5Gコアの試験プロセス自動化戦略/手法論の立て方

～テストの担当者ですが、残業は嫌なので手順を自動化しようと思います～

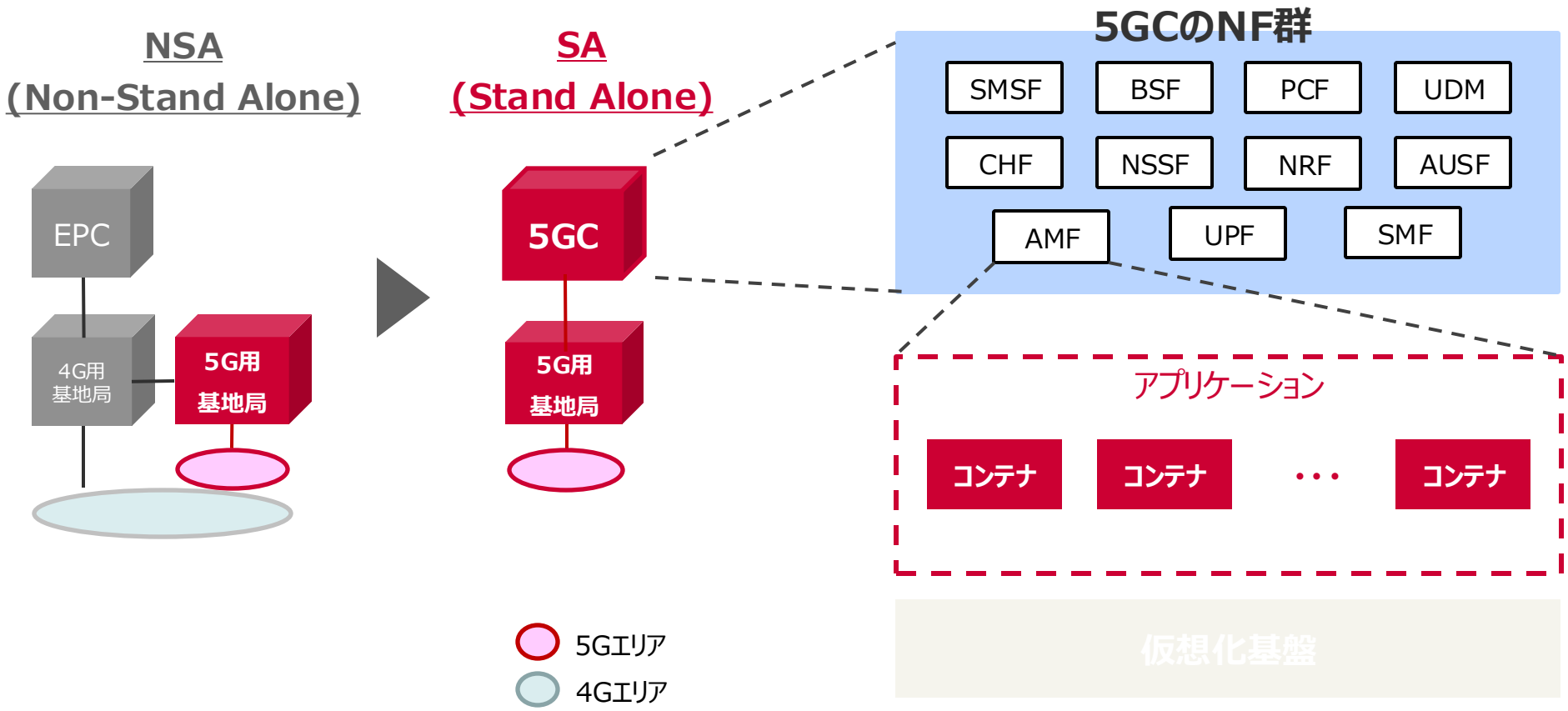
NTTドコモ
小原 啓希　白水 孝始

2025/07/31

はじめに
5Gコア(5GC)について
5GCにおける試験の特性

5Gコア(5GC)について

- 5G通信において5G SA(Stand Alone)を実現するために5Gコア(5GC)を導入している
- 5GCはSBA(Service Based Architecture)を採用しており、AMFやSMFといった複数のNF(Network Function)で構成される
- 我々の5GCは仮想化基盤上で動作している



5GCにおける試験の特性

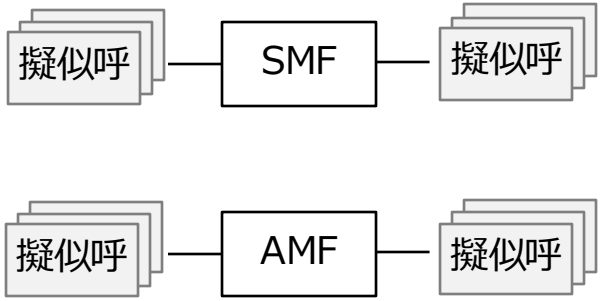
- 5GCは社会インフラを担っていることから膨大なトラヒックに対してパフォーマンスの維持・安定した運用が求められ、それらを確認/検証するために性能試験や長安(長期安定化)試験を始めとした複数種類の試験を実施する
- 特に性能試験や長安試験はNF毎に実施し、対向するNFはシミュレータ(擬似呼)を用いる

