課題1-2. Pythonを書けないといけない

CPU使用率確認

- 確認内容：継続的に90%を超えていないこと
- 確認理由：対象機器がCPU高騰なく安定して正常に動作していることを確認するため

Before（手動時）

```
#show processes cpu | include CPU
Wed Jul 23 11:45:44.094 JST
CPU utilization for one minute: 4%; five minutes: 4%; fifteen minutes: 4%
```

コマンド実行し、目視で判断

After（自動化後）

```
output = device.send_command("show processes cpu | include CPU")
parsed_output = parse_output(command="show processes cpu", data=output)
cpu_results = parsed_output[0]
for interval, cpu_usage in cpu_results.items():
    if int(cpu_usage) >= cpu_threshold:
        print(f"{interval}  is NG . detail = {cpu_usage}%")
    else:
        print(f"{interval}  is OK . detail = {cpu_usage}%")
```

→ 自動化には Pythonが必須

```
cpu_1_min  is OK . detail = 5%
cpu_5_min  is OK . detail = 5%
cpu_15_min  is OK . detail = 5%
```

Pythonで、出力内容を自動判定

【解決策】課題1-1. 手順書作成の手順が多く複雑

■ 手順書の作成をGUIアプリ化し、画面に表示される手順通りに進めるだけで、誰でも簡単に作れるようにした

Before

After

主な改善点

- パーツ一覧から、必要なパーツを選択・並び替え
- ファイルの自動保存機能
- 必要な入力項目は、インプットボックスで直感的に入力できるよう設計

【解決策】課題1-2. Pythonをかけないといけない

■ 生成AIを活用することで、Pythonを理解しなくても手順書を作れるようになった。さらに、確認作業も容易になった

Python知識の不要化

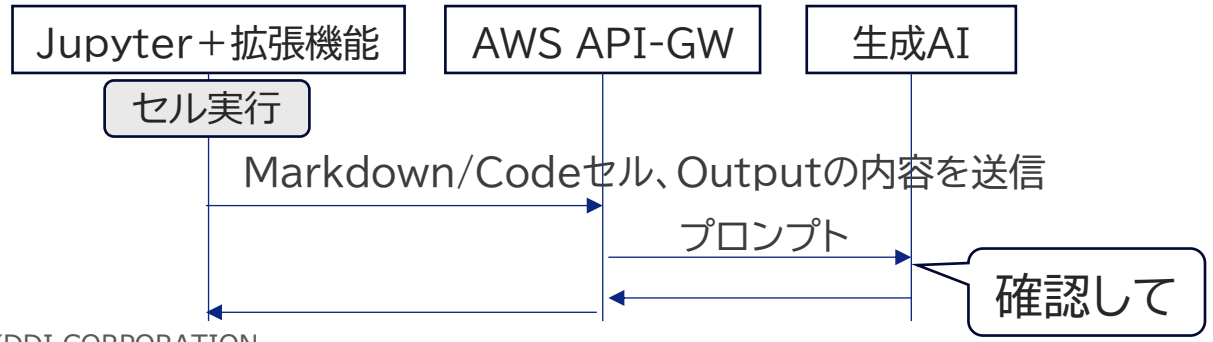
```
output = device.send_command("show processes cpu | include CPU")
parsed_output = parse_out
cpu_results = parsed_outp
for interval, cpu_usage i
    if int(cpu_usage) >= cpu_threshold:
        print(f"{interval}  is NG . detail = {cpu_us
```

```
output = device.send_command("show processes cpu | include CPU")
print(output)
```

計算&結果表示のコードを
目的に応じて記述

計算コード　→生成AI
結果表示コード→廃止
により2行でOKに

仕組み



生成AIによる実行結果の確認支援

CPU使用率確認

- 確認内容: 継続的に90%を超えていないこと
- 確認理由: 対象機器がCPU高騰なく安定して正常に動作していることを確認するため

```
output = device.send_command("show processes cpu | include CPU")
print(output)
```

```
Last executed at 2025-07-15 17:43:22 in 250ms
```

OK
CPU使用率が継続的に90%を超えていないことを確認しました
1分間のCPU使用率: 4%
5分間のCPU使用率: 4%
15分間のCPU使用率: 4%
全ての時間帯で4%と非常に低い値を維持しており、対象機器がCPU高騰なく安定して正常に動作していることが確認できました (AI自信度：100%)

AI確認結果

```
Tue Jul 15 17:43:22.302 JST
CPU utilization for one minute: 4%; five minutes: 4%; fifteen minutes: 4%
```

