



PSSOメソッドガイドブック

～Exastroでシステム構築・運用を効率化する～

目次

はじめに

システム構築・運用の効率化の全体像

自動化の事前準備

Step 1：構成情報の一元管理

Step 2：自動実行の実現

Step 3：構成情報と自動実行の相互連携

自動化されたSIの実施

効果の事例と見積りの観点

自動化後のプロセスと成果物の変更点

まとめ

はじめに



本書の目的

現場のITエンジニアは、非効率なシステム構築・運用に苦しんでいます。このような状況を改善するためには効率化が必要ですが、どのように取り掛かればよいか、迷われている方も多いと思います。

本書では、オンプレミスの環境で、どのような障壁を取り払って、自動化/効率化の準備を進めていくのかを、以下の3つのステップ(通称：**PSSOメソッド**)で分かりやすく解説します。

Step 1：構成情報の一元管理

Step 2：自動実行の実現

Step 3：構成情報と自動実行の相互連携

また、自動化/効率化の見積もりを行えるようにするために、自動化後のプロセスと成果物の変化をフェーズごとに整理しています。

PSSOメソッドとは？

PSSO (Procedures for Streamlining System Operation) メソッドとは設計・作業準備・作業実施の課題を解決し、従来の「手作業でのシステム構築/運用」を「自動化されたシステム構築/運用」へ移行させる自動化ステップのことです。



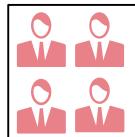
登場人物について

説明の便宜上、関係者について以下の表現を用います。



開発・構築チーム

- システム構築を担当するチームを総称して「構築チーム」と呼ぶことにします。実際のプロジェクトでは、業務担当やインフラ担当などの複数のチームがあることが多いです。



運用チーム

- 稼働中のシステムの運用を担当するチームを「運用チーム」と呼ぶことにします。



各チームの代表者

- チーム間での情報共有や調整を行う、各チームの代表者たちです。



システム構築・運用の効率化の全体像

自動化におけるAS-ISとTO BE

Step 1～3を実施することで、システムの自動構築・運用が可能になります。さらに、プロセスを変更することで、より効率的な自動化を実現できます。

TO-BE

自動化されたシステム構築/運用

AS-IS



自動化の事前準備 (Step 1、Step 2、Step 3)



自動化されたSIの実施 (プロセスと成果物の変更)

手作業でのシステム構築/運用

システム構築・運用に携わるITエンジニアの現場の「苦」



- チーム間での構成情報の共有に遅延やミスが発生する
- データの二重管理や、設定項目名の揺れが設計ミスにつながる
- 複数チームでの開発により、設計書(帳票)の管理が煩雑化する
- 結果としてどれが最新の構成情報か分からぬ



- チーム間の作業順序が複雑で毎回タイムチャートを作成しては使い捨てる
- 作業ごとに手順書を作成/レビューしては使い捨てる
- 手順ごとに個別の設定値を埋め込んでいて、新機種／新OSを追加するごとに手順書のパターンが増える(マルチベンダー対応の障壁)



- 人手作業なので作業時間が一定でない
⇒チーム間で作業待ちが発生
- 人手作業なので人為ミスの懸念から逃れられない

設計・作業準備・作業実施の課題を、3ステップで解決



設計



作業準備



作業実施

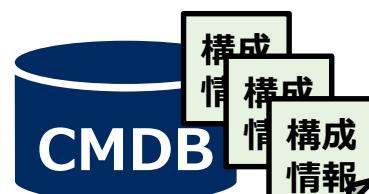
POINT

Step1がPSSOメソッドにおいて最も重要なステップです。

- チーム間での情報共有
- 従来の手順による複数の工程
- 結果として得られる情報
- チーム間のコミュニケーション不足
- 作業ごとに複数の手順を実行する
- 手順ごとに複数の書類を用意する
- に手書きの手帳やノート
- 人手作業が必要な工程
- ⇒チーム間での情報共有
- 人手作業による誤り

解決策

Step 1 構成情報の一元管理



解決策

Step 2 自動実行の実現



連携

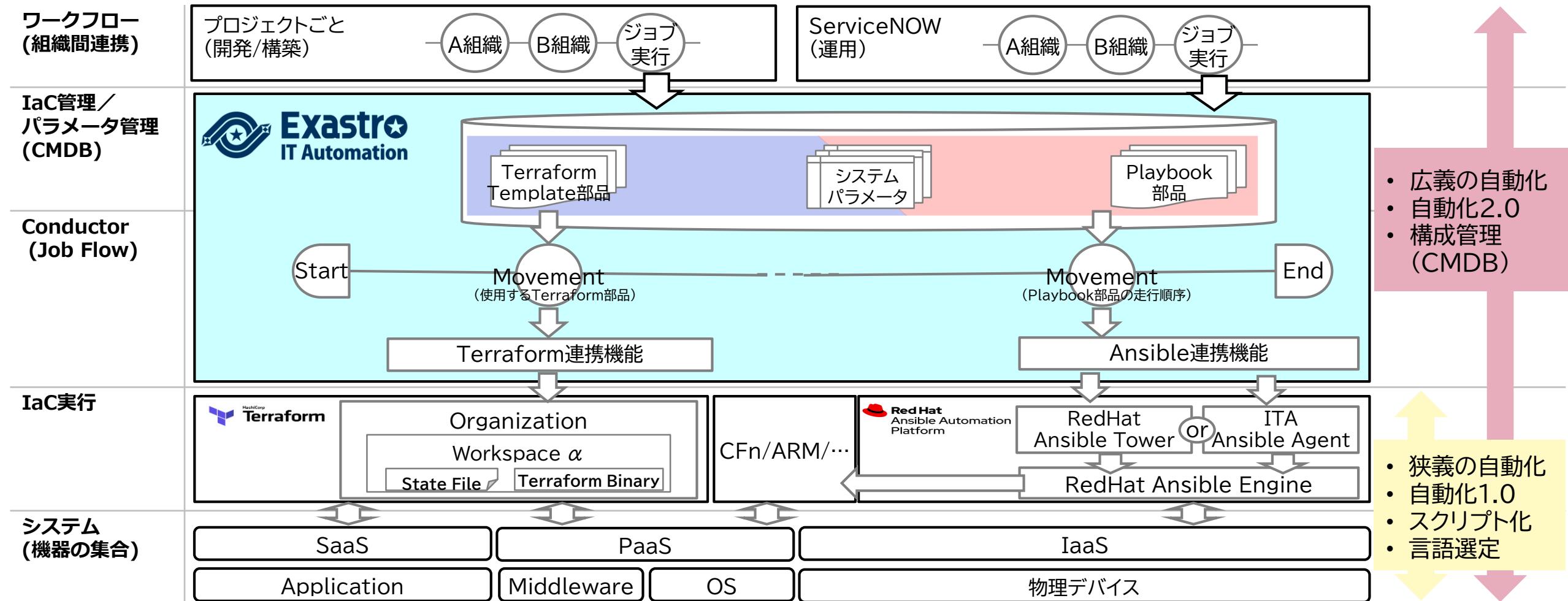
Step 3 一元管理と自動実行の連携



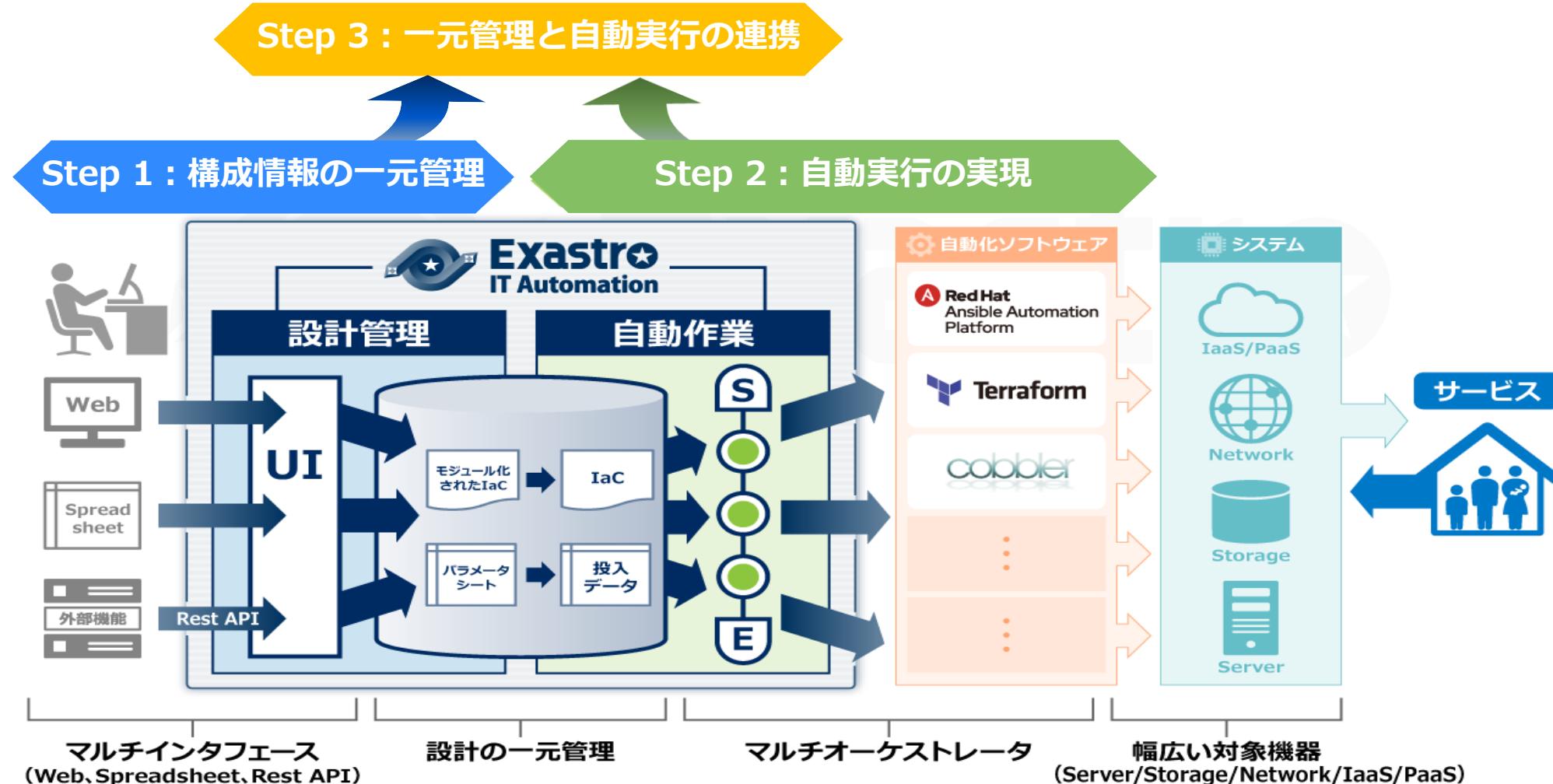
実行

自動化の範囲とステップの関係性

Step 2のみ実施 ⇒ 「狭義の自動化」
Step 1～Step 3を実施 ⇒ 「広義の自動化」



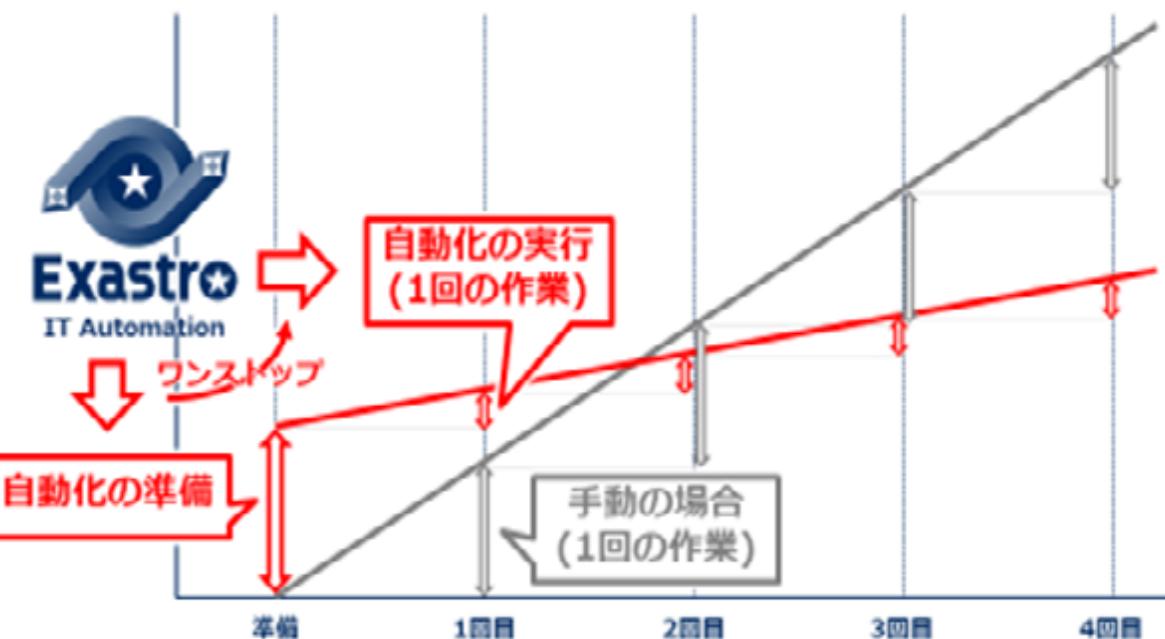
Exastro IT Automationは、3ステップでの解決を支援



自動化によって「QCD」および「タスクと成果物」が変化

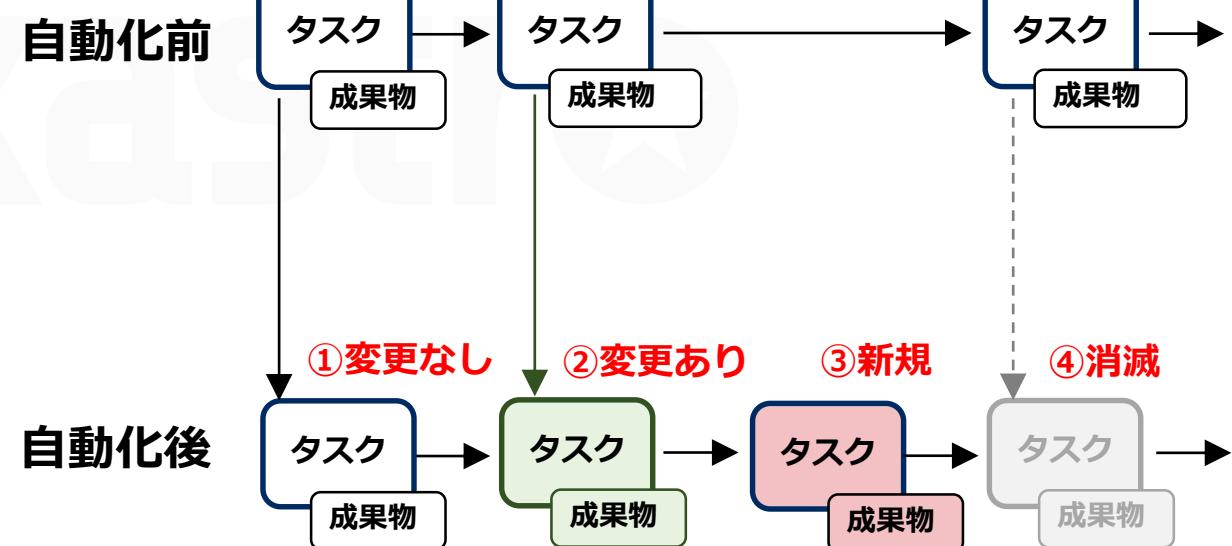
QCD (品質・コスト・納期)

