

10/9 「一歩先行くプラットフォーム構築・運用自動化」
(主催:マイナビ)にて紹介した資料より抜粋



IT Automation

クラウドネイティブなシステムを提供する準備はお済みですか？

CNCF Cloud Native Definition v1.0

Approved by TOC: 2018-06-11

中文版本 | 日本語版 | 한국어 | Deutsch | Español | Français | Polski | Português Brasileiro | Русский (in Chinese, Japanese, Korean, Brazilian, Portuguese, German, French and Spanish below)

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.

日本語版:

クラウドネイティブ技術は、パブリッククラウド、プライベートクラウド、ハイブリッドクラウドなどの近代的でダイナミックな環境において、スケーラブルなアプリケーションを構築および実行するための能力を組織にもたらします。このアプローチの代表例に、コンテナ、サービスメッシュ、マイクロサービス、イミュータブルインフラストラクチャ、および宣言型APIがあります。

これらの手法により、回復性、管理力、および可観測性のある疎結合システムが実現します。これらを堅牢な自動化と組み合わせることで、エンジニアはインパクトのある変更を最小限の労力で頻繁かつ予測どおりに行なうことができます。

Cloud Native Computing Foundationは、オープンソースでベンダー中立プロジェクトのエコシステムを育成・維持して、このパラダイムの採用を促進したいと考えています。私たちは最先端のパターンを民主化し、これらのインベーションを誰もが利用できるようにします。