確認すべき箇所をハイライト

課題2. ベンダー差分

■ 「コマンドの違い」が知識格差を生み、現場の属人化・分断を生み、自動化の拡大を妨げている

実態

やりたいことは同じでも、ベンダー(場合によってはOSの違いでも)によって、コマンドが異なる。

例: インターフェース設定モードに入るコマンド

A社: interface GigabitEthernet0/1
B社: edit interfaces ge-0/0/1

影響

- 業務経験や“慣れ”で担当者が固定化される
- ノウハウが個人に偏り、全体の標準化が進まない

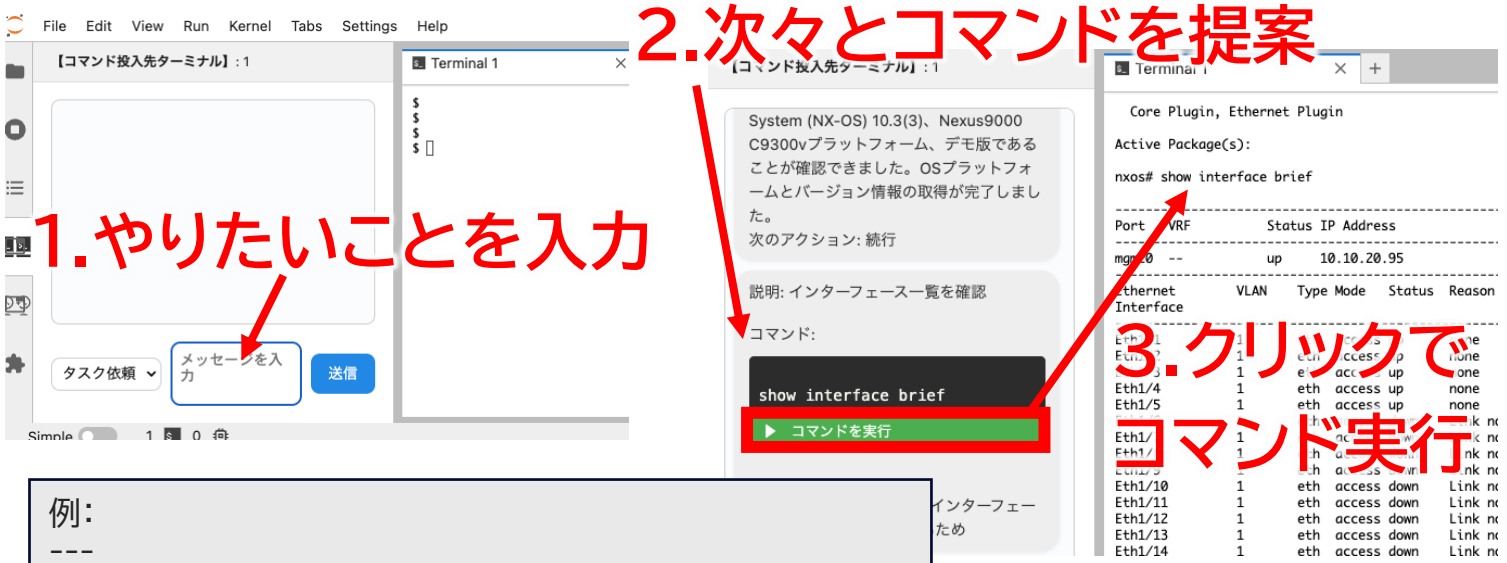
結果

自動化の拡大が非効率に

【解決策】課題2. ベンダー差分

■ 生成AIにベンダー差分を吸収してもらうことで、誰でも楽に手順書を作成できることを目指し試作中

動作イメージ



例:

以下の順番で作業を実施していってください。
1. 対象ホストにssh host: username@hostname
2. パスワードを入力 password
3. コマンド出力のページ送り(ページネーション)を無効化
4. OSやプラットフォームのバージョンを確認
5. interface一覧を確認するコマンドを実行