手作業 ⇒ 自動実行により、QCDが改善



タスクと成果物

タスクと成果物の変化は、以下の4パターンに分類
→ ①変更なし、②変更あり、③新規、④消滅



自動化の事前準備

Step 1 : 構成情報の一元管理

Step 2 : 自動実行の実現

Step 3 : 一元管理と自動実行の連携



自動化の事前準備

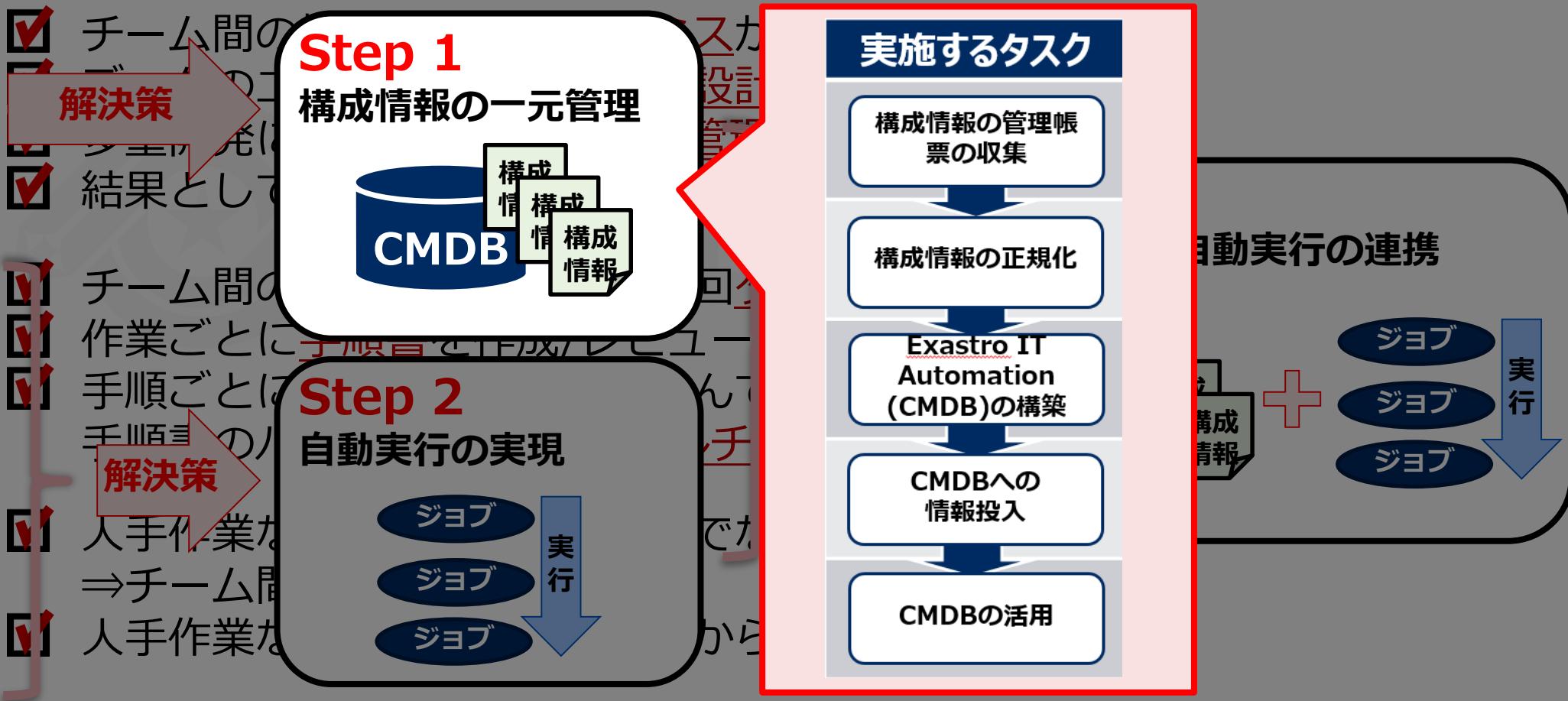
Step 1 : 構成情報の一元管理

Step 2 : 自動実行の実現

Step 3 : 一元管理と自動実行の連携



以降のスライドでは、Step 1の5つのタスクを説明していきます



Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

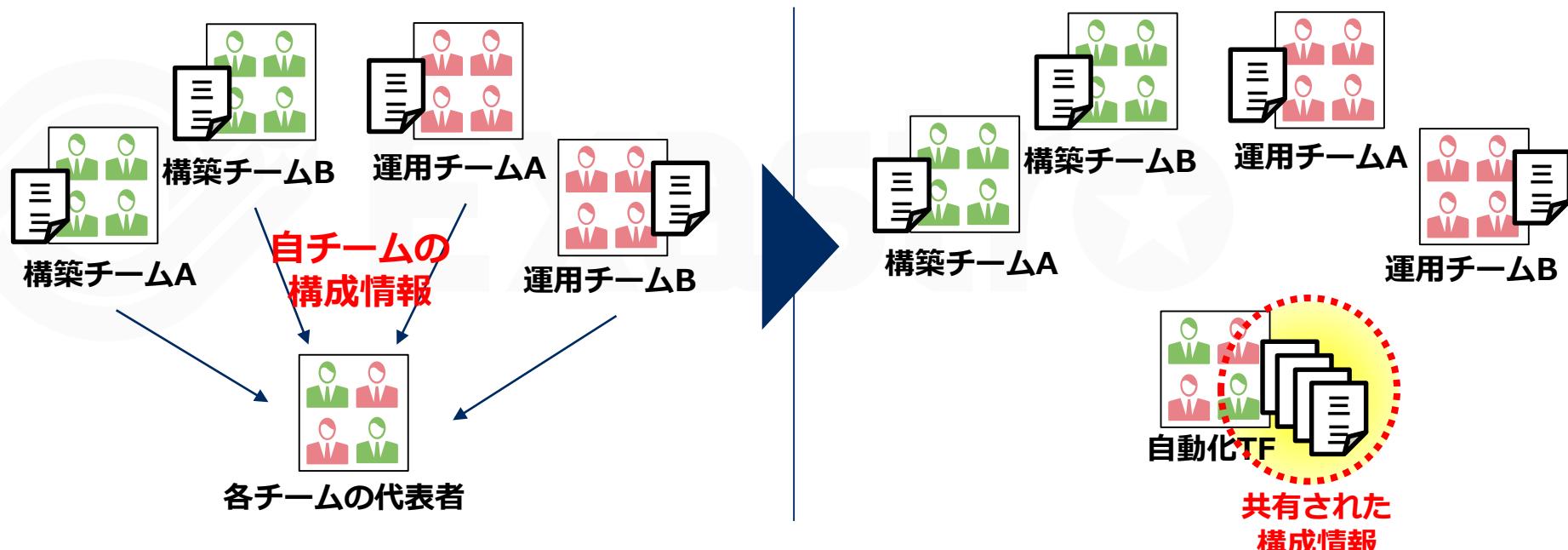
Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

タスクの説明

各チームの代表者は、自チームの構成情報を収集して、他のチームの代表者と共有する



POINT

- ① 目的を明確にして、一元管理の範囲を決める
- ② 既存の構成情報の管理方法は多様である
- ③ 事例～実際のプロジェクトで収集した構成情報

詳細
次頁

Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

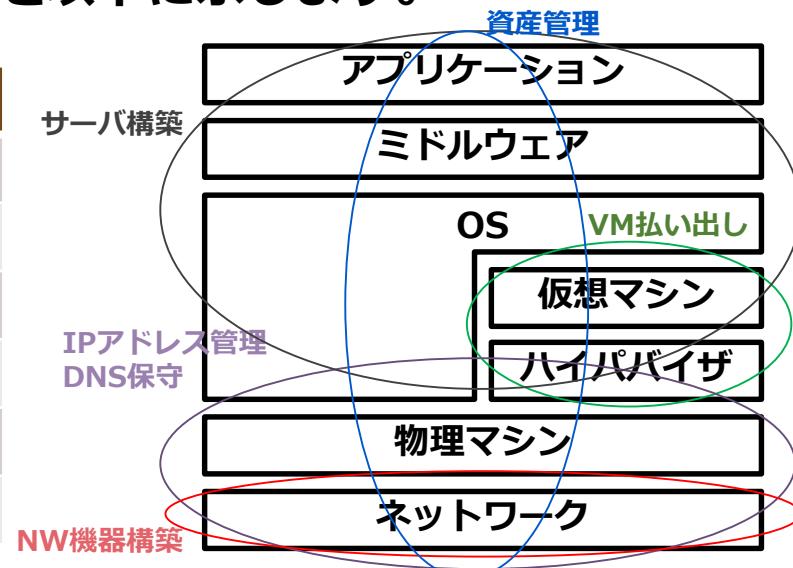
CMDBの活用

POINT

① 目的を明確にして、一元管理の範囲を決める

まずは、何が目的かを明確にする必要があります。目的が明確になると、収集する構成情報の範囲が決まってきます。具体例を以下に示します。

よくある目的の例	情報の範囲
例1) IPアドレス管理	IPアドレス、セグメント、など
例2) 資産管理	製造番号、ライセンス、など
例3) サーバ構築	IPアドレス、ホスト名、など
例4) NW機器構築	インターフェース数、VLAN、など
例5) VM払い出し	ハイパバイザ、VM名、など
例6) DNS保守	DNSサーバ、ドメイン名、など



目的が不明確だと、収集する情報が芋づる式に増えていったり、不要な情報(ゴミ)も収集してしまったりなど、問題が発生してしまいます。

また、目的が複数ある場合は優先度をつけて情報を収集し、順番にCMDBを構築していくことを推奨します。

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

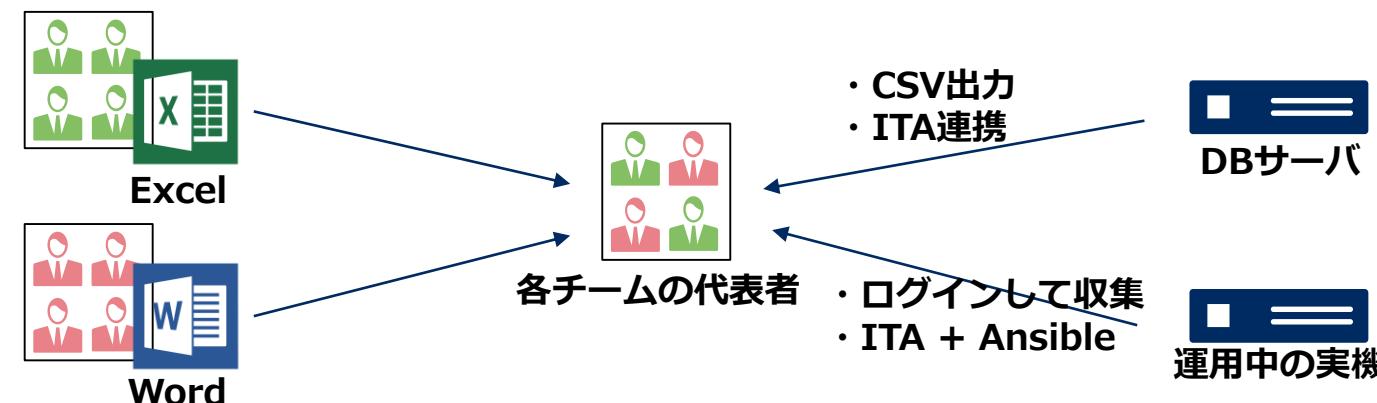
CMDBへの
情報投入

CMDBの活用

POINT

② 既存の構成情報の管理方法は多様である

多くのプロジェクトでは、構成情報はExcelやWordの形式で管理していますので、それらファイルを収集してください。もし、構成情報をDB等に格納している場合は、CSV形式でのダンプや、DBとExastro IT Automationの直接の連携を検討する必要があります。



プロジェクトによっては構成情報の帳票がなく、運用中の実機(サーバマシンなど)から直接収集する必要があります。この場合、Exastro IT AutomationとAnsibleを活用することで、比較的簡単に実機から情報収集することができます。

Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

POINT

③ 事例～実際のプロジェクトで収集した構成情報

ここで、サーバとネットワーク機器の構成管理を実現した事例を紹介します。このプロジェクトでは、サービス停止の影響範囲を容易に特定するために、以下の構成情報を各チームの代表者で共有しました。

チーム	収集した構成情報
サーバG	<ul style="list-style-type: none">・サーバ一覧・サーバに搭載したソフトウェア一覧
ネットワークG	<ul style="list-style-type: none">・IPアドresse一覧・ネットワーク機器一覧・ネットワーク経路一覧
ストレージG	<ul style="list-style-type: none">・パスリスト・ストレージ搭載ディスク一覧
運用監視G	<ul style="list-style-type: none">・メッセージ一覧
業務G	<ul style="list-style-type: none">・コンポーネント一覧・サーバコンポーネント一覧・通信条件一覧

本事例の詳細は、以下のURLで公開しています。

https://exastro-suite.github.io/it-automation-docs/case_ja.html

Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

タスクの説明

各チームの代表者は、重複の排除、項目名の統一、冗長な情報の分割などにより、収集した構成情報を表形式で正規化します。



収集した構成情報



各チームの代表者

正規化

- ・重複排除
- ・項目名の統一
- ・クレンジング
- ・etc

サーバ種別

サーバ種別
Webサーバ
DBサーバ
APサーバ

OS種別

OS種別
RHEL7
RHEL8
WinServer2019

サーバ機器一覧

サーバ種別	号機	ホスト名	OS種別
Webサーバ	#1	web001	WinServer2019
Webサーバ	#2	web002	RHEL8
APサーバ	#1	apsvr001	RHEL8

通信リスト(許可)

通信No.	FROM	プロトコル	TO
①	Webサーバ#1	https	tcp
②	APサーバ#1	ODBC	tcp

POINT

- ① 構成情報の仕分けを行う
- ② 構成情報の項目(表の列)を整理する

詳細
次頁

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

POINT

① 構成情報の仕分けを行う

各チームから集めた構成情報は、以下の観点で仕分けを行います。

- ① 自分のチームに閉じた構成情報か、他のチームと連携している構成情報か
他のチームと連携している情報がある場合、他の情報と分離します。これ
により、構成情報を共通化することができます。
- ② Exastro ITAの画面でプルダウンで選択させるかどうか
構成情報を登録する際、自由記述にするものと、プルダウンから選択式に
するものを分けます。選択式にするものは、その値を「マスター」として
登録します。
- ③ 構成情報の関係（リレーションの設計）
構成情報の関係（依存関係）を決めます。これは、構成情報の作成と登録の
順序に影響するため、重要です。例えば「サーバ一覧」を作成するためには、
その前に「OS種別」の作成と登録が必要になる、などです。

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

POINT

② 構成情報の項目(表の列)を整理する

最終的に、構成情報は表の形にまとめます。そのため表の「列」が何になるかを、以下の観点で整理していく必要があります。

① 構成情報の項目名(表の列名)の統一

同じ構成情報であっても、各チームで別の名前で管理している場合があります。例えば、同じIPアドレスであっても、サーバチームは「IP」として管理していたり、ネットワークチームは「ip_addr」として管理しているなどです。この場合は、チーム間で同じ名前を使うように調整して、共通の構成情報として切り出してください。

② 構成情報のグループ化

構成情報はグループ化したほうが可読性が高まる場合があります。例を挙げると、「IPアドレス」と「ポート番号」は、「接続情報」としてグループ化すると、可読性や保守性が高まります。

Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

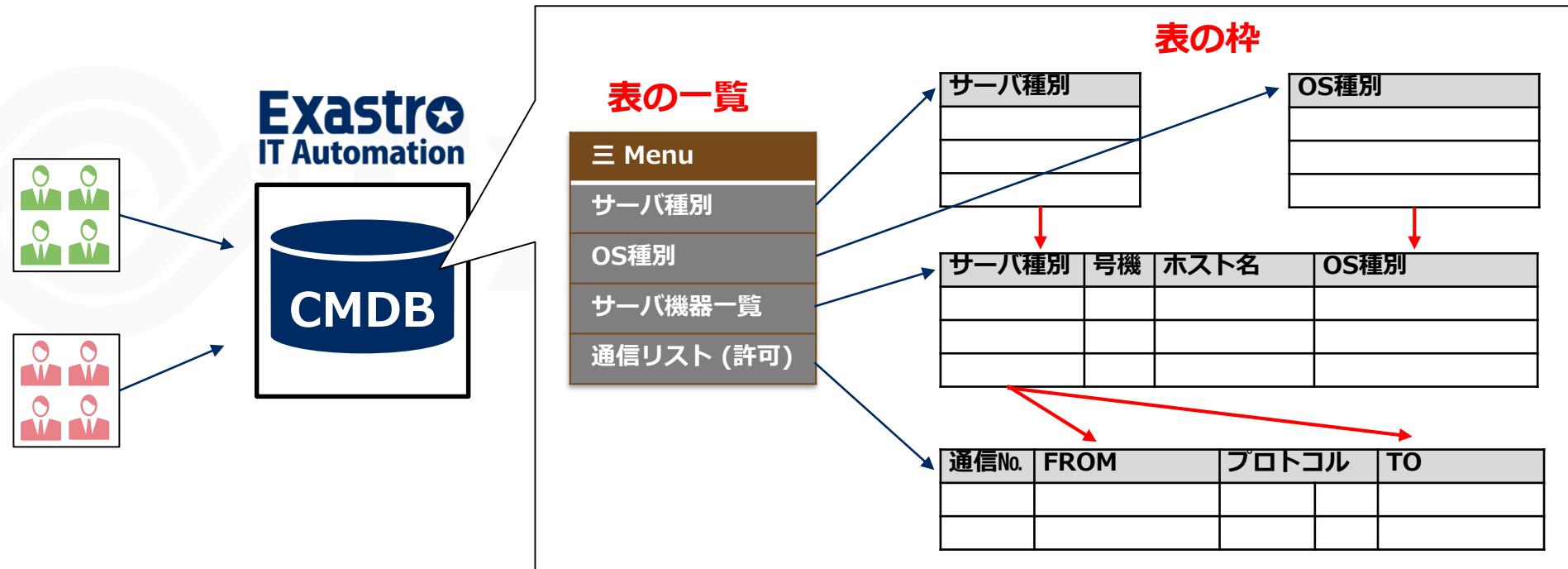
Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