<5GCで実施する試験例>

・機能試験
・対向試験
・運用手順検証
・長安試験
・性能試験
-バースト試験
-スパイク試験
等

実トラヒック相当の背景負荷をかけた状態での試験を多数実施

性能試験や長安試験はNF毎に実施
対向するNFは擬似呼を使用



実トラヒックを擬似することから多数の擬似呼が必要

課題と実施方針

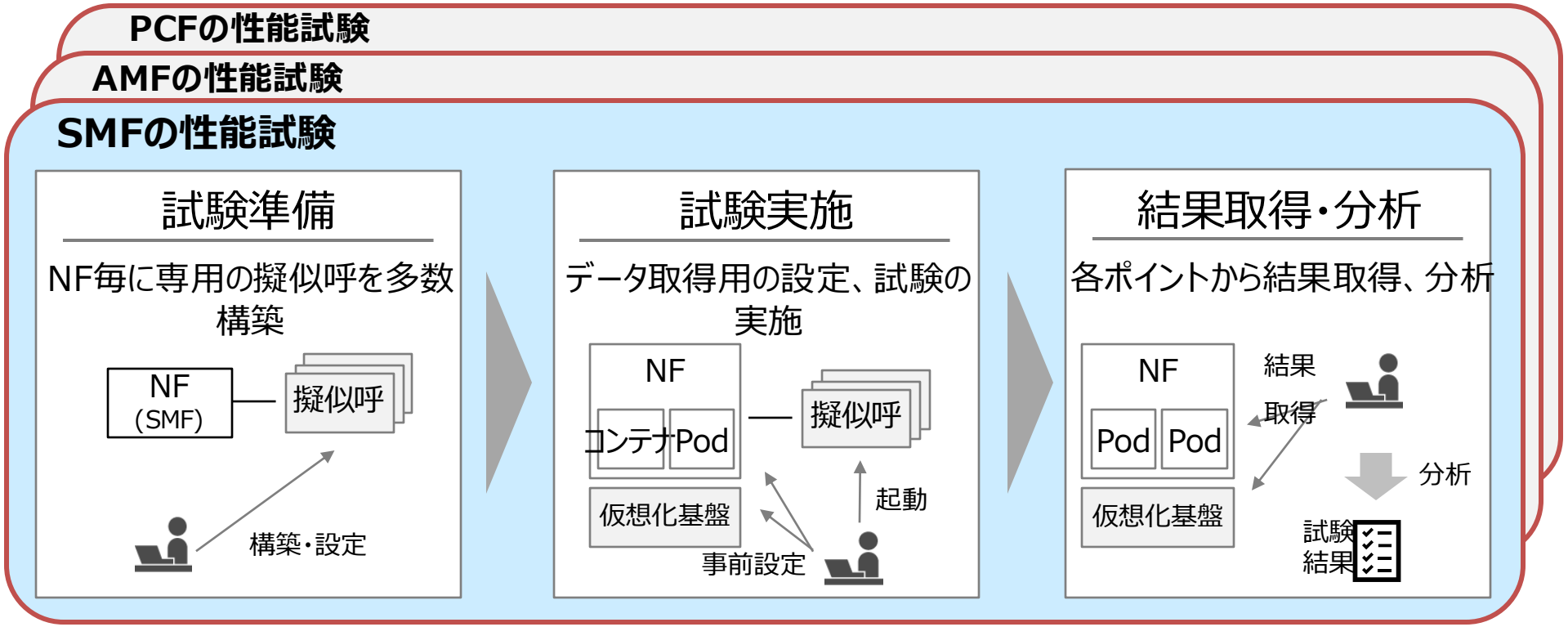
課題

実施方針

課題：試験の効率的な実施に向けた課題

- 試験は試験実施に加え、環境準備や試験後の結果取得・分析などの様々な項目によって成り立っている
- それぞれの試験/作業を明確化・分解し、自動化すべき項目を抽出し、自動化プロセスに置き換えることが課題であり、作業にかかる時間を効率化することが目的である

＜性能試験を実施する際の項目例＞



各試験/作業を明確化・分解し、自動化すべき項目の抽出が重要

実施方針：今回の手法について

- 業務の膨大な項目の中から、フローを改善すべきものを効率的に抽出する手法が必要
- 5GCの試験の課題に対しては”マトリクス分析”と”VSM分析”の2つの手法で分析を実施