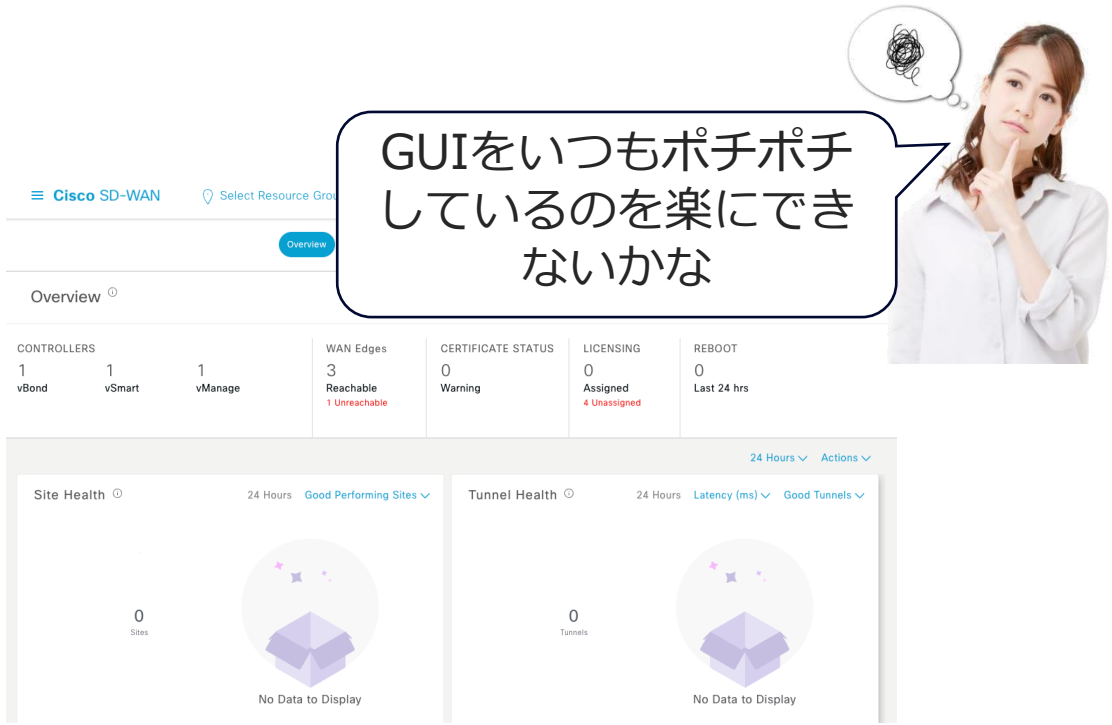
手順検証～手順書作成

ステップ	今	目指す先
やりたいことを整理	人	人
コマンド選定	人	AI
コマンド実行	人	人
結果判定	人	AI
手順書作成	人	AI

凡例: 共通ステップ / ベンダー差分有

課題3. GUI操作自動化

- GUI(Web)の操作自動化にあたって、ツールの習得やバージョンアップへの追随も考慮が必要



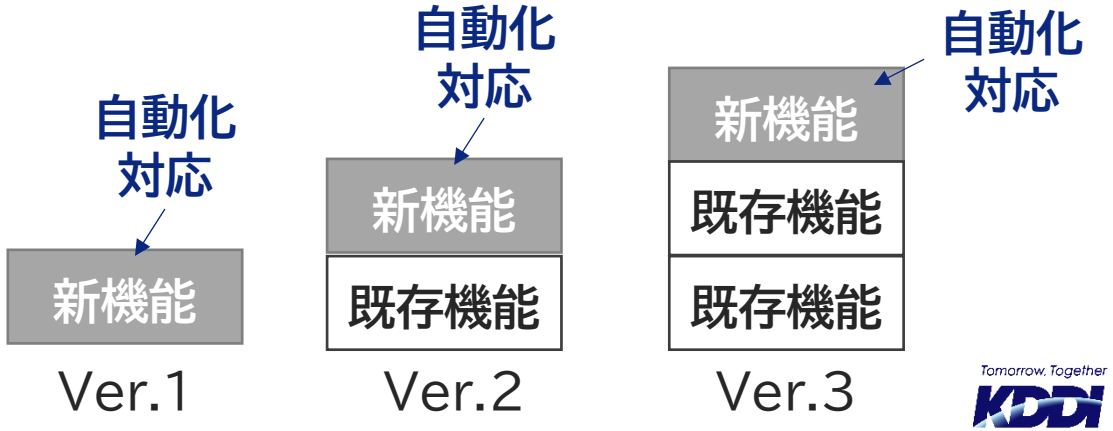
Cisco DevNet Sandbox環境

課題：ツールの習得

ブラウザ操作自動化ツール

- RPA
- E2Eテスト自動化ツール(Selenium/Playwright･･･)

課題: バージョンアップへの追随

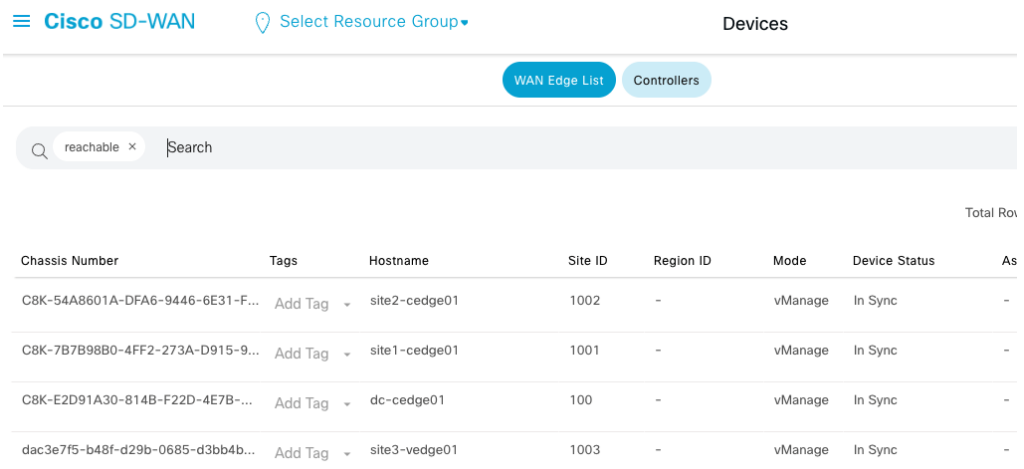


【解決策】課題3. GUI操作自動化

- Cline + Playwright MCPにて、GUI(Web)操作の自動化を実現できないか試作中

例:

Playwright mcpで以下の順番で実施していってください。
1. https://～～～ に接続
2. ユーザ名を入力 username
3. パスワードを入力 password
4. ログインをクリック
5. メニューを開く # “selector”:“～～～”
6. Configurationを選択
7. Devicesを選択
8. Searchにreachableを入力

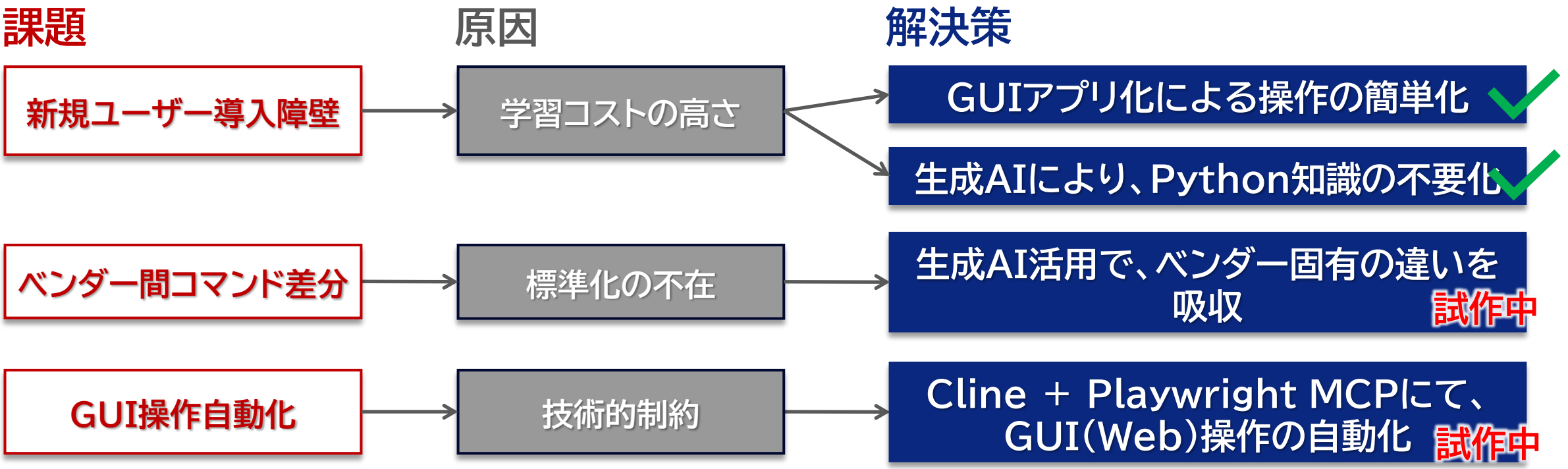


4.次々と実施内容が提案されるので、Approveしていくことで次のStepに進んでいく

デモは後ほど

まとめ

- 誰でもできる自動化を実現するには、学習コストの低減や標準化が重要
- 生成AIの活用により、従来は困難だった壁が突破でき、誰でもすぐに使える自動化の実現に向け大きく前進



興味ある人、募集中！

- 使ってみたい、もっと詳しく話を聞いてみたい、フィードバックしたい等、少しでも興味を持っていただけた方、ぜひSlackや替え玉エリア等にて、ご連絡ください！

今回のJANOGテーマ

縁(en)

メール: to-nagano@kddi.com
Slack: @長野知幸

アプリ開発部門を離れたけど Web GUI操作の自動化にトライしてみた

課題3. GUI自動化のデモンストレーション

自己紹介 - 藪原

藪原　穣
(Osamu Yabuhara)

所属	アクセス技術本部 ネットワークシステム設計部
出身・居住地	千葉県 →（ブラジル）→東京都 町田市
経歴	ロ[前職] 開発&SE(2011～2017) 100G 96ch WDMシステム開発 営業支援・導入支援・保守業務支援 ロソフトウェア開発PO(2022～2025) 5GC NF開発、NW作業の自動化 ロ光NW 設計・開発(2017～2022, 2025～) 光RF・10GE-PON WDM装置検証・導入・設備更改(ADMとかも) 伝送路構築・保守業務の高度化
JANOG参加歴	53@博多(UPFを作ろうとした話) 55@京都(NW作業自動化) 今回、登壇3回目

Tomorrow, Together KDDI

自動化やめちゃったんですか？

半年前まで　押し売り屋さん

最近　NWエンジニアの背中を押す人

背中を押していく過程で分かったこと

CLI作業自動化は割と頑張ってるが、

圧倒的に取り残されていたのがGUIの作業

光伝送領域特有のGUI文化

	伝送屋から見たL3の作業	伝統的な伝送(L0)の作業
UI	基本はCLI＋可視化用のGUI	基本がGUI~~＋ベンダ独自CLI~~
作業対象	2ノード/作業ぐらい ログイン対象少な目	5ノード以上になったりする 一度に複数ログイン
習得方法	ググれば大体すぐわかる Certificateがある	ベンダマニュアル依存 世代が変わると一新
手順書	**Diff/Compareで差分比較 自動化/自律化が容易**	**スクショが大量のExcel 自動化が困難**