タスクの説明

正規化した構成情報を元に、Exastro IT AutomationのCMDBに、構成情報を格納するための「表の一覧」と「表の枠」を作成します。



POINT

- ① 設計値の入力ミスを防ぐために、列に制約を付ける

詳細
次頁

Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

POINT

① 設計値の入力ミスを防ぐために、列に制約を付ける

CMDBに構成情報を登録する際に、設計値の入力ミスがあるとCMDBをクリーンな状態に保つことができません。

Exastro IT Automationでは表の列に制約を付けることで、設計値をの入力時に正しさを検査することができます。これにより、CMDBをクリーンな状態に保つことができます。

制約
英数字
ハイフン
ピリオド

制約
n.n.n.nの形式
(nは数字)

制約
プルダウン選択

ホスト名	IPアドレス	OS種別	選択式のため 入力ミス排除
web-server	10.0.10.100	Windows Server 2019	…
log-server	log-server	RHEL 8	…
DBサーバ	10.0.10.100	Windows Server 2019	…
…	…	Windows Server 2016	…
…	…	RHEL 8	…

Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

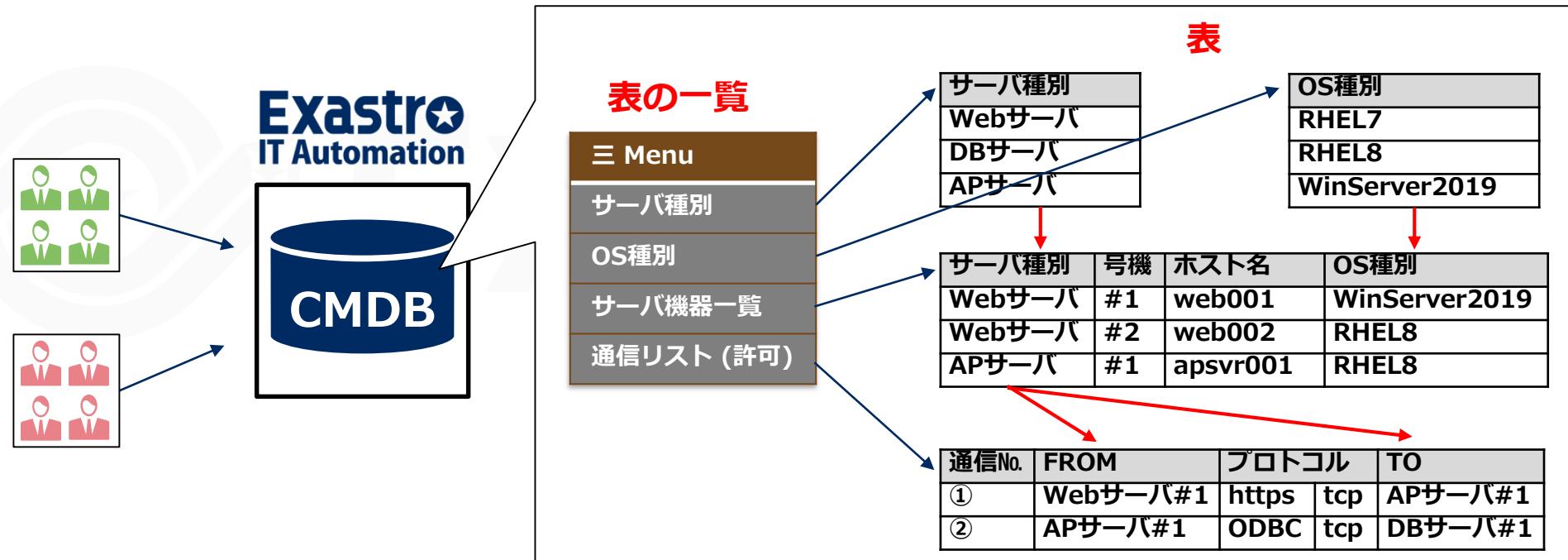
Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

タスクの説明

各チームは、正規化した構成情報をCMDBに登録していきます。



詳細
次頁

Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

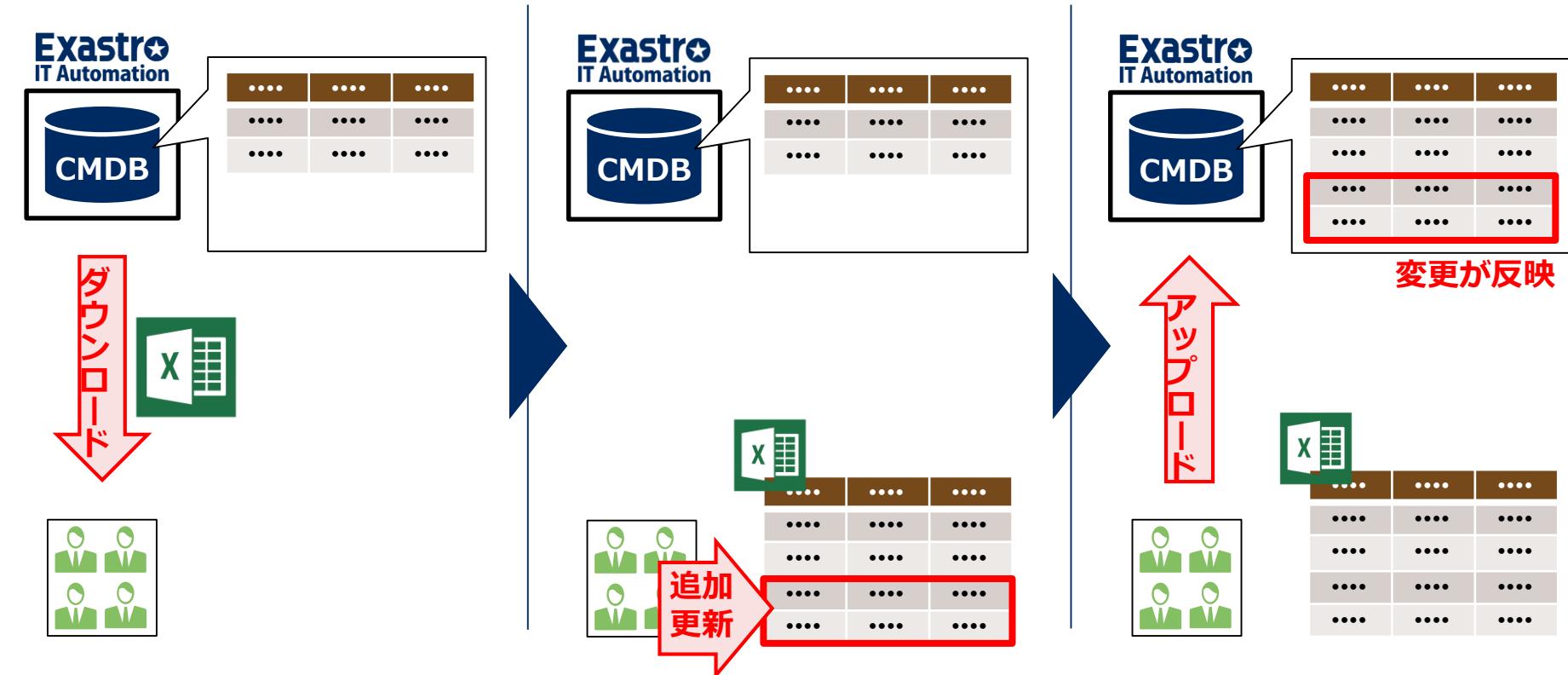
CMDBへの
情報投入

CMDBの活用

POINT

① エクセルを利用した一括登録を活用する

Exastro IT Automationの表は、Excelファイルの形式でダウンロードできます。このExcelファイルに対して構成情報を追加/更新し、アップロードすることで、効率的に構成情報を登録することができます。



Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

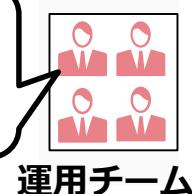
CMDBの活用

タスクの説明

構成情報を参照したり、更新したりして、目的に応じて活用します。
また、設計値をExcelでダウンロードして納品物にすることもできます。

【CMDBの参照や更新】

パッチ適用のために
全サーバのバージョン情報を
参照したい

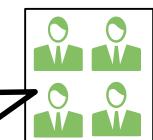


運用チーム

Exastro
IT Automation



Webサーバ増設の
ため、サーバー一覧
を更新しよう



サーバ構築チーム

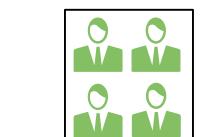
【Excelを納品物として提出】

Exastro
IT Automation



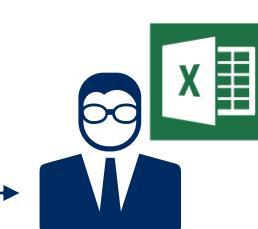
注意

事前にお客様との
合意が必要です。



プロジェクト

納品物の提出



お客様

POINT

① 事例～サービス停止の影響範囲の調査

詳細
次頁

Step 1：構成情報の一元管理

実施するタスク

構成情報の管理帳
票の収集

構成情報の正規化

Exastro IT
Automation
(CMDB)の構築

CMDBへの
情報投入

CMDBの活用

POINT

① 事例～サービス停止の影響範囲の調査

ここで、CMDBを利用したサービス停止の影響範囲の調査の事例を紹介します。

課題

大規模キャリアシステムで、計画および突発の機器停止によるサービス影響の調査に多くの工数が必要。

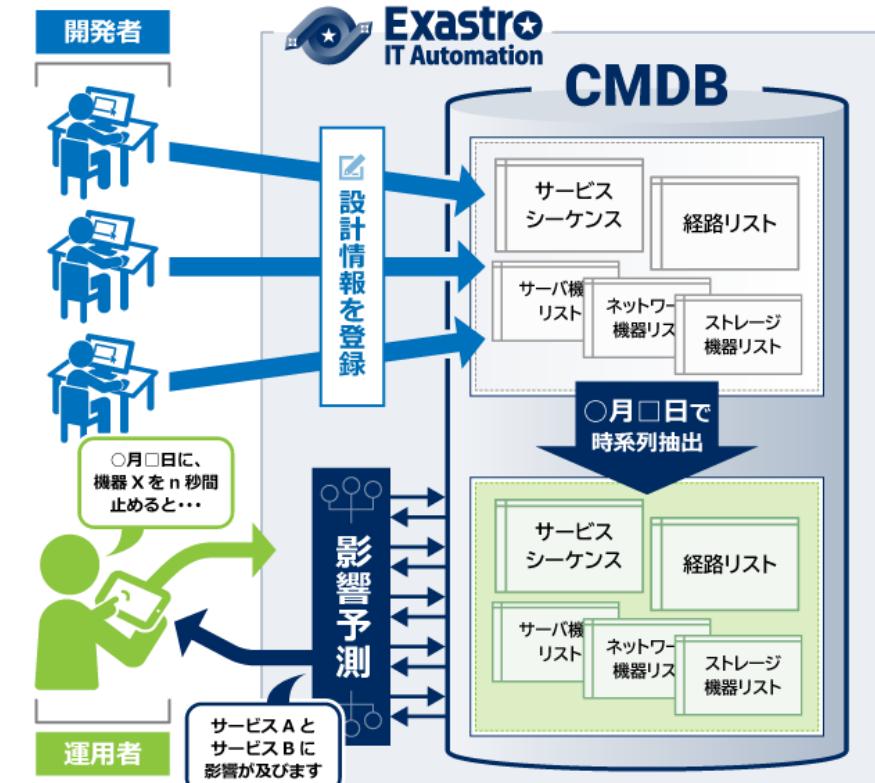
CMDB構築

解決

システムを構成管理することにより、機器停止によるサービス影響を自動予測することを可能とした。

効果

1回の調査費用80万が不要に。年間で約9,400万円(120回の調査)の費用削減効果があった。



本事例は、以下のURLで公開しています。

https://exastro-suite.github.io/it-automation-docs/case_ja.html

自動化の事前準備

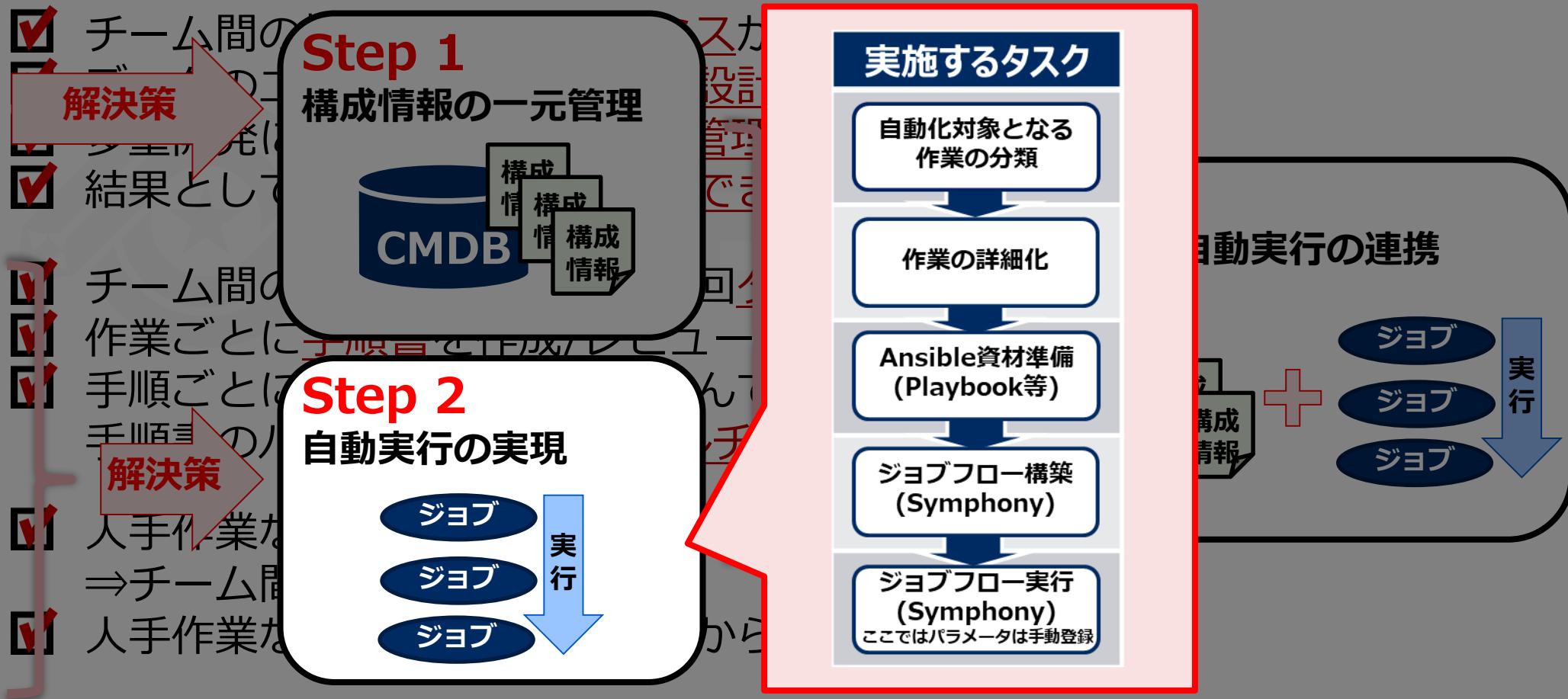
Step 1 : 構成情報の一元管理

Step 2 : 自動実行の実現

Step 3 : 一元管理と自動実行の連携



以降のスライドでは、Step 2の5つのタスクを説明していきます



Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

タスクの説明

手作業で実施している作業を整理して、自動化する作業を選定します。
整理が複数のチームをまたぐ場合は、各チームの代表者が調整します。

共通作業

- ・監視エージェント導入
- ・疎通確認(ping)
- ・hostsファイル配布
- ・etc

サーバ構築

- ・OS設定
- ・OSアップデート
- ・SELinux設定
- ・firewalld設定
- ・etc

NW機器構築

- ・IF設定
- ・VLAN構築
- ・通信許可設定
- ・etc

各チームの代表者

POINT

- ① 「丁度よい」粒度で作業を分類する
- ② 作業の効果を見積もって、優先度をつける

詳細
次頁

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

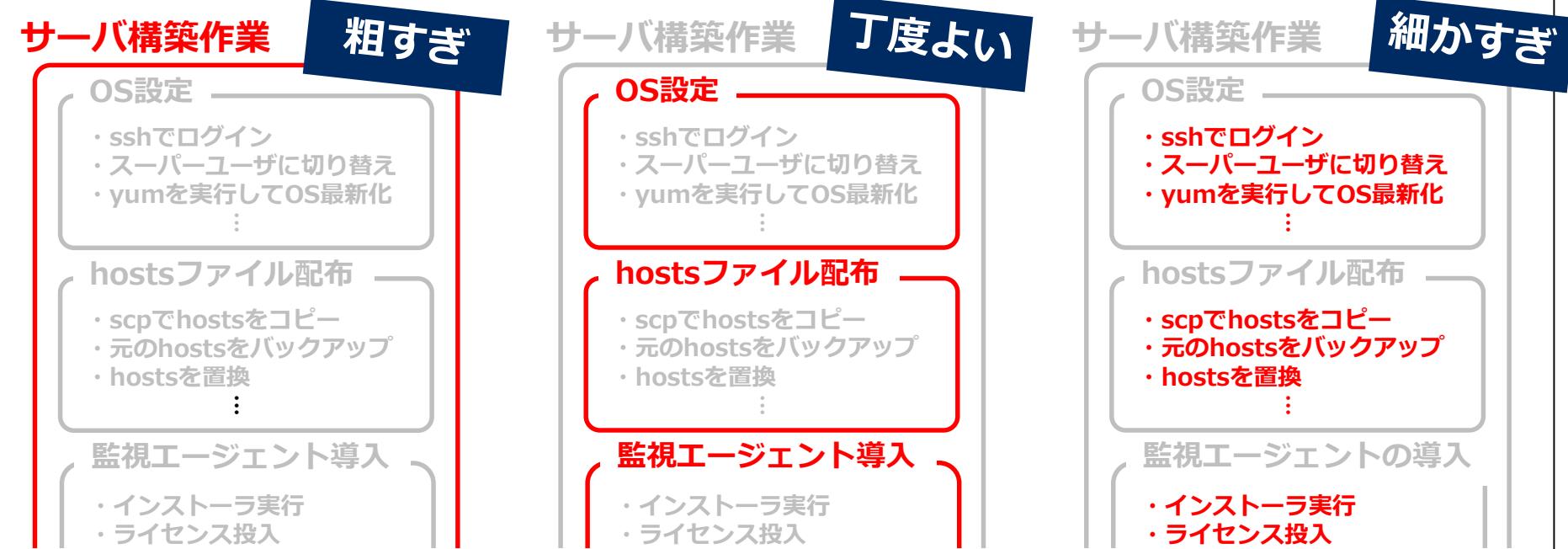
ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

POINT

① 「丁度よい」粒度で作業を分類する

自動化の候補となる作業を「丁度よい」粒度で分類します。例えばサーバ構築の場合、左下の図のような単にサーバ構築作業全体では粒度が粗すぎます。また、右下の図のような個々の作業手順では粒度が細かすぎます。

「丁度よい粒度」とは、繰り返し実行できる意味のある単位です。この例では中央下の図のような、「OS設定」などが丁度よい粒度になります。



実施するタスク

自動化対象となる
作業の分類

↓
作業の詳細化

Ansible資材準備
(Playbook等)

↓
ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

POINT

② 作業の効果を見積もって、優先度をつける

分類した作業に対して、自動化した場合の効果を見積もります。効果が分かれれば、自動化の優先度を付けたり、自動化するかしないかを判断したりできます。

効果の見積もりの観点は、年間の実施回数、対象の機器台数、1回あたりの作業工数や作業時間などです。定量的な数字がなければ大、中、小などの仕分けでも可能です。以下に、まとめ方の例を挙げます。

作業	実施回数	機器台数	作業工数	作業時間	優先度	備考
OS設定	50回	50台	10H	5H	高	1回に要員2名必要
hostsファイル配布	200回	50台	1H	0.5H	中	年に4回の更新
監視エージェント導入	30回	30台	5H	5H	低	
Webコンテンツ更新	600回	5台	1H	1H	高	月平均10回の更新
アクセスログ集計	60回	5台	2H	2H	低	月末に実施

一般論として、共通的な作業は年間の実施回数が多くなるため、自動化の効果が高い傾向にあります。また、作業の粒度を見直すことで、共通的な作業を発見できる場合もあります。

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

タスクの説明

分類した作業を詳細化して、具体的な作業手順に落とし込みます。
作業の詳細化は、既存の手順書などを参考にすることができます。

分類した作業

OS設定

hostsファイル配布

監視エージェント導入

Webコンテンツ更新

アクセスログ集計

.....

詳細化

詳細化

詳細化

OS設定

- sshでログイン
- スーパーユーザに切り替え
- yumを実行してOS最新化
- etc...

hostsファイル配布

- scpでhostsをコピー
- 元のhostsをバックアップ
- hostsを置換
- etc...

監視エージェントの導入

- インストーラ実行
- ライセンス投入
- etc...

POINT

①バックアップ/実作業/エビデンス取得の3点セットで

詳細
次頁

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

POINT

① バックアップ/実作業/エビデンス取得の3点セットで

詳細化した作業は、以下のような3点セットで構成にすることを推奨します。

(1) バックアップ

(2) 実処理

(3) エビデンス取得

このような構成にすることで、バックアップとエビデンスを確実に取得できるため、作業を安全に再利用することができます。

具体例として、hostsファイルの配布手順をこの構成で詳細化すると、以下のようになります。

処理	具体的な作業手順
(1) バックアップ	現行のhostsファイルをバックアップ
(2) 実処理	新しいhostsファイルを所定の場所にコピー
(3) エビデンス取得	名前解決成功の結果を保存

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

タスクの説明

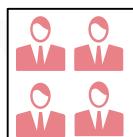
詳細化した手順を実施するAnsibleの資材(Playbook等)を準備します。
資材は、新規に作成するか、既存のものを再利用します。

Ansible資材とは、実行に必要な以下のファイル群です
・ Playbook
・ Role
・ ファイル
・ テンプレート

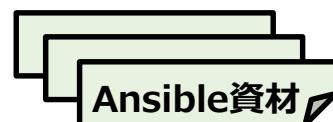
【Ansible資材の準備】



手順書から
新規作成

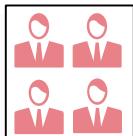


既存の
ものを
再利用



Ansible資材

【Ansible資材の登録】



登録



Exastro
IT Automation



登録



Ansible
資材

POINT

- ① 再利用可能な既存の資材を活用する
- ② 作業の実施ごとに変わる値は、変数化しておく
- ③ 同じような処理は「繰り返し」で簡潔に
- ④ 定型の設定ファイルは「ひな形」で生成する

詳細
次頁

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

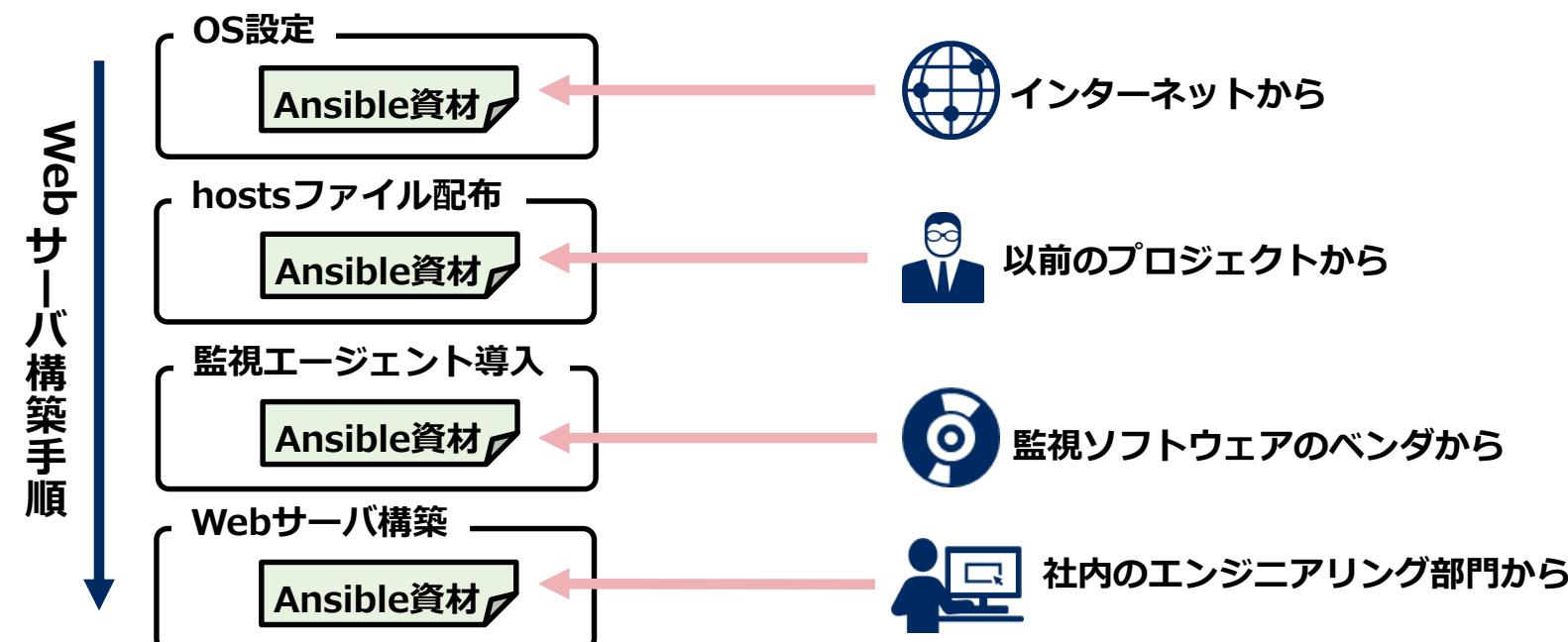
ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

POINT

① 再利用可能な既存の資材を活用する

Ansibleの資材は、全て自分で作る必要はありません。もし既存のAnsibleの資材があるならば、それを再利用することで効率的に資材の準備を行うことができます。

以下は、様々な場所からAnsibleの資材を調達して、Webサーバの構築手順を組み立てているイメージ図です。



実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

POINT

② 作業の実施ごとに変わる値は、変数化しておく

マシンに設定するホスト名など、作業の実施ごとに値が変わるものがあります。このような値を、Ansibleの資材に固定値で埋め込んでしまうと、作業を実施するたびにAnsibleの資材の修正が必要になってしまいます。

これを解決するためには、Ansibleの資材に「変数」を利用します。例えば、ホスト名を変更するAnsibleの資材(Playbook)は次のように記述します。

変数化前のPlaybook

```
- hostname:  
  name: web01
```

変数化

変数化後のPlaybook

```
- hostname:  
  name: {{ VAR_hostname }}
```

左のPlaybookには「web01」というホスト名が固定で記述されています。このままでは、別マシンに「web02」を設定するには、Playbookの修正が必要です。

一方、右のPlaybookではホスト名の部分が {{ VAR_hostname }} という形式で変数化されています。別途、変数に具体値を設定しておくことで、作業の実行時に変数化された部分を期待する値で置換することができます。

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

POINT

③ 同じような処理は「繰り返し」で簡潔に

自動実行する作業を整理すると、同じような作業を何度も実施していることがあります。この場合は「繰り返し」を利用すると簡潔になります。AnsibleのPlaybookの場合は、loopという命令を利用することができます。

以下に、/dir1と/dir2と/dir3の、3つのディレクトリを作成するPlaybookの例を示します。左側は同じような処理を3回記述しています。これに対し、右側はloopを利用して3回繰り返しているため簡潔になり、保守性が高まります。

繰り返しを利用していないPlaybook

```
- file:  
  path: /dir1  
  state: directory  
  
- file:  
  path: /dir2  
  state: directory  
  
- file:  
  path: /dir3  
  state: directory
```



繰り返しを利用したPlaybook

```
- file:  
  path: "{{ item }}"  
  state: directory  
loop: {{ VAR_dirs }}
```

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

POINT

④ 定型の設定ファイルは「ひな形」で生成する

複数台のサーバに対して設定ファイルを配布する状況では、設定ファイルの内容はほぼ同じで、一部の設定値だけ異なるということがよくあります。このような場合は「ひな形」を活用して設定ファイルを生成すると効率がよいです。

Ansibleでは、拡張子が.j2であるファイルがひな形になります。Playbookと同様に、ひな形でも変数が利用できます。以下はApacheの設定ファイルの生成例ですが、青い部分が変数で、赤い部分が生成後に置換された値です。

httpd.conf.j2 (ひな形)

```
<VirtualHost *:80>
  ServerName {{ VAR_hostname }}
  DocumentRoot {{ VAR_docroot }}
</VirtualHost>
```

生成

```
<VirtualHost *:80>
  ServerName www.test.com
  DocumentRoot /contents
</VirtualHost>
```

生成

```
<VirtualHost *:80>
  ServerName www.dev.com
  DocumentRoot /public
</VirtualHost>
```

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

Appendix : playbookの管理方法

Ansibleの資材(Playbook等)の管理方法について、実際に起きた問題とその解決方法を記載します。

PROBLEM

- ① 複数のディレクトリにまたがり同じplaybookが存在する
- ② 同じ名前なのに処理内容が違うplaybookが存在する
- ③ バージョン管理ツールとITAでplaybookの内容に差異

SOLUTION

- ① 複数の処理で同じPlaybookを利用できるようにする
- ② 事前に命名規則を決め、同名のファイルを非許容とする
- ③ ITAのCI/CD For IaC機能を利用して管理する

詳細
次頁

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

PROBLEM

① 複数のディレクトリにまたがり同じplaybookが存在する

共有ディレクトリで管理しようとした際に、処理単位でディレクトリを作成したために複数ディレクトリに同ファイルが存在してしまった。

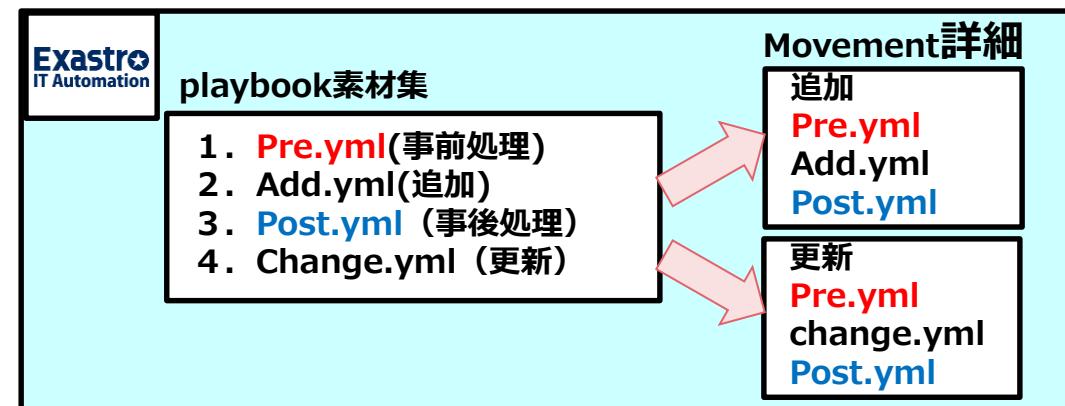
共有ディレクトリ

共有/追加/ Pre.yml Add.yml Post.yml	共有/更新/ Pre.yml change.yml Post.yml
---	--

SOLUTION

①複数の処理で同じPlaybookを利用できるようにする

ITAでは、playbookを一元管理し、同じPlaybookを複数のMovementで利用できる仕組みとなっております。



Step 2：自動実行の実現

実施するタスク

PROBLEM

自動化対象となる
作業の分類

作業の詳細化

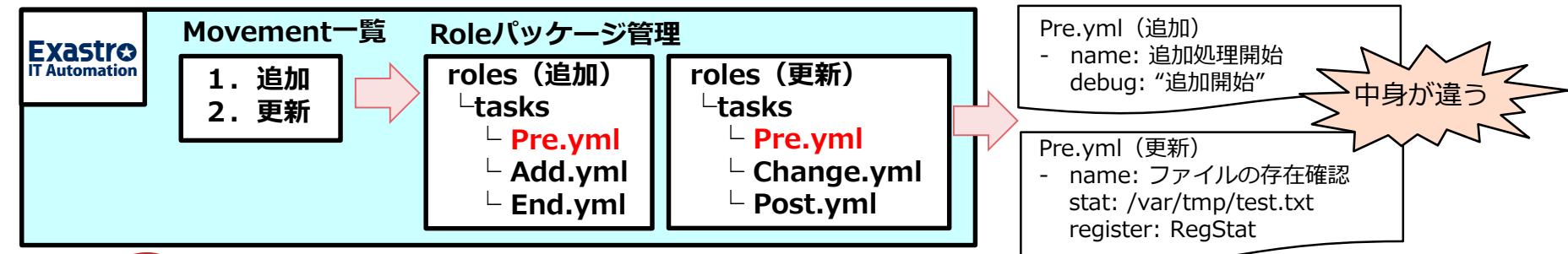
Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