手法１：マトリクス分析

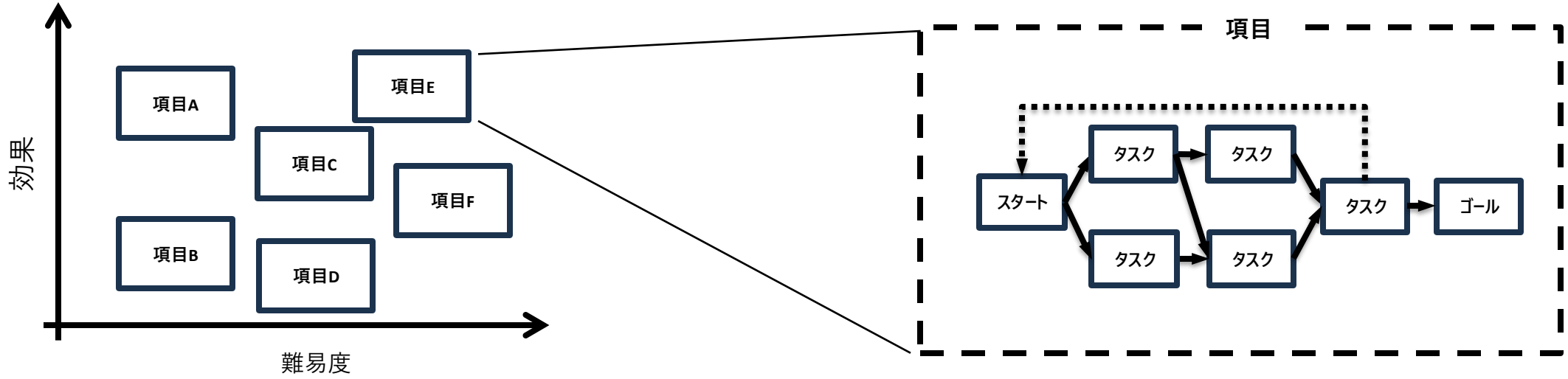
・特定の変数や要素の関係を視覚化し、評価・比較する手法

・さまざまな分野で広く利用されており、データの整理や洞察を得るために非常に効果的

手法２：VSM分析

・作業の流れを視覚化して、製品やサービスの提供に関するプロセスを理解し、改善するための手法

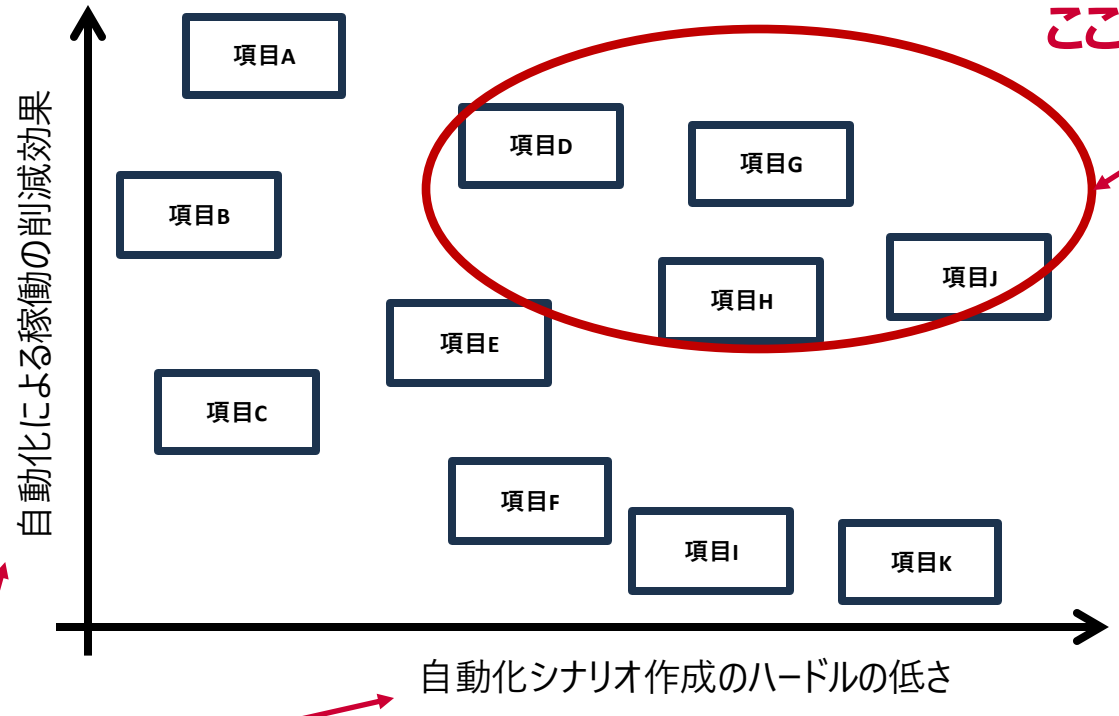
・一般的に無駄を削減し、効率を向上させることを目的として行われる



実施方針：マトリクス分析

- 各タスクを2つの評価軸で見積り、優先度の高い項目を抽出する
- 私たちのチームでは以下を軸に検討(チーム目的に合わせて設定するのをおススメします)
  - 自動化シナリオ作成のハードルの低さ(着手のしやすさ)
  - 自動化稼動の削減効果(事業インパクト)

時間をかけずにおおよそで見積もる



ここの項目の優先度が高い

大まかな流れ

① 自動化したい領域の関係者を集める

② **目的に合わせて評価軸を設定**する

③ 項目を各人で洗い出す(ブレスト)

④ 出てきた項目をグループ分け

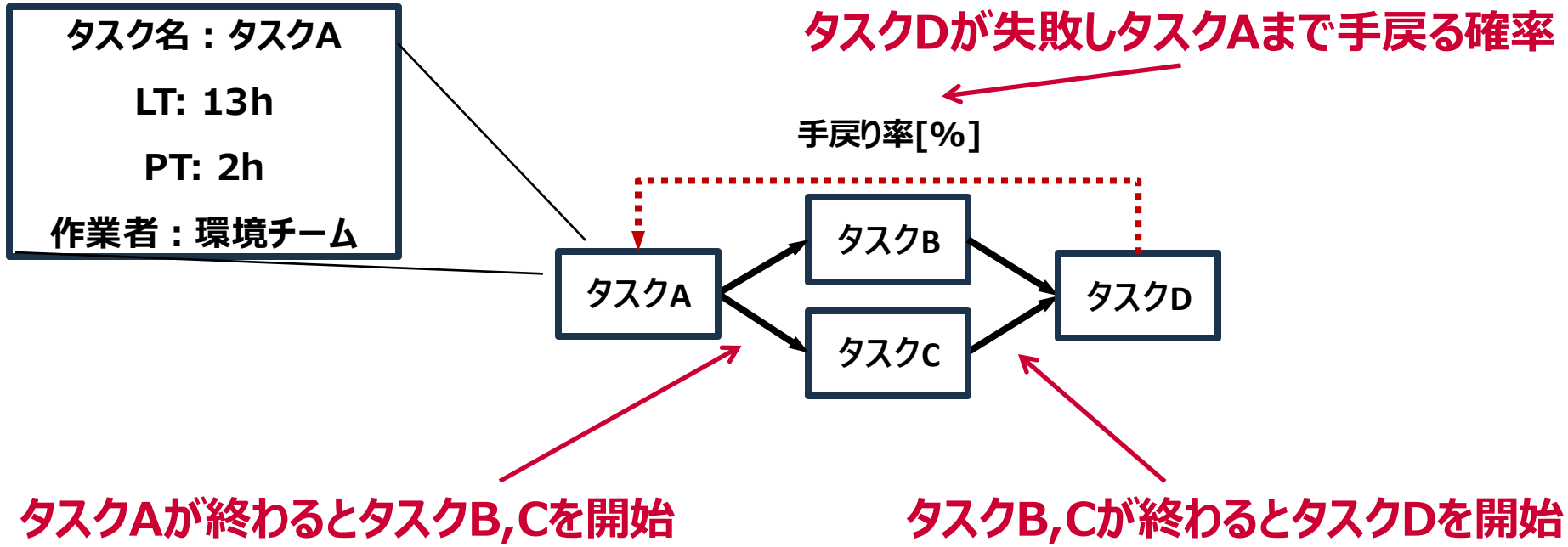
⑤ グルーピングされた項目をマッピング**(時間を掛けずにおおよそで見積)**

⑥ **右上の項目が優先度が高い**

目的に合わせて評価軸を設定する

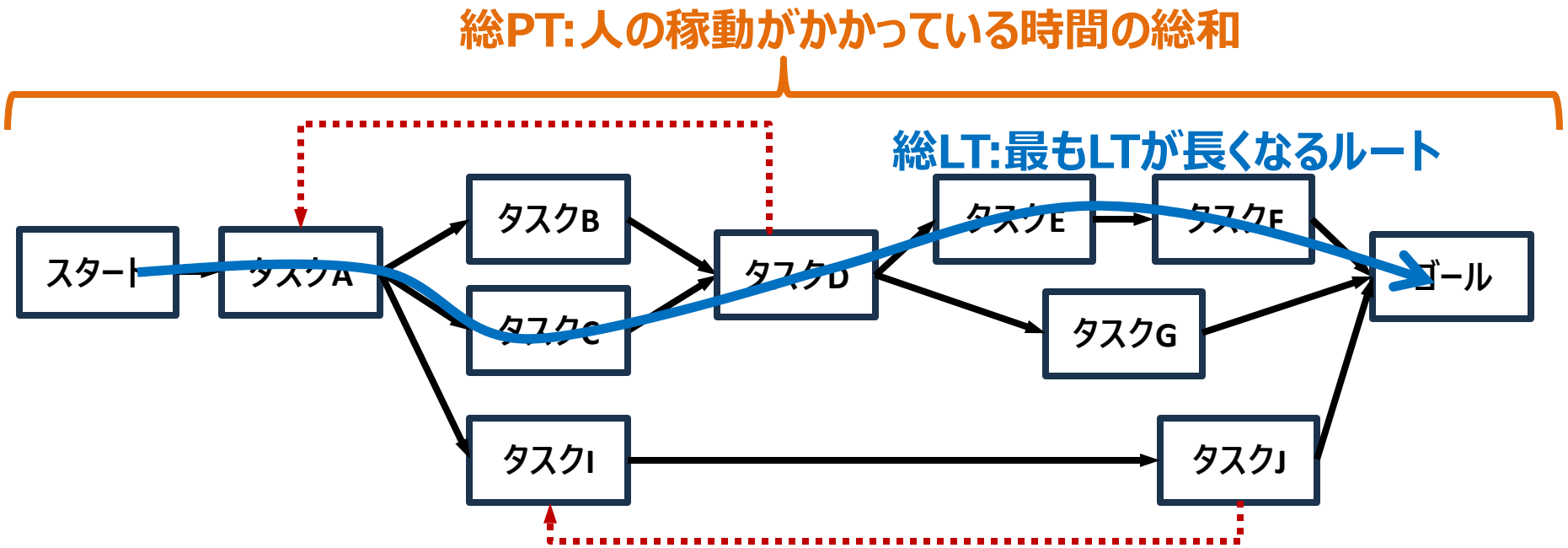
実施方針：VSM分析1/3

- 抽出した項目ごとに、現状の詳細なタスクを洗い出す
  - Lead Time(LT)：タスクの開始から終了までにかかる時間のことで、待ち時間も含める
  - Process Time(PT)： 実際に作業をしている時間のこと.待ち時間は含めないが、作業人数が2人以上の時は人数をかけた延べ時間
  - 手戻り率：タスクが失敗する確率