<https://github.com/cncf/toc/blob/master/DEFINITION.md>



クラウドネイティブなシステムを提供できる技術者は不足しています

新技術を取り込みにくい

密結合で保守しづらいアプリ

従来のスタティックな環境

手動構築・手動運用
多くのITエンジニアが
張り付いている

ITエンジニア不足

Mobile

Social

IoT

5G

Tech

スケーラブルなアプリ
コンテナ
サービスメッシュ
イミュータブルインフラストラクチャ
宣言型API

近代的でダイナミックな環境

パブリッククラウド
プライベートクラウド
ハイブリッドクラウド

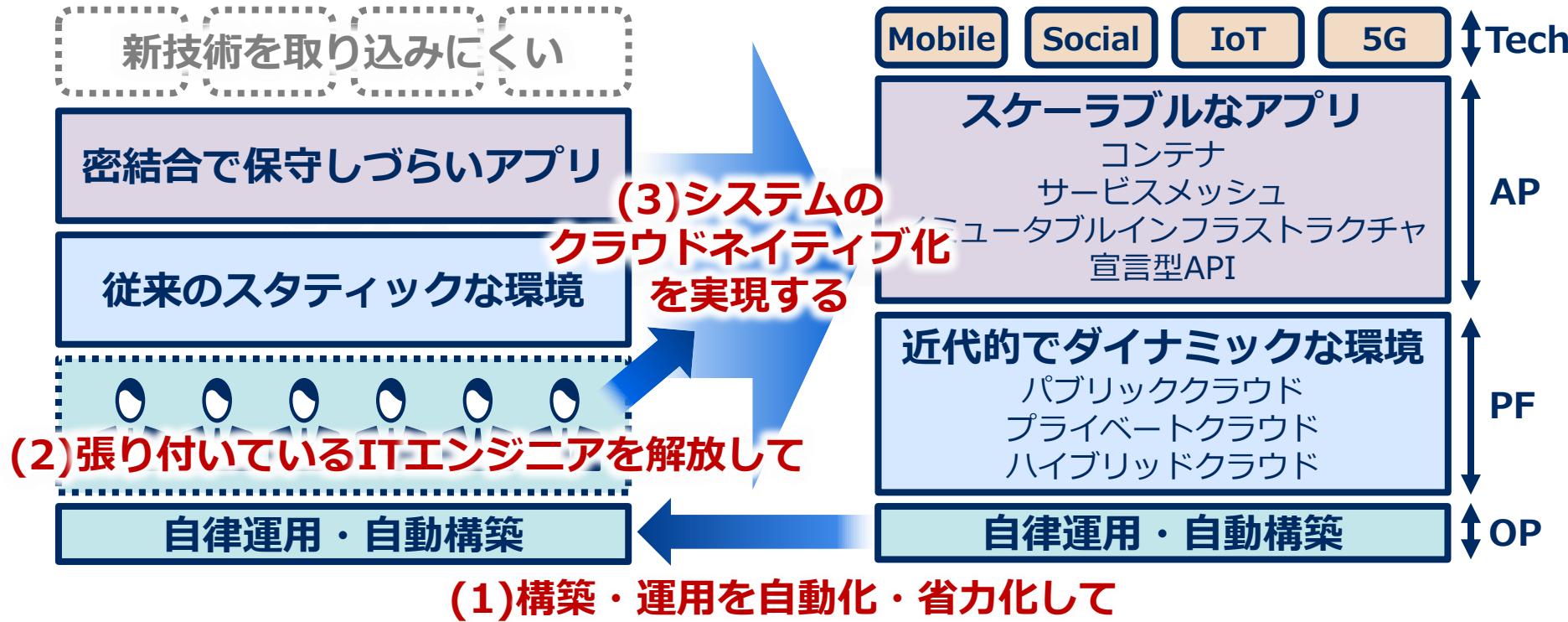
自律運用・自動構築

AP

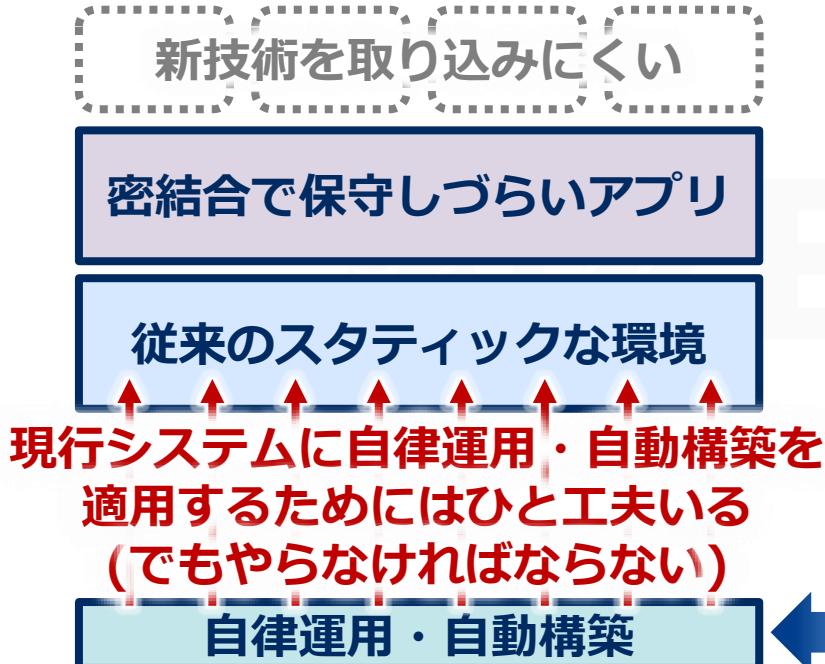
PF

OP

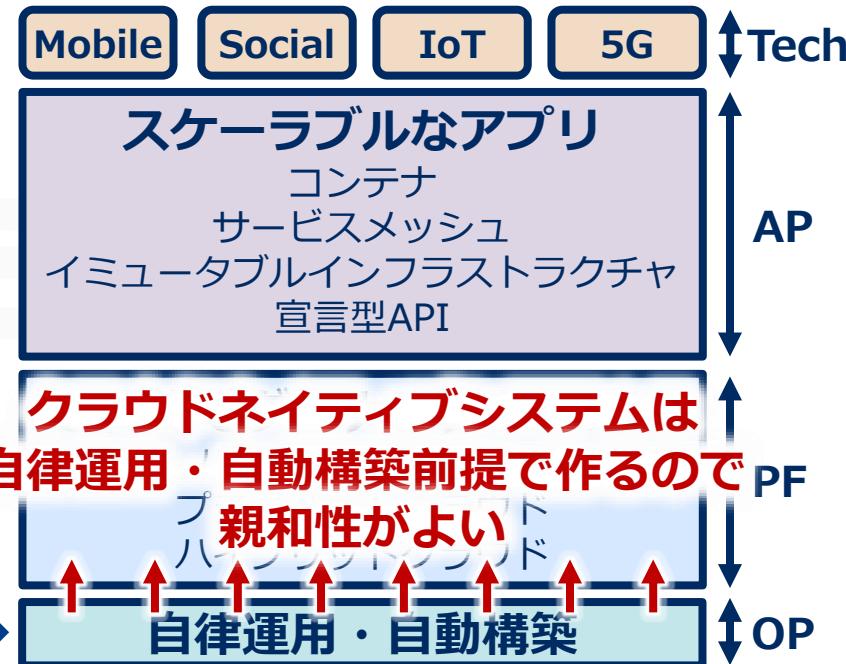
現行システムの構築・運用を自動化してITエンジニアを確保しましょう



現行システムの構築・運用を自動化するためには工夫が必要です



違いは？



以降は現行システムの構築・運用を自動化するための工夫について話します

新技術を取り込みにくい

密結合で保守しづらいアプリ

従来のスタティックな環境

以降のテーマ

自律運用・自動構築

Mobile

Social

IoT

5G

Tech

スケーラブルなアプリ

コンテナ

サービスメッシュ

イミュータブルインフラストラクチャ
宣言型API

近代的でダイナミックな環境

パブリッククラウド

プライベートクラウド