② 同じ名前なのに処理内容が違うplaybookが存在する

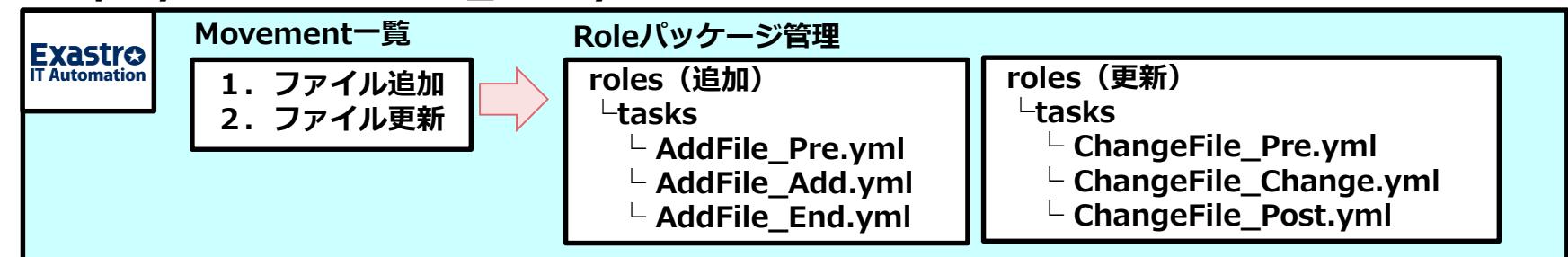
Ansible Legacy Roleにて、まったく同じファイル名にもかかわらず処理内容がことなるファイルを作ってしまった。そのため、追加のPre.ymlを修正したついでに同名のファイルということで更新のPre.ymlも更新してしまいバグが起きた。



SOLUTION

② 事前に命名規則を決め、同名のファイルを非許容とする

Ansible Roleは別パッケージであれば同名ファイルを許容可能ですが、バグが起りやすくなるため、事前に命名規則を決めて同名のファイルを非許容とすることをお勧めします。
例：playbookは「処理名_XXX.yml」の形式とする



Step 2：自動実行の実現

実施するタスク

PROBLEM

自動化対象となる
作業の分類

作業の詳細化

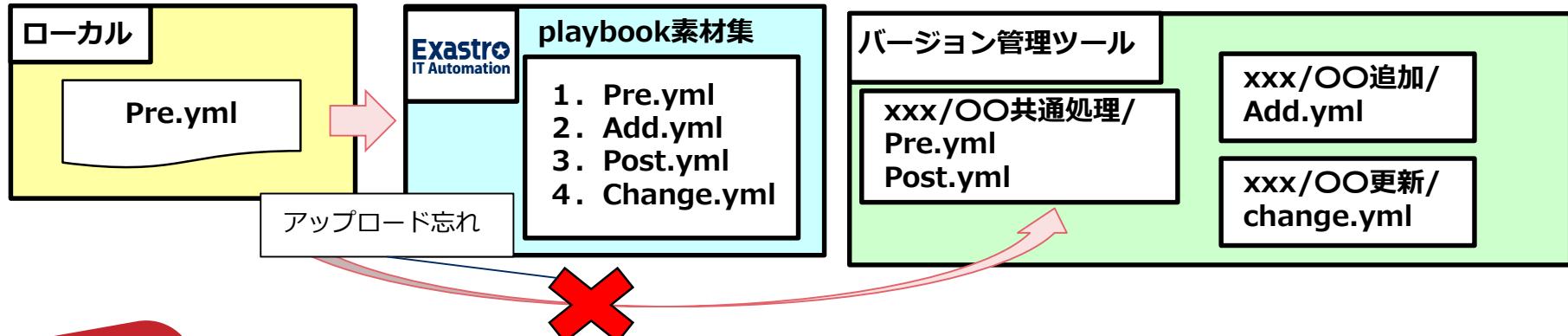
Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

③ バージョン管理ツールとITAでplaybookの内容に差異がでた

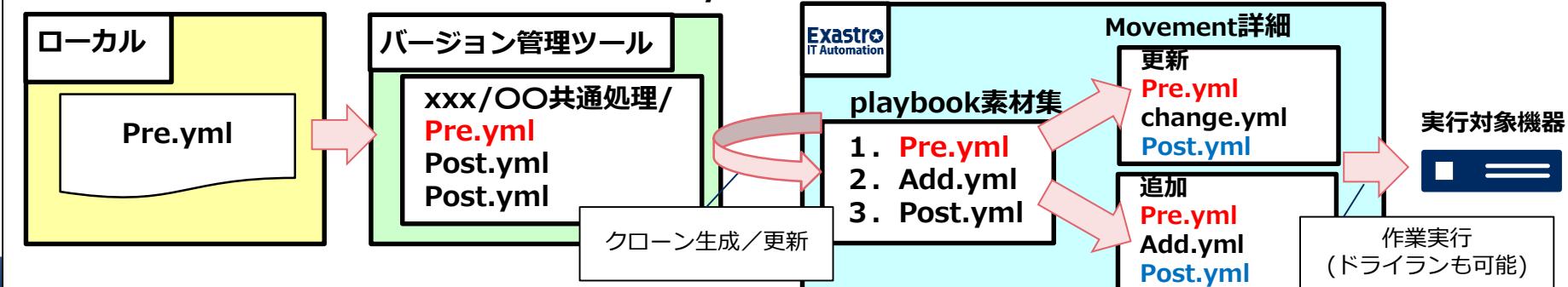
Playbookの追加や改修をした際に、ITAとバージョン管理ツール（Gitなど）の両方に手動でアップロードしていたが、アップロード忘れが起きて修正が反映されていなかった。



SOLUTION

③ ITAのCI/CD For IaC機能を利用して管理する

ITAの、CI/CD For IaC機能を活用することをお勧めします。本機能を活用することで、バージョン管理ツールの資材の更新のタイミングで、ITA上の資材の更新を自動で実施し、更に事前にMovementと連携しておくことでCI/CDの実現が可能です。



Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

参考

Ansible Trail Map - The Journey of Automation -

Ansible Trail Mapは、Ansibleを学習し活用するためにレッドハット社が提供する手引書です。Playbookの書き方など明解に説明されていますので本ガイドブックと合わせてご活用ください。



<https://www.redhat.com/ja/explore/ansible/trailmap>

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

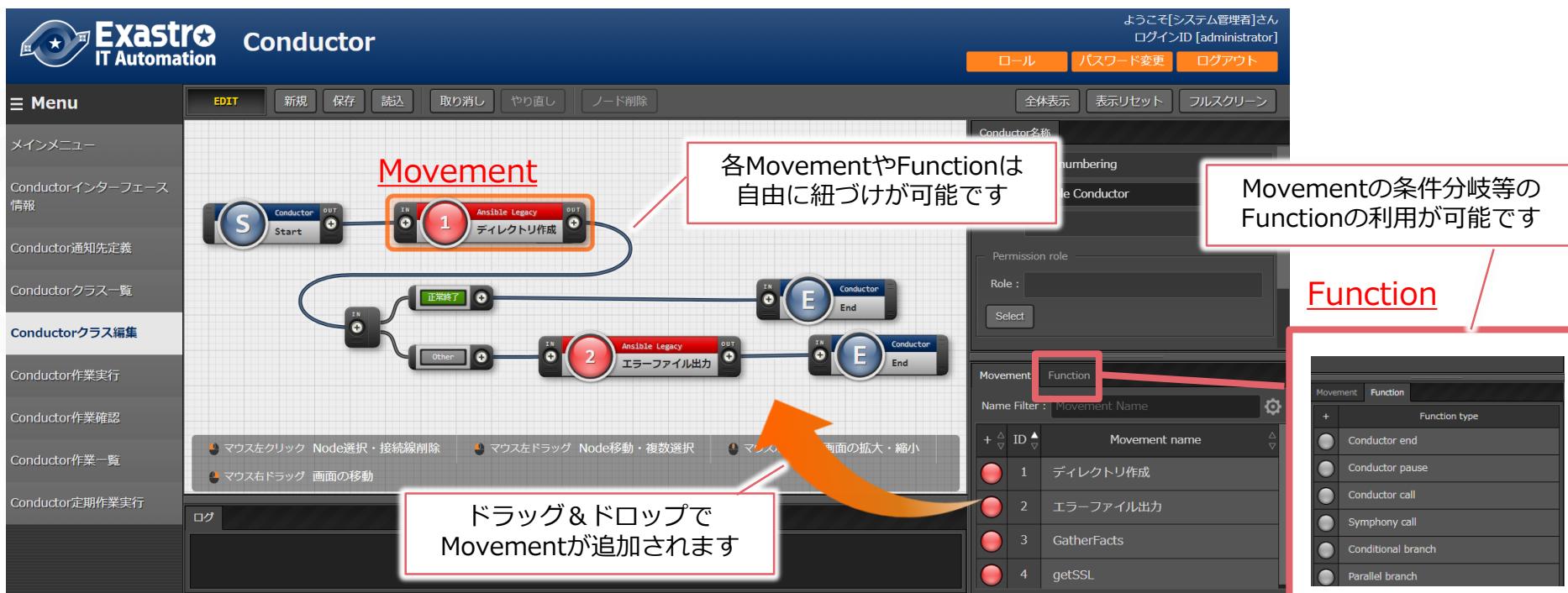
ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

タスクの説明

Exastro IT Automationでジョブフローを作成します。

【ジョブフロー作成画面】



POINT

① ジョブとジョブフローの組み立て方を理解する

詳細
次頁

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

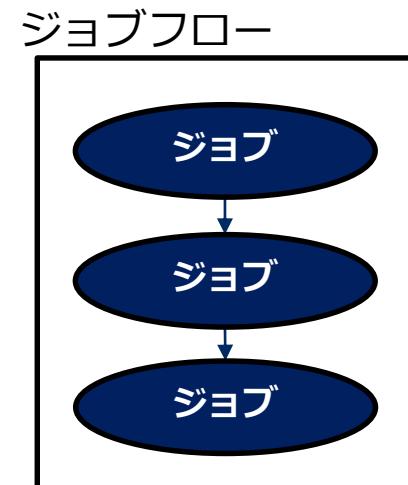
ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

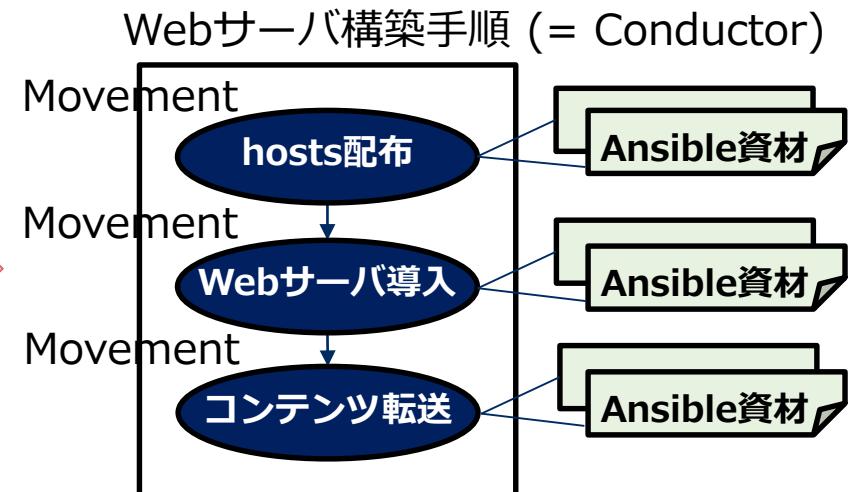
POINT

① ジョブとジョブフローの組み立て方を理解する

Step 2の最初のタスク「自動化対象となる作業の分類」で分類した作業を、ここでは「ジョブ」と呼ぶことにします。「ジョブフロー」とは、ジョブを順番に実行するために、いくつかのジョブを並べてまとめたものです。



Exastroで実現



Exastro IT Automationでは、ジョブフローを「Conductor」という機能で、またジョブを「Movement」という機能で実現しています。MovementにAnsible資材(Playbook等)を関連付けることで、実際の作業を実施できます。

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

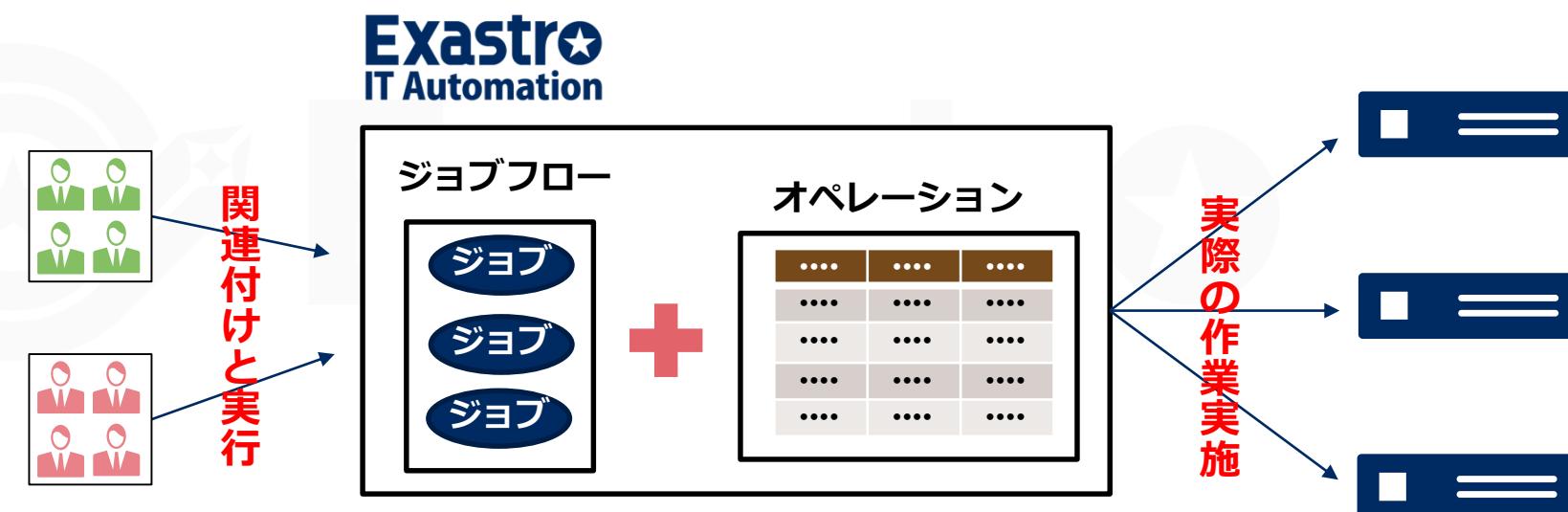
Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

タスクの説明

ジョブフローとオペレーションを関連付けて、作業を自動実行する



POINT

- ①ジョブフローとオペレーションの関係を理解する

詳細
次頁

Step 2：自動実行の実現

実施するタスク

自動化対象となる
作業の分類

作業の詳細化

Ansible資材準備
(Playbook等)

ジョブフロー構築
(Conductor)