実施方針：VSM分析2/3

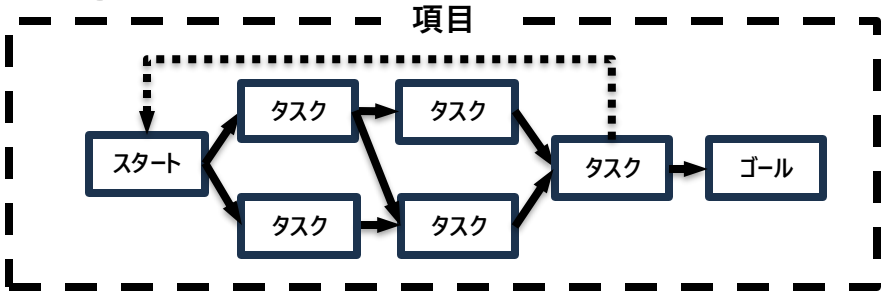
- 総PT, 総LT, 直行率を計算する
  - 総LT：最もLTが長くなるルート上のタスクのLTの総和(その項目に掛かっている時間)
  - 総PT：すべてのタスクのPTの総和(その項目で人の稼動がかかっている時間)
  - 直行率：すべてのタスクの成功率(1-手戻り率)の総積 (タスクが一度も失敗せず成功する確率)
- LTが長い場合は待ち時間が多い
- PTが長い場合は人手がかかっている



実施方針：VSM分析3/3

- As-Is(現状)の分析が終わったらTo-Be(理想)の形のVSM分析も行い、総PT, 総LT, 直行率を比較し改善効果を測る
- To-Beを作る際、As-Isのプロセスを考慮せずインプット/アウトプットから考える