なぜ伝送領域ではGUIを使い続ける？

✓ 隣接関係や障害時影響範囲の把握しやすさ

✘ CLIもあるけど、動作検証できてない or 非開示/動作保証外

とはいえ、GUI操作の**手順書を維持し続けるコストが馬鹿にならない**

- Version Upで少し画面が変わると、スクショ全部撮り直し
- ステップ数が多くなり、複雑で読み解きにくい。
- ベテランのサポートが必須。不慣れな人には扱いづらい。

維持コストを減らすために、どうにかできないか？
GUI作業手順書をそのまま自動化したら、楽になるかな？
→**アプリ開発の専門家にアイディア出してもらおう！**

GUI作業の維持をどうやって楽にするか？

■作業対象 → WebGUI

■楽にするための手段

RPA

- 社内で事務作業・プロビ作業に利用実績あり
- 今のところ伝送GUIでは使ってなさげ
- 逐次確認していくのには、不向き？
- 画面のちょっとした変化に追従できない恐れ

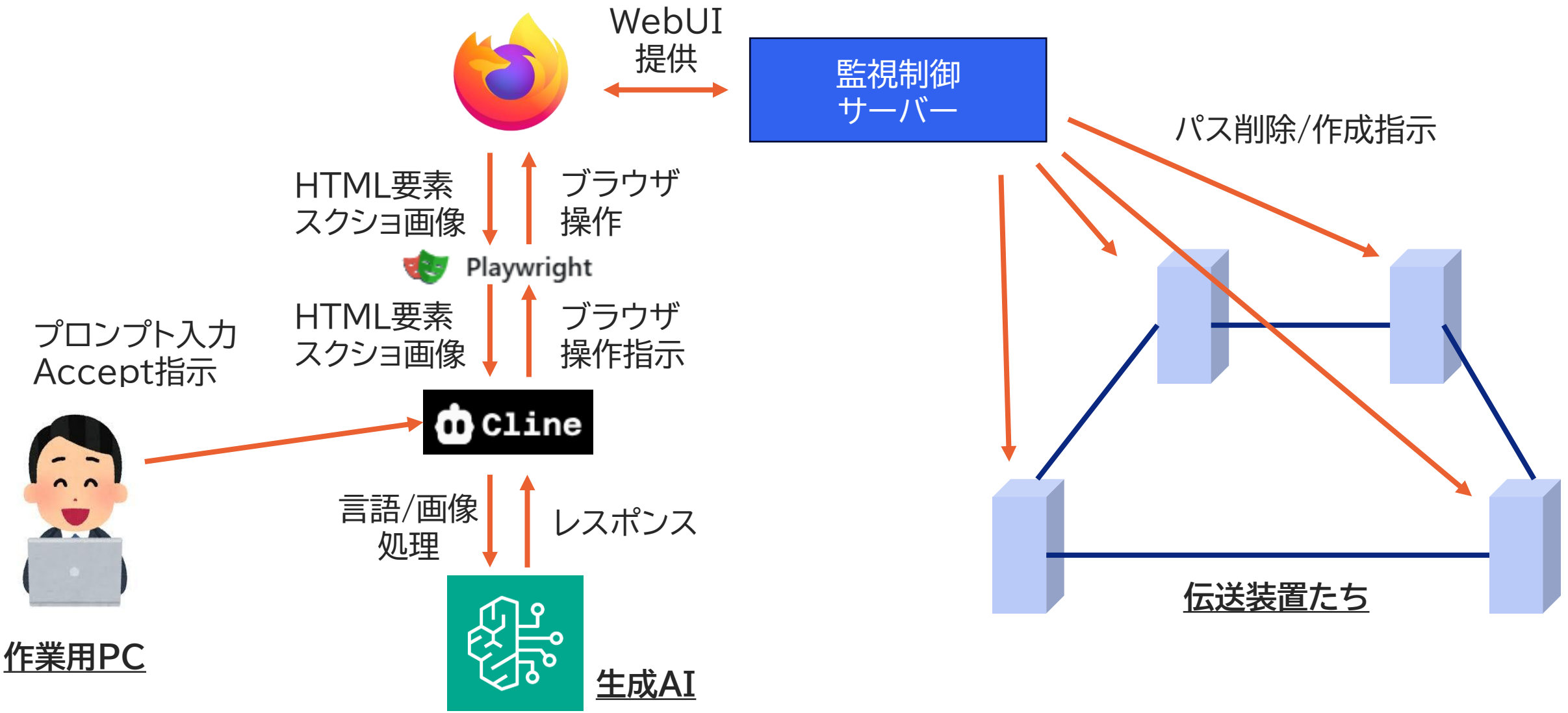
Selenium

- スクレイピングとかで利用されている
- 社内でもWebGUIしか持たないシステムの自動化に活用中
- がりがりPython等でコーディングが必要だれでも使える感じには遠い