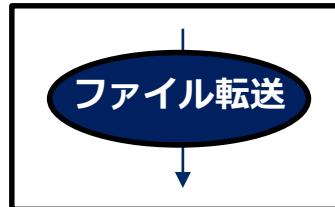
ジョブフロー実行
(Conductor)
ここではパラメータは手動登録

POINT

①ジョブフローとオペレーションの関係を理解する

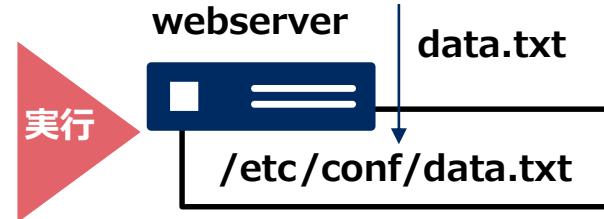
ジョブフローに対して、「対象機器」と具体的な「設定値」を関連付けるものがオペレーションです。以下に、ファイルをサーバに転送するだけの、シンプルなジョブフローの実行を考えてみます。

ジョブフロー



オペレーション

対象機器	転送元	転送先
webserver	data.txt	/etc/conf



ジョブフローに対し、オペレーションにより「対象機器」と「転送元」と「転送先」が関連付けられています。上記の組み合わせではwebserverにdata.txtが配置されます。

ここで、オペレーションを別のものに切り替えることにより、様々な機器に対して、様々なファイルを転送することができます。

自動化の事前準備

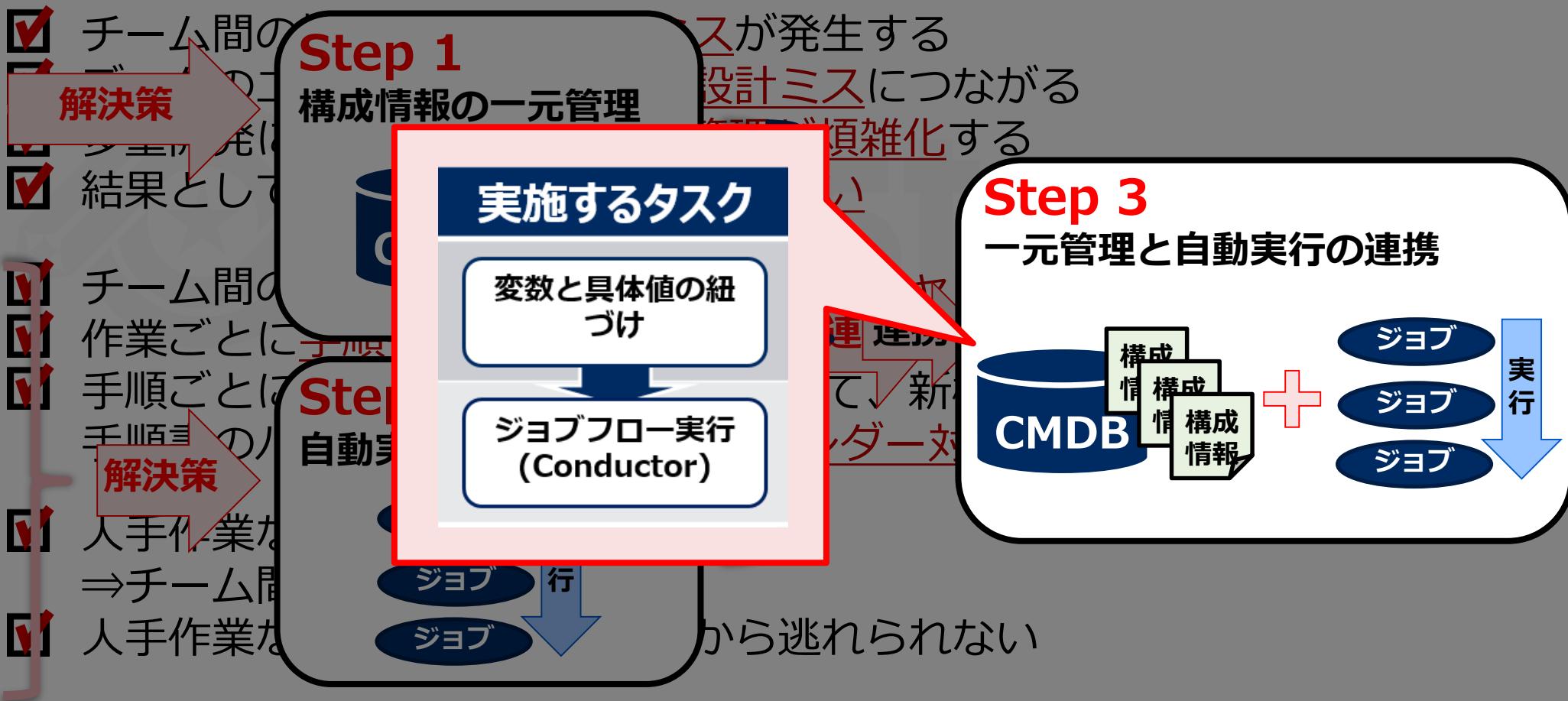
Step 1 : 構成情報の一元管理

Step 2 : 自動実行の実現

Step 3 : 一元管理と自動実行の連携



以降のスライドでは、Step 3の2つのタスクを説明していきます



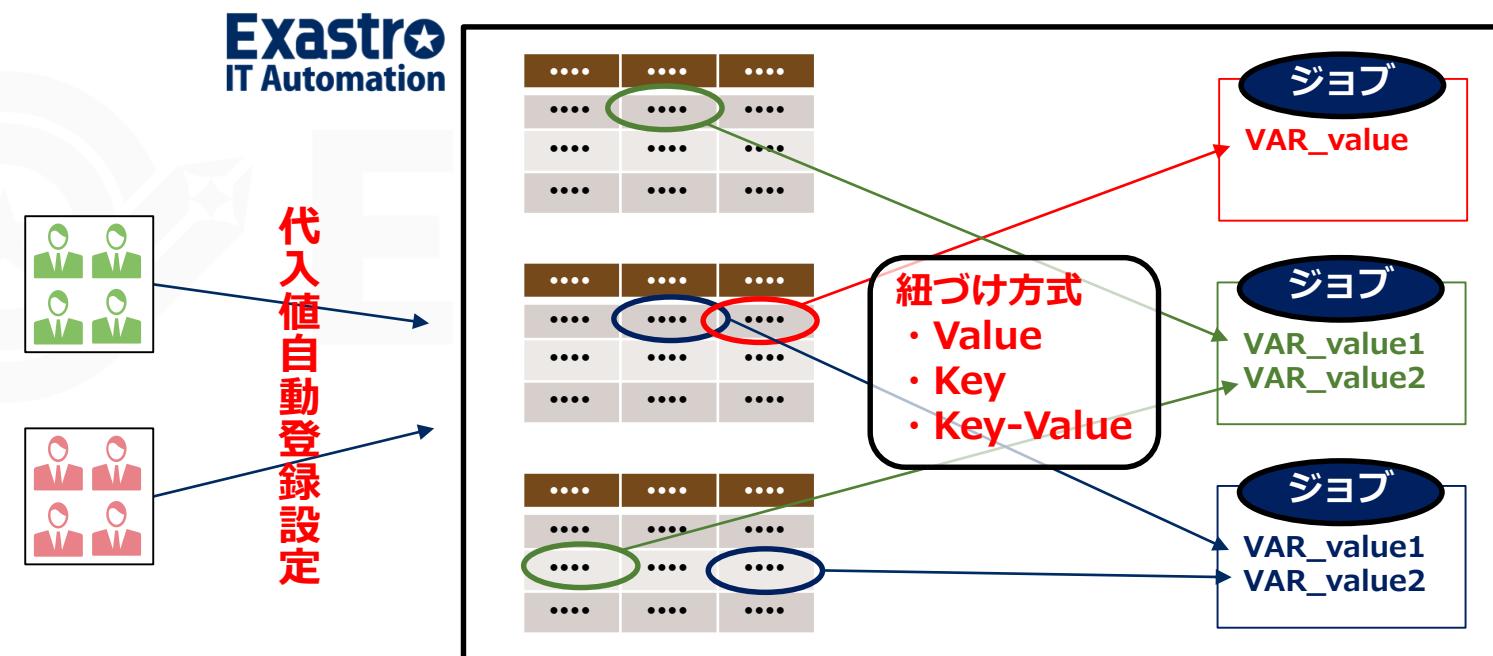
実施するタスク

変数と具体値の紐づけ

ジョブフロー実行
(Conductor)

タスクの説明

Exastro IT Automationの「代入値自動登録設定」を利用して、パラメータシートの値とジョブ内の変数とを紐づける



POINT

- ① Valueタイプの活用方法
- ② Keyタイプの活用方法
- ③ Key-Valueタイプの活用方法

詳細
次頁

実施するタスク

変数と具体値の紐づけ

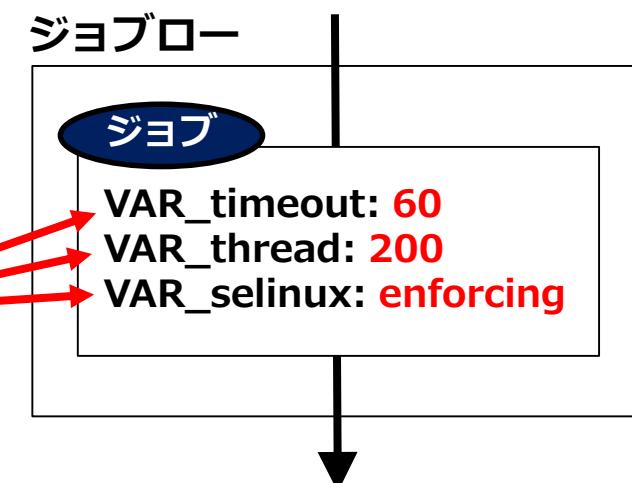
ジョブフロー実行
(Conductor)

POINT ① Valueタイプの活用方法

Valueタイプは基本的なタイプであり、表の中の値を変数に紐づけるものです。システム設定やコマンドライン引数など、様々な場面で活用できます。

具体例として、サーバの各種設定と変数の紐づけの方法を以下に示します。

ホスト名	タイムアウト	スレッド数	SELinux
web1	60	200	enforcing
web2	60	200	enforcing
db-server	30	50	permissive



上記の例では「web2」の各値を、ジョブが持つ各変数と紐づけています。

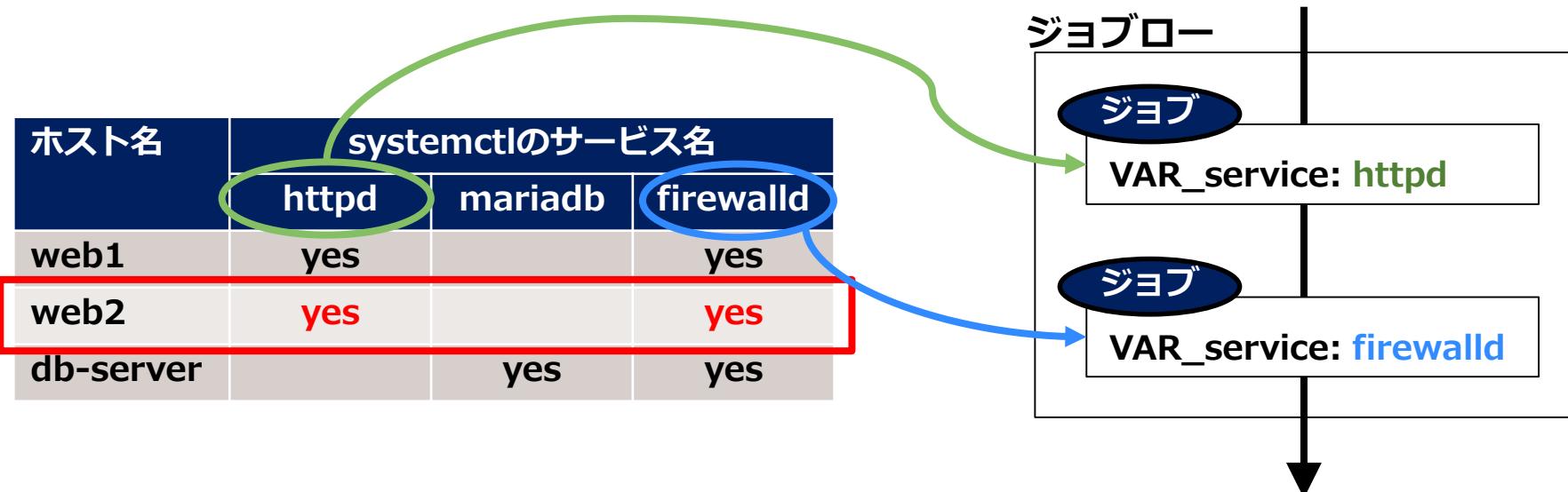
実施するタスク

変数と具体値の紐づけ

ジョブフロー実行
(Conductor)

POINT ② Keyタイプの活用方法

Keyタイプは表の列名を変数に紐づけるもので、主にフラグとして活用します。具体例として、サーバ上で動作させるサービスと変数の紐づけの方法を以下に示します。



上記の例では「web2」は列「httpd」と「firewalld」の値が「yes」になっているため、この列名が変数の具体値として紐づけられて、ジョブが実行されます。

実施するタスク

変数と具体値の紐づけ

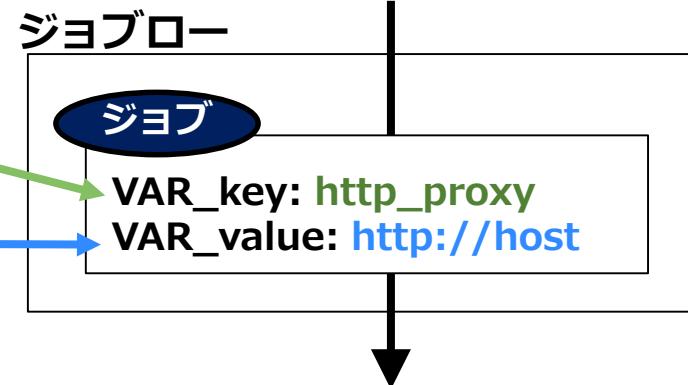
ジョブフロー実行
(Conductor)

POINT

③ Key-Valueタイプの活用方法

Key-Valueタイプを利用すると、KeyとValueの両方を変数に紐づけることができます。具体例として、環境変数の定義表を利用して、サーバに環境変数を設定する方法を以下に示します。

ホスト名	PATH	http_proxy
web1	/bin:/usr/bin	http://host
web2	/bin:/usr/bin	http://host
db-server	/bin:/sbin	http://proxy