As-Is

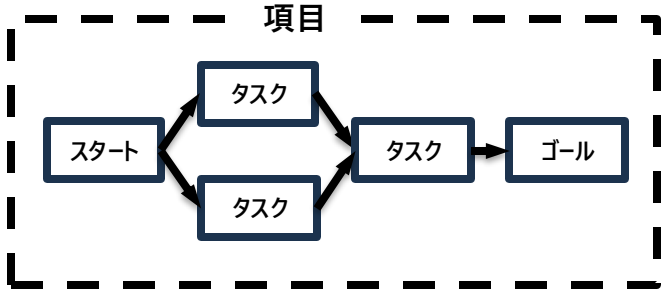


総PT：〇〇h

総LT：〇〇h

直行率：〇〇%

To-Be



総PT：〇〇h

総LT：〇〇h

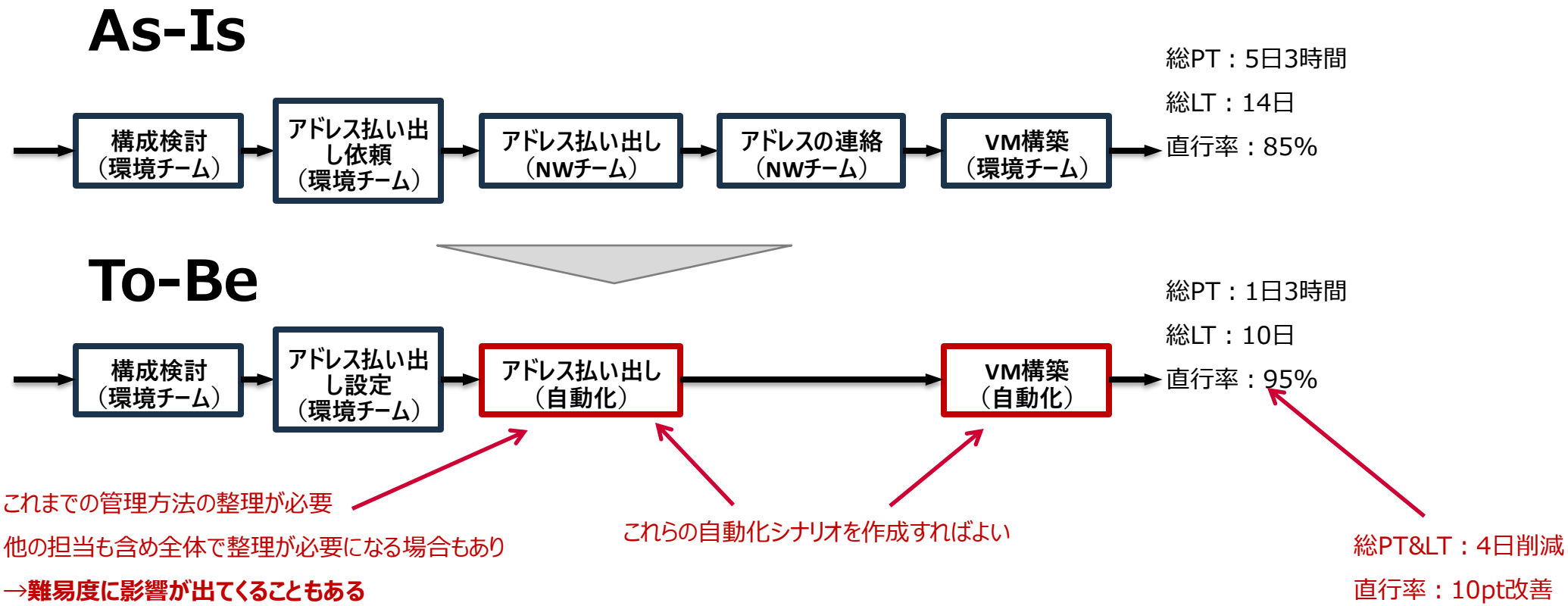
直行率：〇〇%

工数を1/10にするには？

極力、人の手を排除するには？

実施方針：再評価と改善実施

- 施策の効果の再評価
- 施策の難易度の再評価
- 整理項目の洗い出し
- 自動化シナリオの作成部分の洗い出し



各選定施策の概要と実施前分析
各選定施策の概要
実施前分析

各選定施策の概要

- マトリクス分析とVMS分析の手法を利用して，取り組みやすさと時間削減効果を評価し，自動化により改善すると見積もった施策は以下3つ

① 擬似呼VMの構築自動化
  - 擬似呼で使用するVMをまとめて構築することで工程簡素化&サービス化

② 性能試験の結果確認自動化
  - 性能試験における全ホストのCPU使用率測定&結果収集&グラフ化

③ 擬似呼ダイナミック化
  - 擬似呼のリソースプールから需要に応じて擬似呼を自動デプロイ

各選定施策の事前分析

- 抽出した3つの施策に対して自動化の提案を行い、AsIsとToBeはのPT,LTを算出
- 各施策の概要と結果は次頁以降

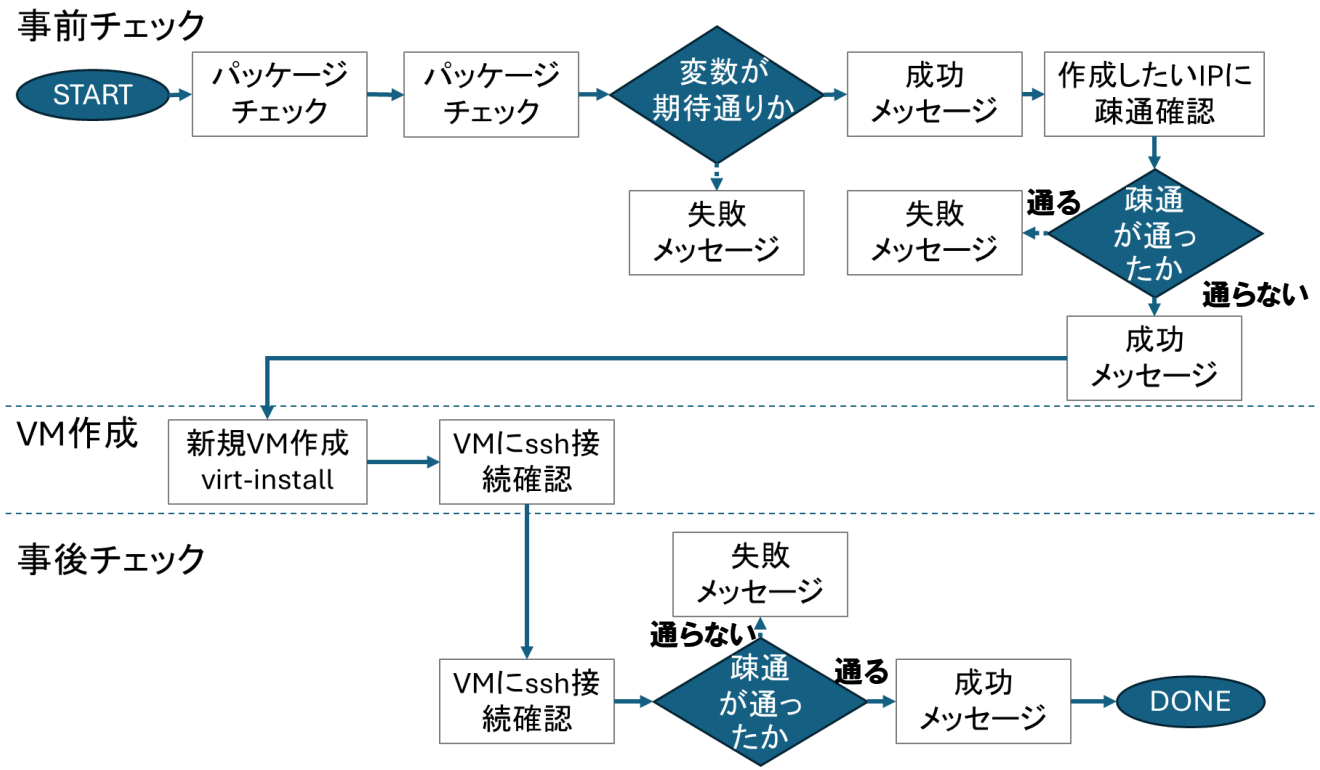
	施策1		施策2		施策3	
時間	LT	PT	LT	PT	LT	PT
AsIs	2h	2h	1w32h	1w24h	3m1w	22.5h
ToBe	0.1h	0.1h	21h	14h	21h	19.5h