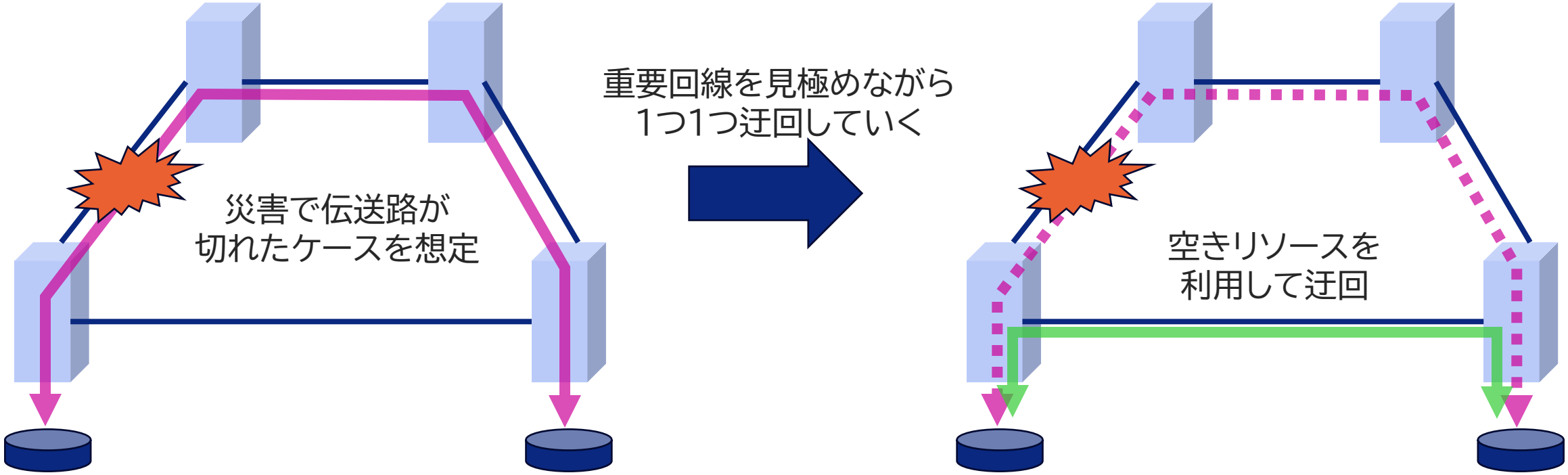
生成AI活用

- ネットワーク作業や保守への活用を模索中
  JANOG54/ネットワークオペレーションにおける生成AI技術の活用検討について
- 専用アプリの操作を覚える必要なく、逐次チャットで確認しながら、実施できそう
- なんか、Playwright MCP使うと、いい感じにブラウザ操作と連携できそう
- 画面スクショを解釈して動かすことができるので、変化に追従できそう
- 触ってみたい！

デモ環境



デモシナリオ －災害発生時の100Gパス迂回



- 通常時は伝送装置の自律切替等で迂回。今回は大規模災害時で自動救済できなくなってしまったケースを想定
- 普段から操作になれている人なら、すぐに実施可能。しかし、ベテランがたまたま非番ということも。
  手順書があっても慣れてない人は、「このパラメータどこに入れるんだっけ…？」「この手順書に書いてあるメニューどこにあるんだっけー？？」と、結構ドキドキしてしまう場面

不慣れな人の迷いを救ってあげられるシステムを目指す！

プロンプトを準備

■作業の概要 → 詳細作業ステップ → ざっくりパラメータを指定

> **Playwright MCPでブラウザを使用して伝送装置監視制御システムへ接続し、障害発生中のサービス経路迂回を実行します。以下のステップを踏んで実施してください。**

STEP 1: ログインページを開く

STEP 2: 迂回対象サービスの検索

1. Layersの中から「WDM」が選択されていることを確認する。ラジオボタンにて「WDM」以外が選択されている場合は「WDM」を選択してページ遷移を待つ
2. 「Services」タブをクリックし、その中に次の名前を含むサービスを検索してください。
    - 検索対象のサービス名: for Demo
3. 1つのパスが見つかった場合、そのパスの「Working Path」を表示して覚えておいてください。