上記の例では列名が環境変数名になっており、「http_proxy」という環境変数名と、その値「http://host」の両方を、変数に紐づけています。

Step 3：一元管理と自動実行の連携

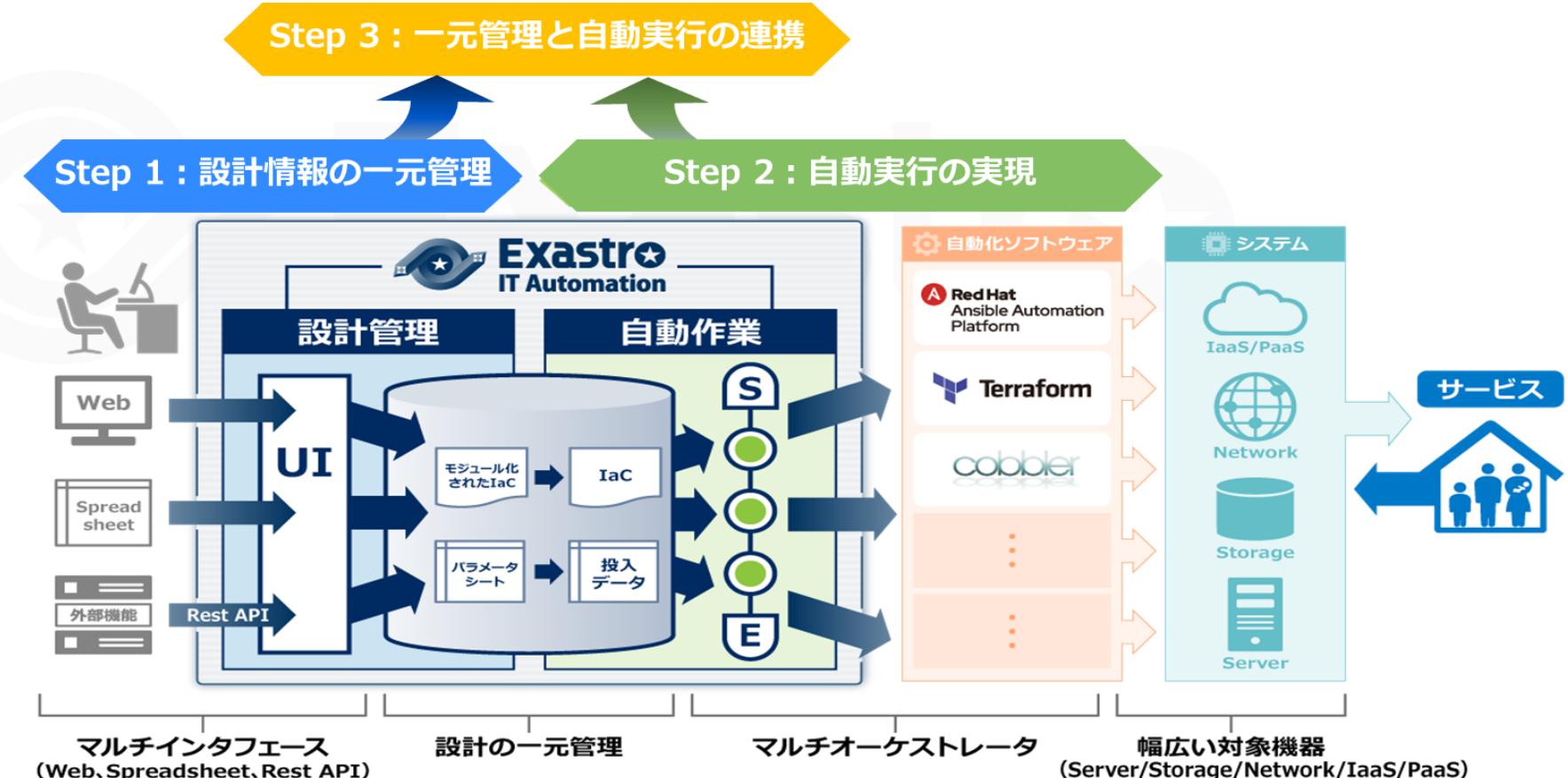
実施するタスク

変数と具体値の紐づけ

ジョブフロー実行
(Conductor)

タスクの説明

ジョブフローとオペレーションを関連付けて、作業を自動実行する。「パラメータ編集 → 実行」の2アクションでシステム構築を実現。



自動化されたSIの実施
効果の事例と見積りの観点
自動化後のプロセスと成果物の変更点



自動化されたSIの実施
効果の事例と見積りの観点
自動化後のプロセスと成果物の変更点



効果の見積もりの観点（再掲）

分類した作業に対して、自動化した場合の効果を見積もります。効果が分かれれば、自動化の優先度を付けたり、自動化するかしないかを判断したりできます。

効果の見積もりの観点は、年間の実施回数、対象の機器台数、1回あたりの作業工数や作業時間などです。定量的な数字がなければ大、中、小などの仕分けでも可能です。以下に、まとめ方の例を挙げます。

作業	実施回数	機器台数	作業工数	作業時間	優先度	備考
OS設定	50回	50台	10H	5H	高	1回に要員2名必要
hostsファイル配布	200回	50台	1H	0.5H	中	年に4回の更新
監視エージェント導入	30回	30台	5H	5H	低	
Webコンテンツ更新	600回	5台	1H	1H	高	月平均10回の更新
アクセスログ集計	60回	5台	2H	2H	低	月末に実施

一般論として、共通的な作業は年間の実施回数が多くなるため、自動化の効果が高い傾向にあります。また、作業の粒度を見直すことで、共通的な作業を発見できる場合もあります。

事例：ネットワーク機器構築 (1/2)

事例の概要

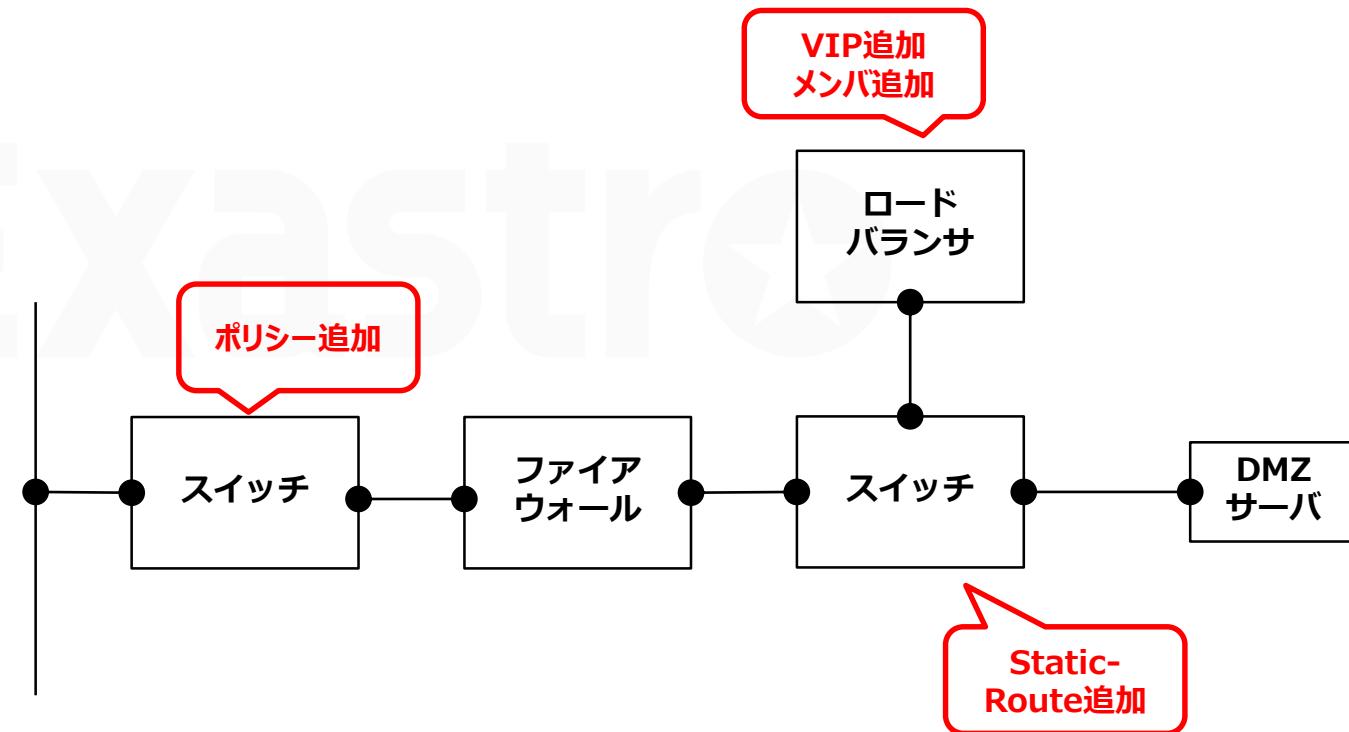
- キャリア系のシステムにおける、ネットワーク機器の増設
- 仮想IPの追加作業を自動化し、作業コストを自動化前後で比較

自動化対象の構成

- 構成は右図参照
- ネットワーク機器は合計30台

自動化対象の構成と作業

- ロードバランサへの仮想IPとメンバの追加
- ファイアウォールへのポリシーの追加
- スイッチへの静的ルートの追加



事例：ネットワーク機器構築 (2/2)

自動化前後の工数の増減と追加作業

		要件定義	基本設計	詳細設計	運用設計	製造	評価	リリース	合計
自動化前	工数(人H)	20.1	22.4	11.2	0	19.7	12	58.4	143.8
自動化後	工数(人H)	28.7	20.6	20.3	0	12.1	4	9.5	95.2
	増減(%)	(↑ 43%)	(↓ 8%)	(↑ 81%)	-----	(↓ 39%)	(↓ 67%)	(↓ 84%)	(↓ 34%)
新規作業	自動化検討		CMDB登録		ジョブフロー実行				

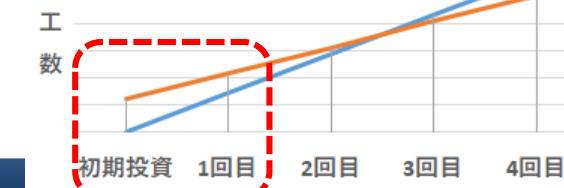
投資回収の考え方

- 自動化の準備(初期投資)の工数 : **123.4人H**
 - Step 1 : 44.7 人H Step 2 : 63.5 人H Step 3 : 15.2 人H
- 自動化前の工数 : **143.8人H** ⇒ 自動化後の工数 : **95.2人H**
 - 作業工数は**34%減**で、初期投資を含めて**3回目**で投資回収が可能
- 案件により、自動化準備(初期投資)と自動化実施を個別に行う場合と、同時に行う場合がある。この事例では個別に実施。

個別に実施



同時に実施



自動化されたSIの実施
効果の事例と見積りの観点
自動化後のプロセスと成果物の変更点



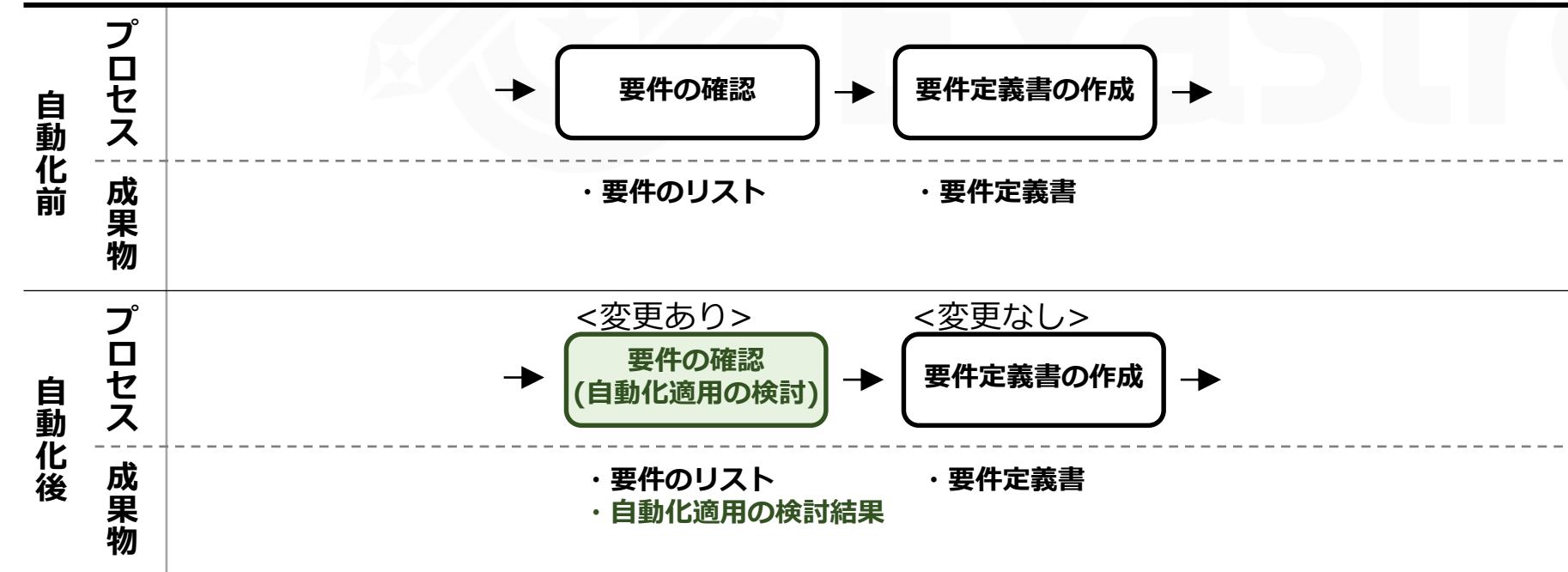
フェーズにおけるQCDの変化

凡例： ☺ 変化なし ☺ 改善 ☺ 追加の検討項目あり

	要件定義	基本設計	詳細設計	運用設計	製造	テスト	リリース
	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D
自動化前	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺
自動化後	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺

プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



解説

要件の確認の段階で、自動化の適用範囲等について検討を行い、合意を取る必要がある。その検討分としてCとDが増加する。

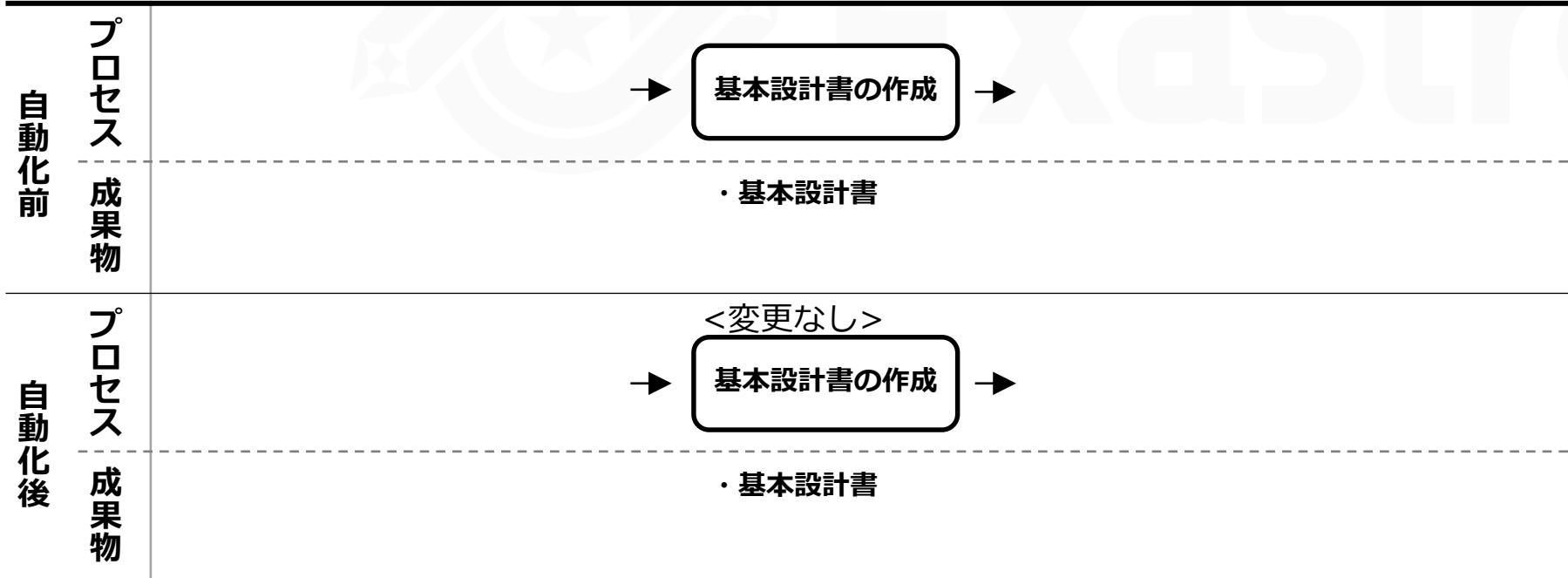
フェーズにおけるQCDの変化

凡例： ☺ 変化なし ☺ 改善 ☺ 追加の検討項目あり

	要件定義			基本設計			詳細設計			運用設計			製造			テスト			リリース		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
自動化後	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺

プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



解説

基本設計に取り込むべき内容は、自動化の事前準備の段階で検討済みであるため、ここでは追加の作業はない。

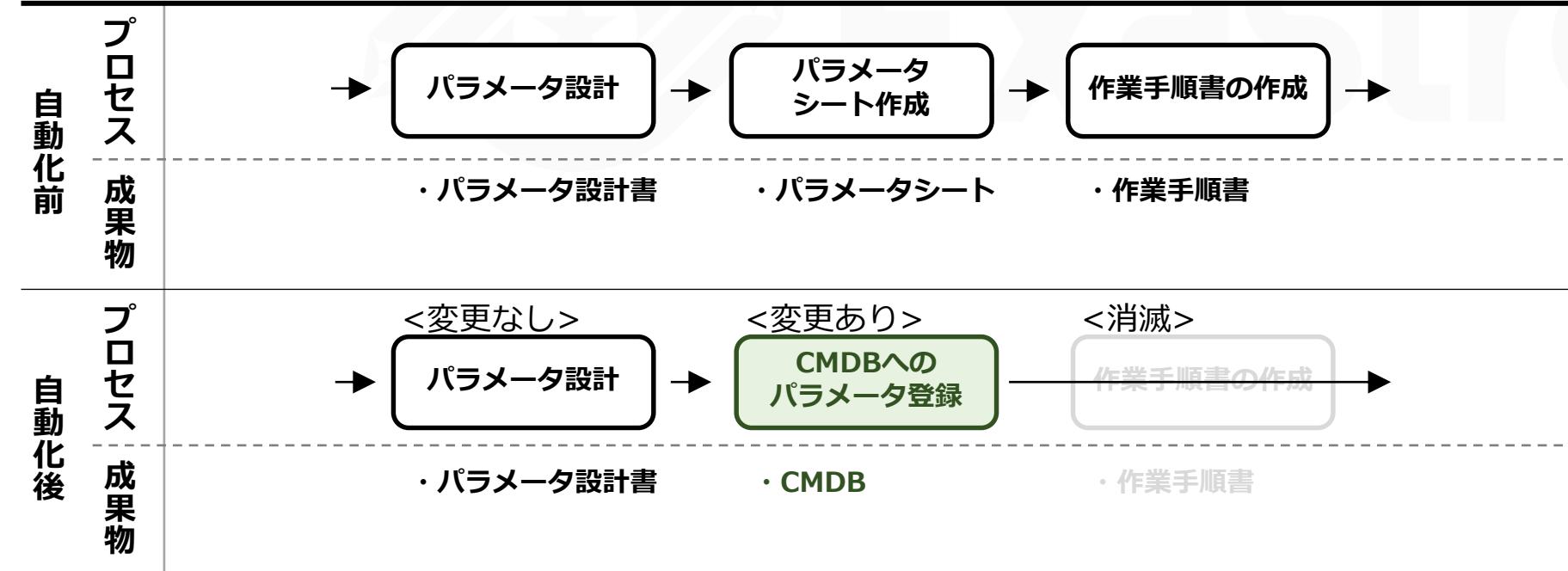
フェーズにおけるQCDの変化

	要件定義			基本設計			詳細設計			運用設計			製造			テスト			リリース		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊
自動化後	😊	😢	😅	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊

凡例： 😊 変化なし 😃 改善 🤖 追加の検討項目あり

プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



解説

パラメータ設計で作成された各パラメータは、CMDBに登録していく。これによりパラメータの形式化が進み、表記ゆれや曖昧さが低減するため、Qが改善する。

また、パラメータの適用順序などの作業手順は、自動化の事前準備の段階で作成したジョブフローに置き換わるため、作業手順書の作成のタスクが消滅する。これにより、CとDも改善する。

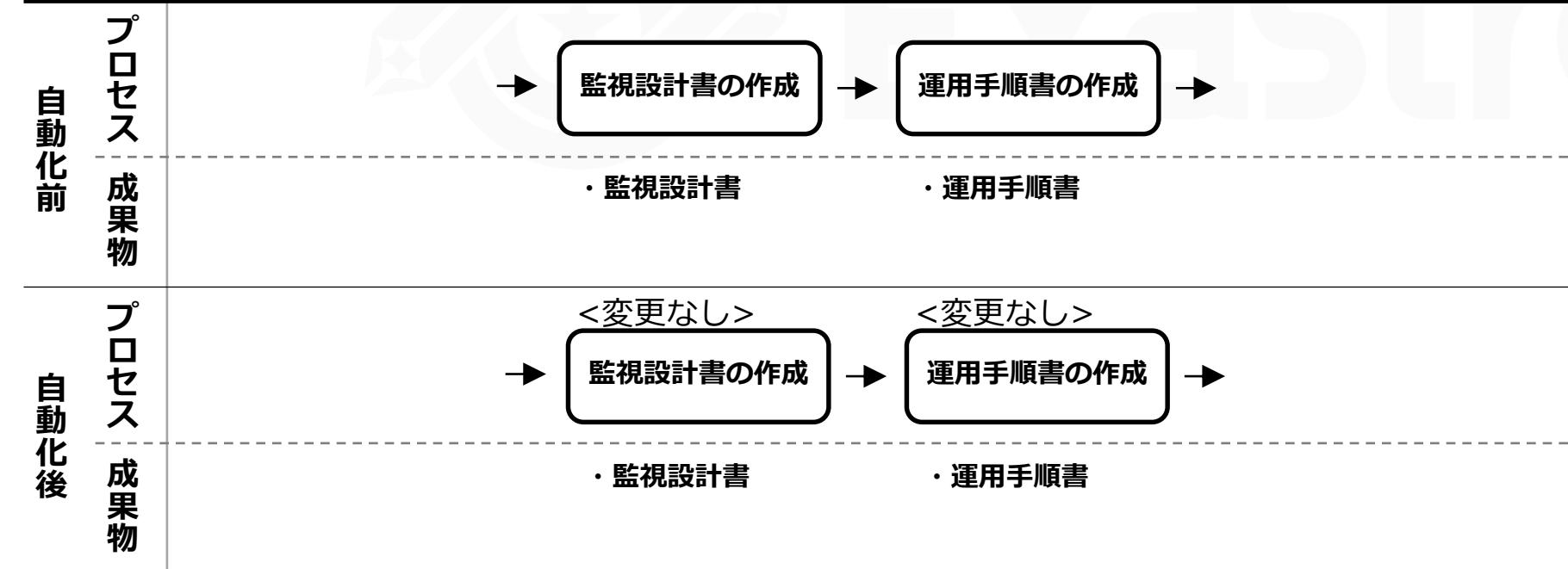
フェーズにおけるQCDの変化

凡例： 変化なし 改善 追加の検討項目あり

	要件定義	基本設計	詳細設計	運用設計	製造	テスト	リリース
	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D
自動化前							
自動化後							

プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



解説

本書は構築の自動化に焦点を当てているため、運用の自動化は範囲外とする。

運用の自動化を実施する場合は、QCDおよびプロセスの変化が発生する。

フェーズにおけるQCDの変化

凡例： ☺ 変化なし ☺ 改善 ☺ 追加の検討項目あり

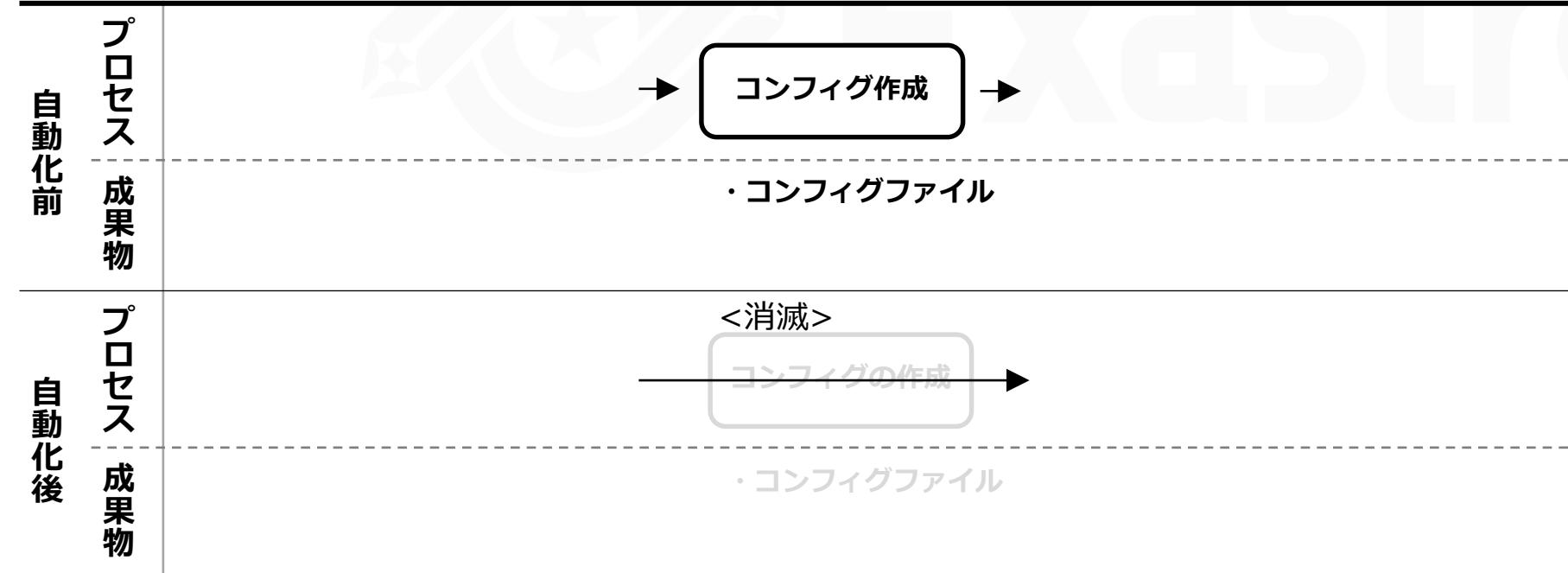
	要件定義	基本設計	詳細設計	運用設計	製造	テスト	リリース
	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D	Q C D
自動化前	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺
自動化後	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺	☺ ☺ ☺

解説

詳細設計に基づき作成されるコンフィグファイルは、IaCとCMDBにより自動生成されるため、コンフィグ作成のタスクは消滅する。

プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



テスト

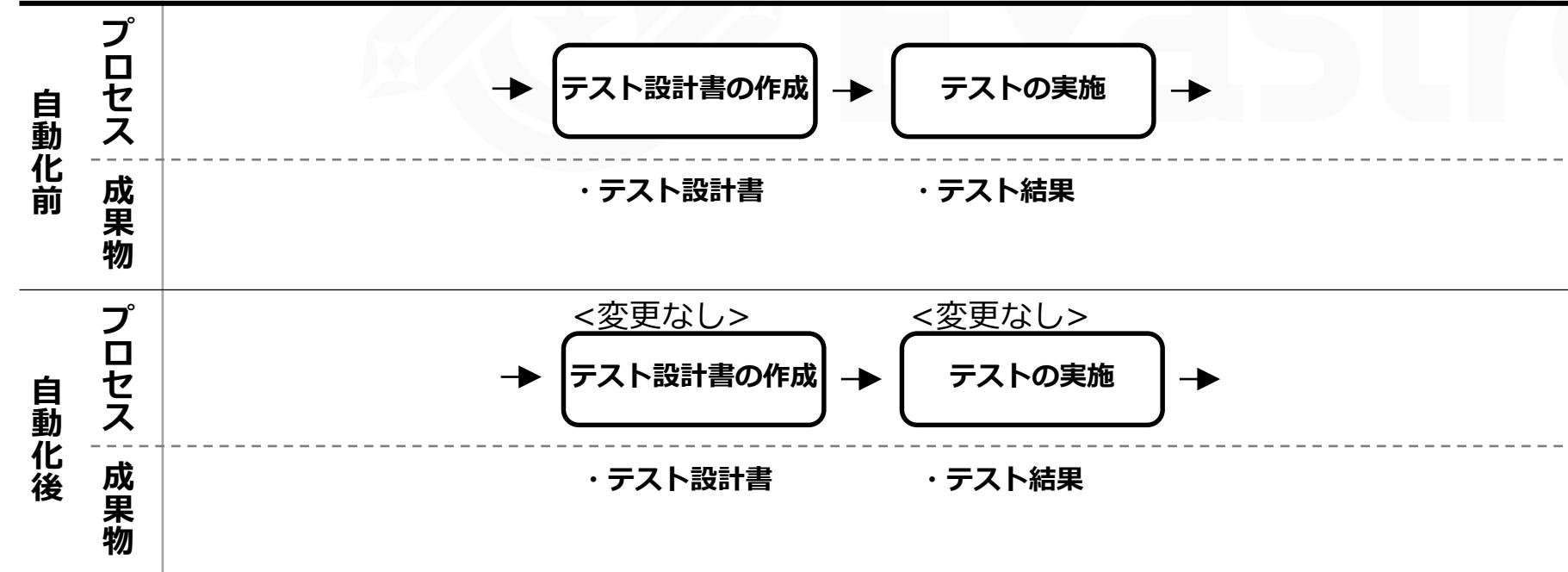
フェーズにおけるQCDの変化

凡例： ☺ 変化なし ☺ 改善 ☺ 追加の検討項目あり

	要件定義			基本設計			詳細設計			運用設計			製造			テスト			リリース		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
自動化後	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺

プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



解説

本書は構築の自動化に焦点を当てているため、テスト自動化は範囲外とする。

テストの自動化を実施する場合は、QCDおよびプロセスの変化が発生する。

リリース

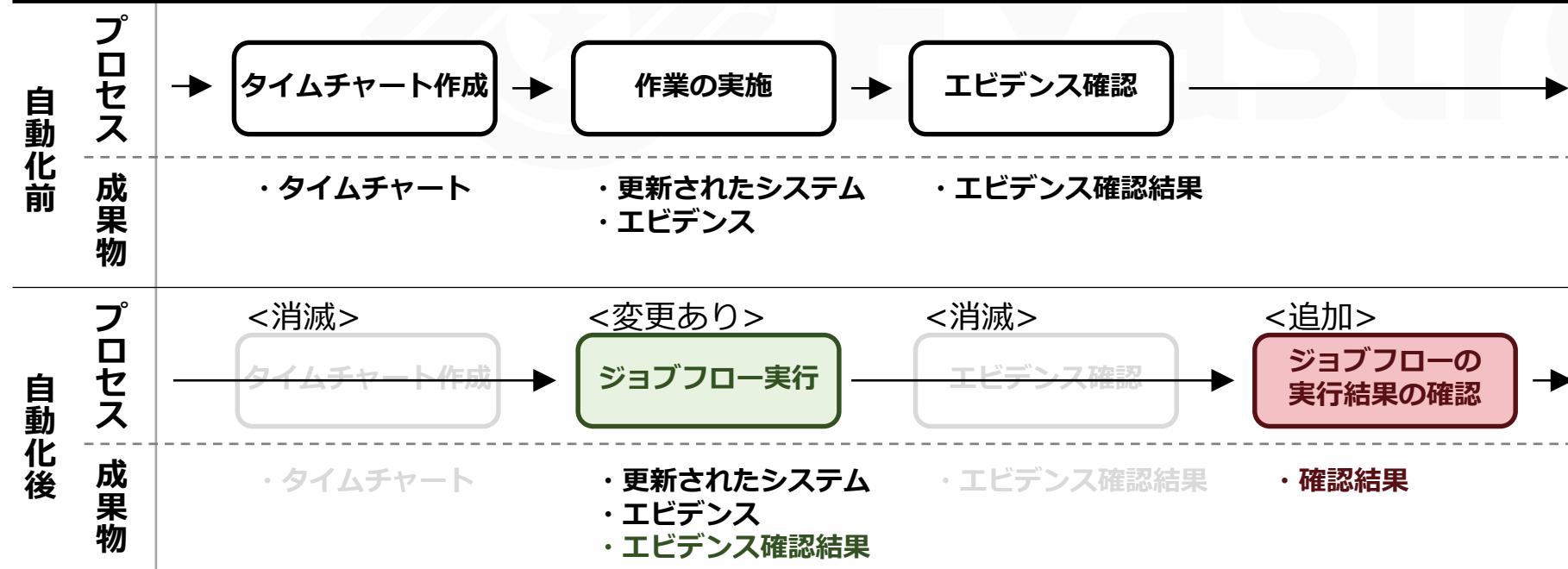
フェーズにおけるQCDの変化

凡例： 変化なし 改善 追加の検討項目あり

	要件定義			基本設計			詳細設計			運用設計			製造			テスト			リリース		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
自動化前																					
自動化後																					

プロセスと成果物の変化

凡例： 変更なし 作業名 変更あり 作業名 追加 作業名 消滅 作業名



解説

詳細設計で作成したジョブフローの実行がメインの作業になる。

タイムチャート作成がジョブフロー実行に置き換わるため消滅。

またジョブフロー中でエビデンスの確認も実施するため、タスクとしてのエビデンス確認も消滅。

よって、ジョブフローの実行と結果の確認のみとなり、QCDとともに改善する

まとめ



Step 1～3を実施することで、システムの自動構築・運用が可能になります。さらに、プロセスを変更することで、より効率的な自動化を実現できます。

TO-BE

自動化されたシステム構築/運用

AS-IS



自動化の事前準備 (Step 1、Step 2、Step 3)



自動化されたSIの実施 (プロセスと成果物の変更)

手作業でのシステム構築/運用



Exastro 