※mはmonth(月),wはweek(週),hはhour(時間)を示す

各施策の詳細と結果
施策1,施策2,施策3
考察

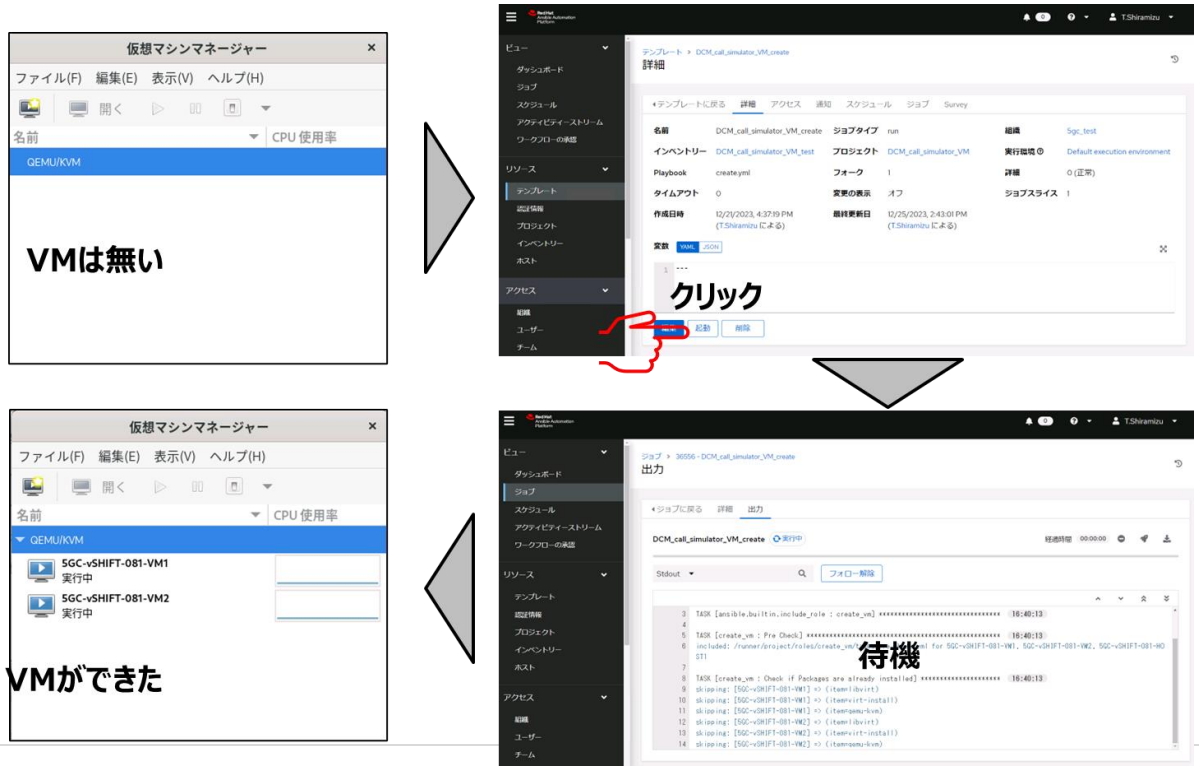
概要（施策1）：VM構築自動化

- VM構築をAnsibleで自動化
- 冪等性担保のために事前チェック及び事後チェックまで含めてPlaybook化



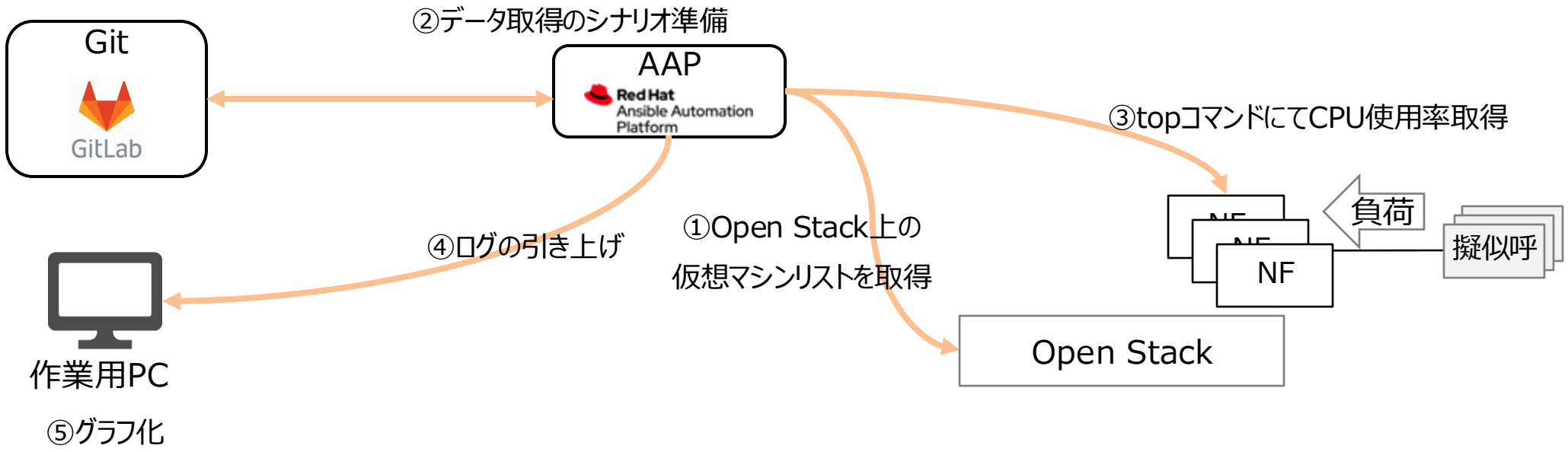
結果（施策1）：VM構築自動化

- PlaybookをRole化してAnsible Automation Platform(AAP)で管理
- Ansibleを知らない運用者でもGUI画面上で1クリックし実行可能とするサービス化を実施
- 総PTが2時間の作業を5分以内に実行可能になった
- サービス化によって登場人物が減ることにより調整稼動も削減



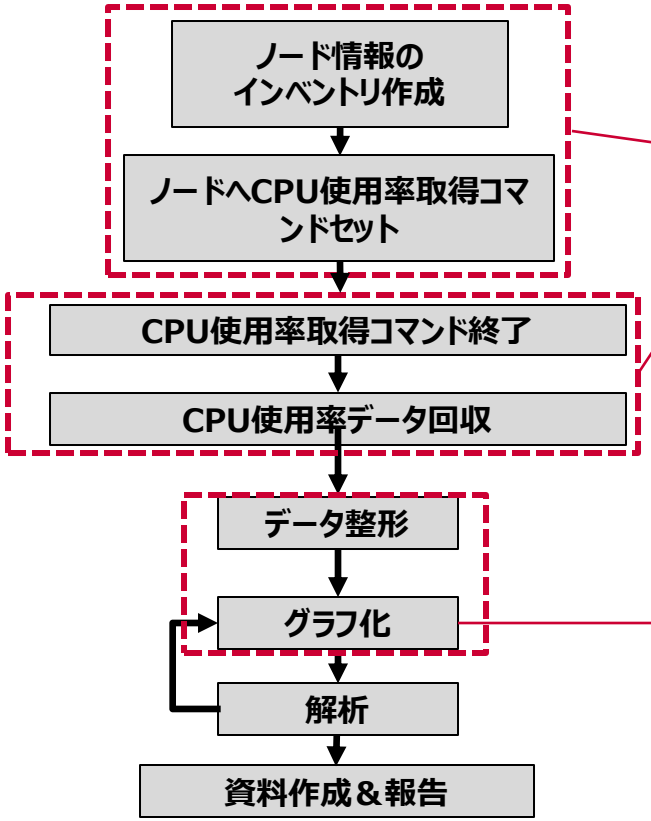
概要（施策2）：性能試験の結果確認自動化

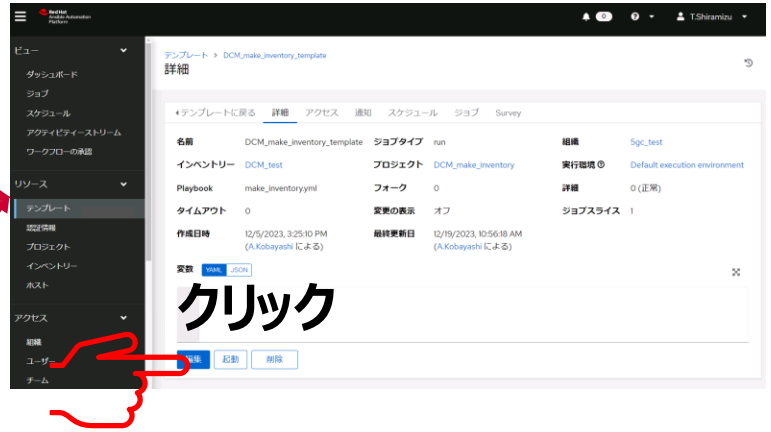
- NFの性能試験を行う際のCPU使用率取得とグラフ化を自動化
- 以下手順をAsiblePlaybookにて自動化
  - ①Open Stack上の仮想マシンリストを取得
  - ②データ取得のシナリオ準備
  - ③topコマンドにてCPU使用率取得
  - ④ログの引き上げ
  - ⑤グラフ化