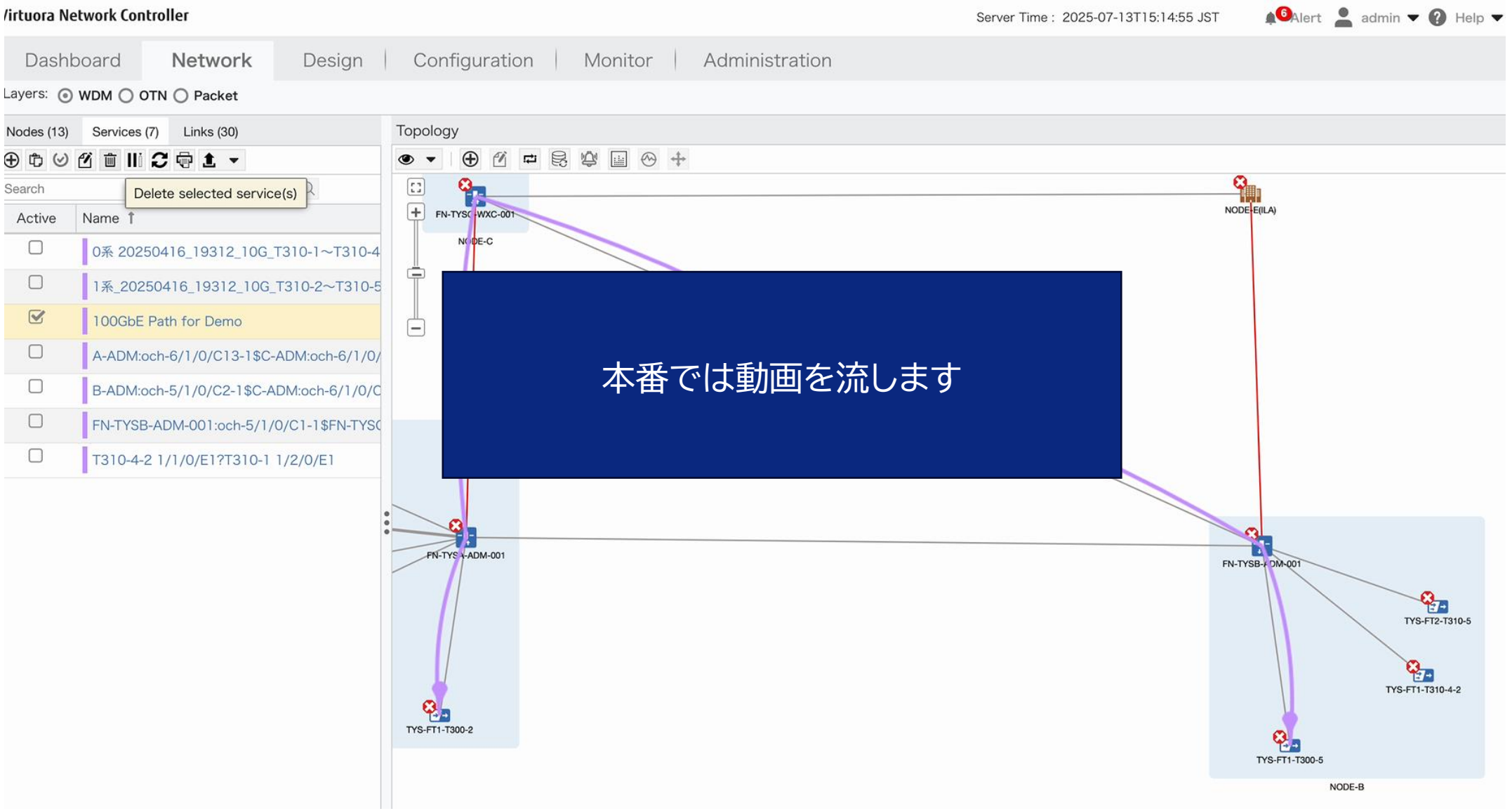
STEP 3: 迂回対象サービスの削除

1. Service一覧が表示されている画面に戻り、先程検索したサービス名が記載されている箇所の左側の
2. tabpanel Services の中のtoolbarにおける、5個目のボタンをホバーして、ツールチップの内容を表であった場合、クリックしてください。
3. Delete Serviceダイアログの中で、チェックボックスが次の状態になっていることを確認します。も
4. 削除対象のパスについて、次の情報を記憶しておいてください
    - Sourceのノード名
    - Targetのノード名
    - Source TPのポート番号
    - Target TPのポート番号

STEP 4: 迂回先サービスの作成

1. Servicesタブのツールバーの1番左側にある、ツールチップでCreate a serviceと表示されるボタンをクリックする
2. 開いたダイアログボックスのフィールドにそれぞれ以下を入力し、Nextをクリックする
    - Name: 100GbE Path for Demo (After Re-Route)
    - State: Active
    - Source
        - Node: 記憶しておいたSourceノード名
        - End Point: 記憶しておいたSource TPのポート番号
    - Target
        - Node: 記憶しておいたTargetノード名
        - End Point: 記憶しておいたTarget TPのポート番号
3. Route Constraintsタブに移動してResource Type "Links"とし、障害が起きているリンクであるLinkをExcludeします。具体的には以下の条件を満たすリンクをSelected Linksから選択し、"Add"ボタンを押してください。
    - TYSAとTYSCを結ぶリンク
4. Createボタンが有効になったら、Createをクリックします。
5. 作成されたサービスの状態を確認します。サービス詳細を表示させてください。

流してみた（デモ動画）



やってみてわかったこと

迷いは解消された？

- ちゃんとした**パラメシートの準備不要でうれしい**
- 実際の画面表示と表記ゆれがあるプロンプトでも、**雰囲気を察してくれてなんとかなる**
- **たまに間違える**けど、**プロンプトの工夫次第でもっと精度上がりそう**
- **詳細作業ログが残せる**