ハイブリッドソリューション

こっちにも適用できる技術だが

AP

PF

OP

現行システムの構築・運用に携わる ITエンジニアの「苦」



現行システムのSIに携わるITエンジニアの現場の声をまとめてみました



設計

- チーム間の情報伝達に遅延やミスが発生する
- データの二重管理や独自文言が設計ミスにつながる
- 多重開発により設計書(帳票)の管理が煩雑化する
- 結果として設定の前後性を確認できない



作業準備

- 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
- 手順ごとにコンフィグを埋め込んでいて、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)



作業実施

- 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

現行システムの運用に携わるITエンジニアの現場の声をまとめてみました



管理不足

- 運用上変更してよいパラメータと変更してはいけないパラメータが把握できていない
- システムのパラメータの現在値や過去の変更履歴を管理できていない
- 結果としてせっかくパラメータ化されているのに運用で利用できていない



高負荷

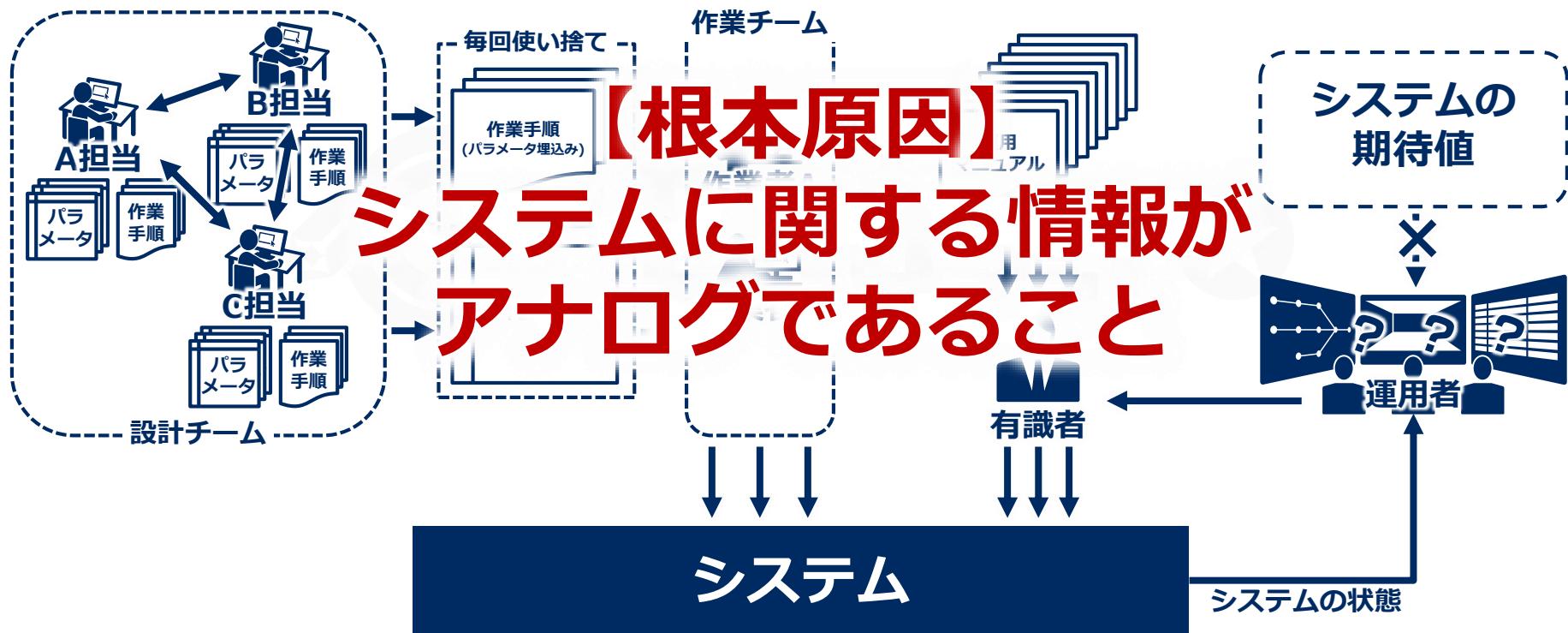
- システムは複雑化の一途を辿っており作業量は増大するばかり
- 何か起こるとExcelで書かれた大量のマニュアルを読み替えながら複数人体制で1作業ずつ慎重に実行するしかない
- 結果としてシステムの故障時間が長くなりサービスにも影響が出る