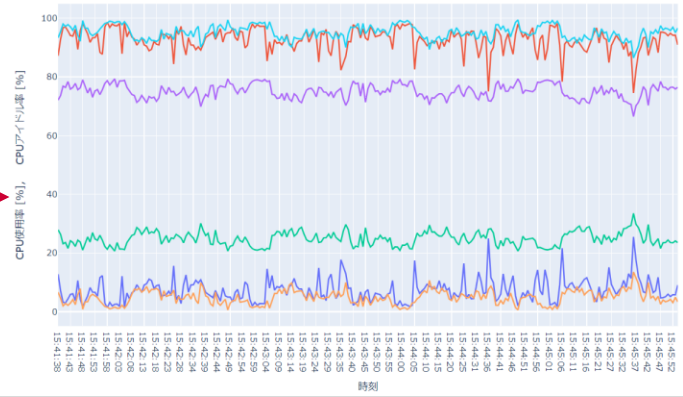


結果（施策2）：性能試験の結果確認自動化

- 施策1と同様にAnsible実行をサービス化し、1week以上の削減を達成
  - LT　1w8h　→　0.5h
  - PT　1w5h　→　0.5h
- データ成型やグラフ化はPythonによる自動化を実施しています
  - 開発環境と資料作成環境が分かれているため



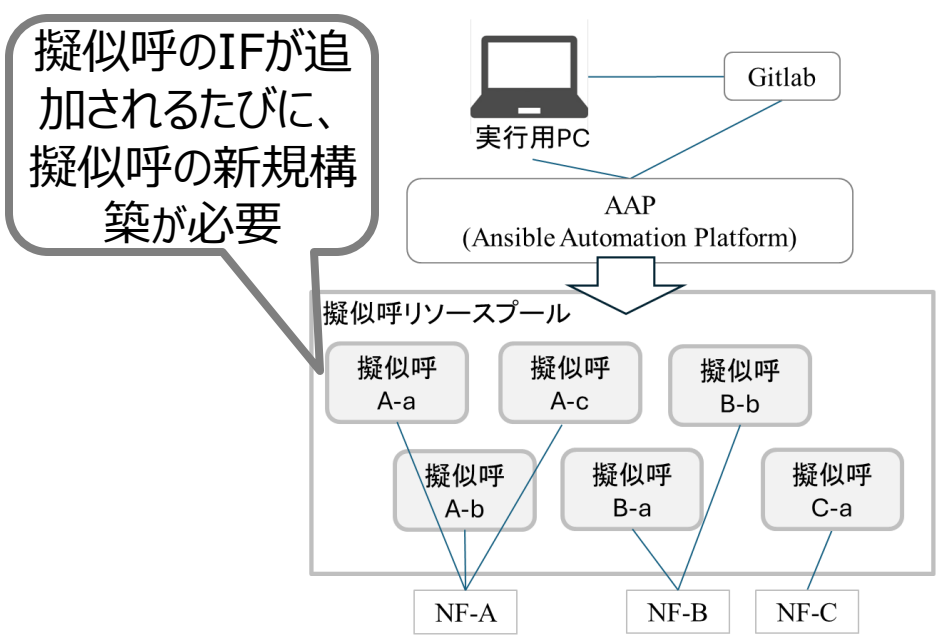




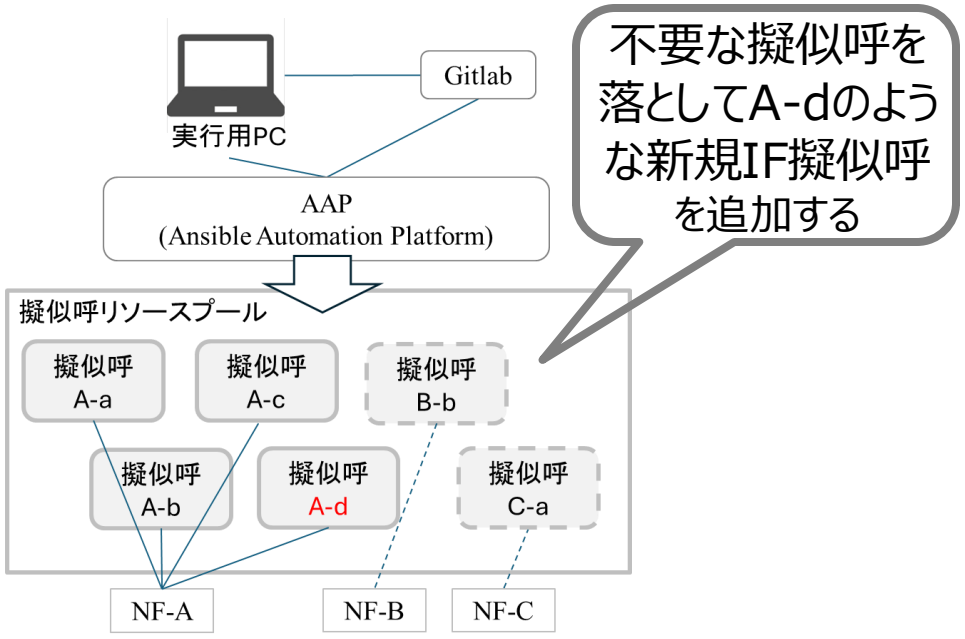
グラフ化後のイメージ

概要（施策3）：擬似呼ダイナミック化

- 性能試験の準備にあたる擬似呼の構築にて，ダイナミックインフラストラクチャの思想を元に実施
- (a)のように新規擬似呼構築が新規で必要だった構成を (b)の構成にすることで物理工事などを実施しないで擬似呼を追加する施策