発展性は？

- 日本語さえできれば操作できる。他ベンダの**GUI操作自動化も楽に実現できそう**
- ベテランが作業する方が早い**属人化解消や覚えることを減らす方向での改善としてはアリ**
- 作業時間短縮や自律化が目的なら、CLI/API操作（NETCONF、T-API）や自律分散プロトコル（GMPLS等）を検証・導入したほうが良さげ

伝送装置の自動化の進め方、方向性はどのようにお考えでしょうか？？

まとめ・今後の予定

■ やはり原点は**コミュニケーションと痛感**した
- ユーザーヒアリングや現場との対話を重視したことで**課題発見と解決に繋がった**
- 1→n展開には困難もあったが、道筋を見つけ、**OJTやGUI整備により新規ユーザーも短期間で利用開始可能に**
- **AI支援を活用**したコマンド差分やGUI操作の対応も進行中
- 今後は機能拡充とユーザー拡大を目指す

■**是非社外の方々にも使っていただきたい**

ディスカッションしたいこと

ユーザーを拡大する際に、考慮すべきことは他にあるか？
OJT、その他…？
ユーザーとの対話はどのようにしているか？
ストーリーマッピング、早期トライアル、生成AIもアリ？

新しい仕組みに対する現場の抵抗感を下げるために、生成AIは心理的なハードルを下げる手段として有効なのか？

GUIは今後も残るか？
使っていきたいか？
GUIの自動化はどうする？

プラットフォーム横展開時、どこまで「共通化」すべきで、どこまで「個別最適」を許容するべきか？

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。

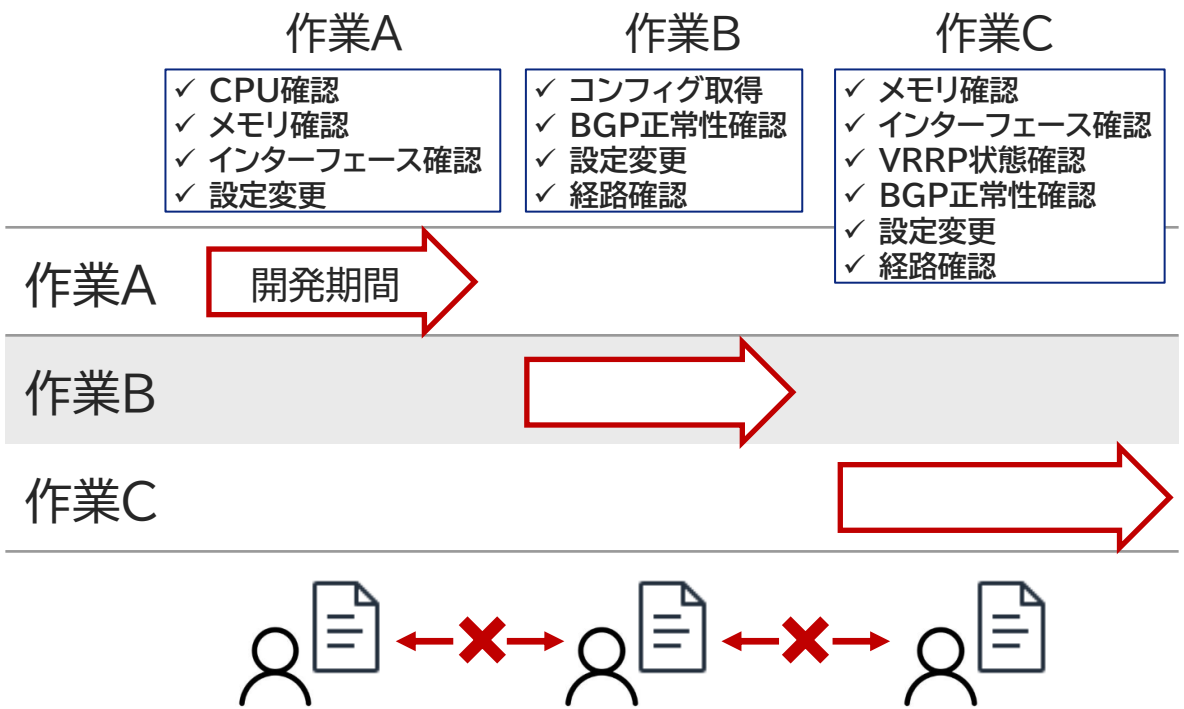
KDDI VISION 2030

【参考】自動化支援プラットフォームでの自動化構想

■ 自動化コードを細かく分け、共通部分は使い回しできるようにすることで、開発工数を持続的に減らす。また共通部分は社内組織横断で共有可能

既存の自動化

◆ 特定の作業でしか使えない
◆ 似たような自動処理を個々で“1”から開発



自動化支援プラットフォーム構想

✓ 他作業への使い回しが可能
✓ コード/ノウハウの蓄積・共有