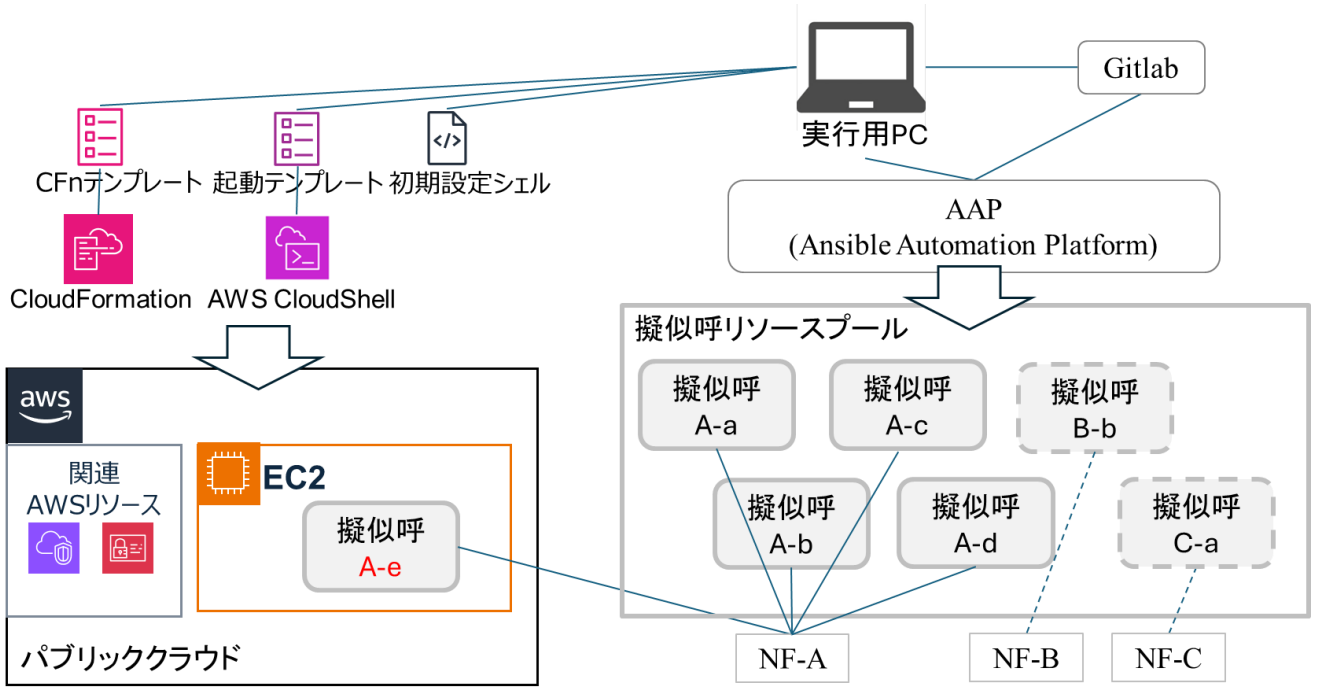
(a)提案3実施前の構成イメージ



(b)提案3実施後の構成イメージ

概要（施策3）：擬似呼ダイナミック化

- オンプレミスの環境だけではなくパブリッククラウドを利用することも実施
- さらに柔軟な構成で擬似呼を準備可能にした
- AWS利用の場合は負荷を掛けるNFや擬似呼の機能によって，環境間のトラヒック量が変わるため，コスト観点の考慮が必要

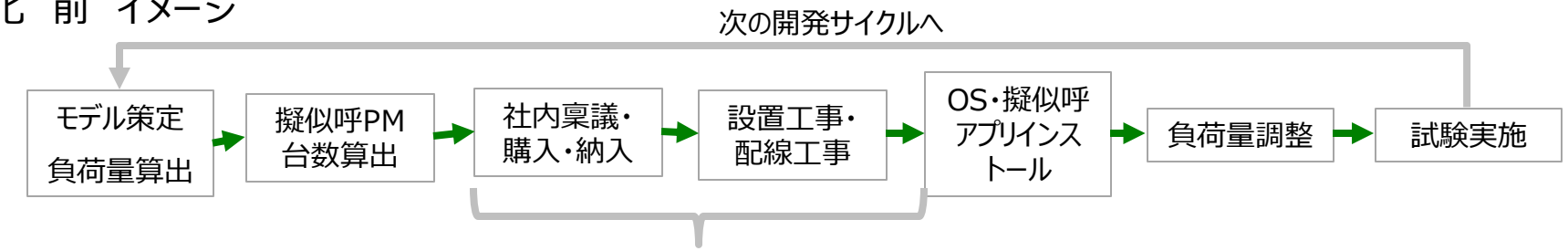


結果（施策3）：擬似呼ダイナミック化

- LTが3m1wから数日で実施できるようになった
- 元々多くのLTの原因となっていた納入や物理工事のLTをなくすことができたのが効果として大きい
  - LTを小さくすることで限られた環境準備・構築/試験期間の融通が効くようになった

擬似呼

ダイナミック化　前　イメージ



この部分の検討稼動/物理リソースの確保/作業に時間がかかっていた

擬似呼を準備するこの部分のLTは3m1w4h(PTは36h)

LT:Lead Time 待ちも含む実際にかかった期間

PT:Process Time タスクの実際の作業時間

擬似呼

ダイナミック化　後　イメージ

ダイナミック化で流れそのものを変更することでスケジュール短縮/稼働削減

4.考察

- 施策1及び施策3では分析を行った結果とほぼ期待する通りの結果が得られた
- 施策2ではToBeよりもAAPのサービス化によるGUI操作で想定以上の時間短縮効果が得られたと考察

	施策1		施策2		施策3	
時間	LT	PT	LT	PT	LT	PT
AsIs	2h	2h	1w8h	1w5h	3m1w	22.5h
ToBe	0.1h	0.1h	2h	2h	21h	19.5h
実測	0.1h	0.1h	0.5h	0.5h	19h	15.5h
作成稼働(参考)	3w	10h	2m	1w	6m	-

※mはmonth(月),wはweek(週),hはhour(時間)を示す

苦労したことや体制の話
　苦労したこと
　体制の話
　今後の展望
まとめ

苦労したこととそのアンサー

- もともと試験費を削減する目的で自動化という手段を実施
  - 進めてると自動化を目的にしがち
  - 単純な作業を自動化する行為だけでは費用対効果が出てこなくなる
  - 自分たちによる自動化内製化に着手(施策1,2は完全内製化にて実施)
- 内製化だとすぐに費用対効果を出すのが難しい…だけど内製化は必要
  - PTで金額概算値、LTでスケジュール効果を算出して必要性を訴えていく
  - 実際に2021に自動化/内製化の取り組みを始めて今年で4年(足掛けも加えると5年)になりました…
- 一足飛びにはできないので、ロードマップを5カ年くらいで作って、定期的に見直していく取り組みが必要
  - もちろん元々の体制やスキルセットで状況は異なるので、そこは各体制で時期や取り組みをアレンジしてみてください

草の根活動や体制の話

- 元々内製化を行う環境が無かったが、手を動かす環境が出来た
  - 自由に使って良いサーバを余った部署からもらってきて、施策1を実施
  - その他にも現在の内製化では様々な環境を活用中※JANOG55 LT プリスクール構想
  - (ShowNet2025のテスト環境はその一環でした)
- 時間を無理やり作った
  - 最初は週2時間、該当メンバでがっつりモブプロの形式で開発を進めた(ほかの業務もそれぞれあるのでこの時間は集中して行う必要がありました。。。)
- 人が居なかった
  - 例えば当時2022の施策時は手を動かすメンバは4名
  - 今は2ライン進めていて、約20名

周囲のマインドセットも変わってきた

まとめ

- 5GCの試験において、フレームワークを用いてプロセス改善や自動化/内製化を行った
- プロセス改善の事例と取り組んでみた結果、PT,LT削減に効果が出た
- 実際には苦労したこと/草の根活動もあったが軌道に乗ってくるとチーム全体の動きが変わってくる

参考