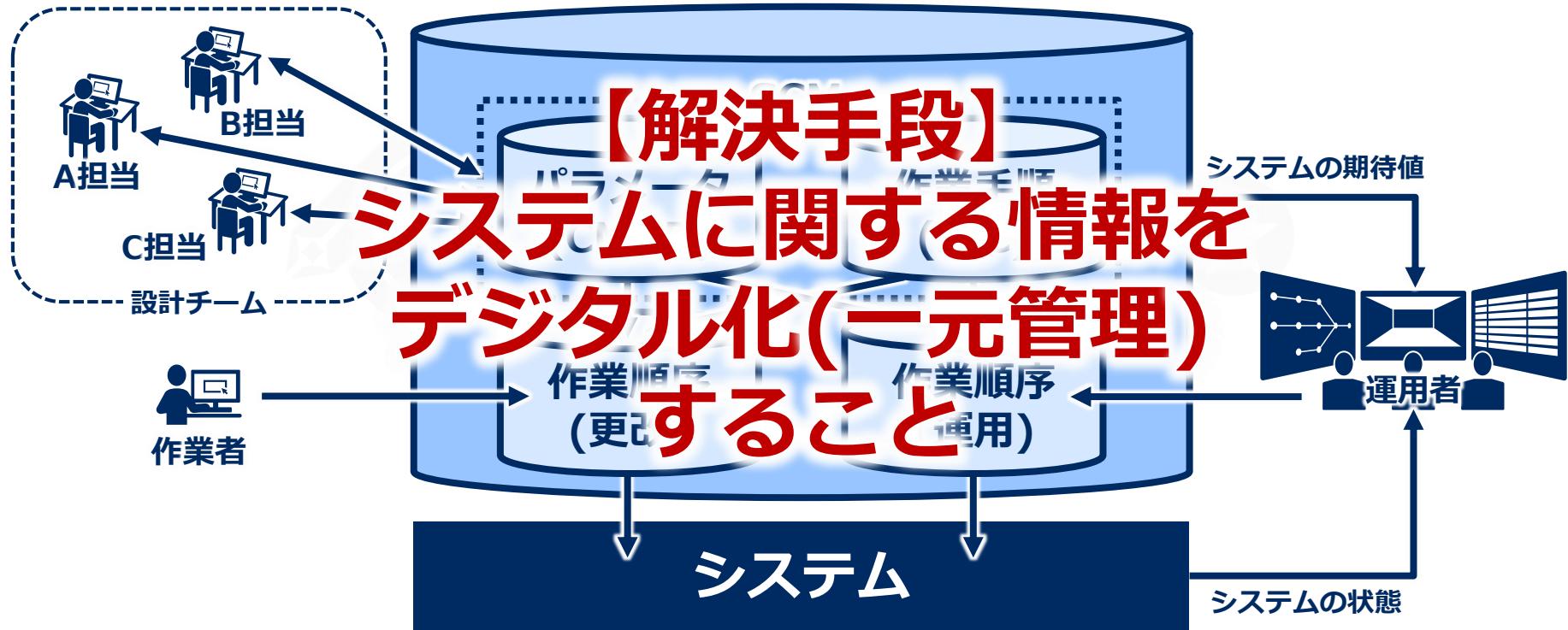
属人化

- 有識者不在により作業が進まない
- 有識者がいなくなるとノウハウは消失する
- 既知事象/未知事象の切り分けが難しく有識者の経験に頼らざるを得ない
- 結果として有識者を異動させられない

関係者はアナログ情報を正確に伝えることに多くの時間を費やしています



システムに関する情報をデジタル化して一元管理すればよいのですが…

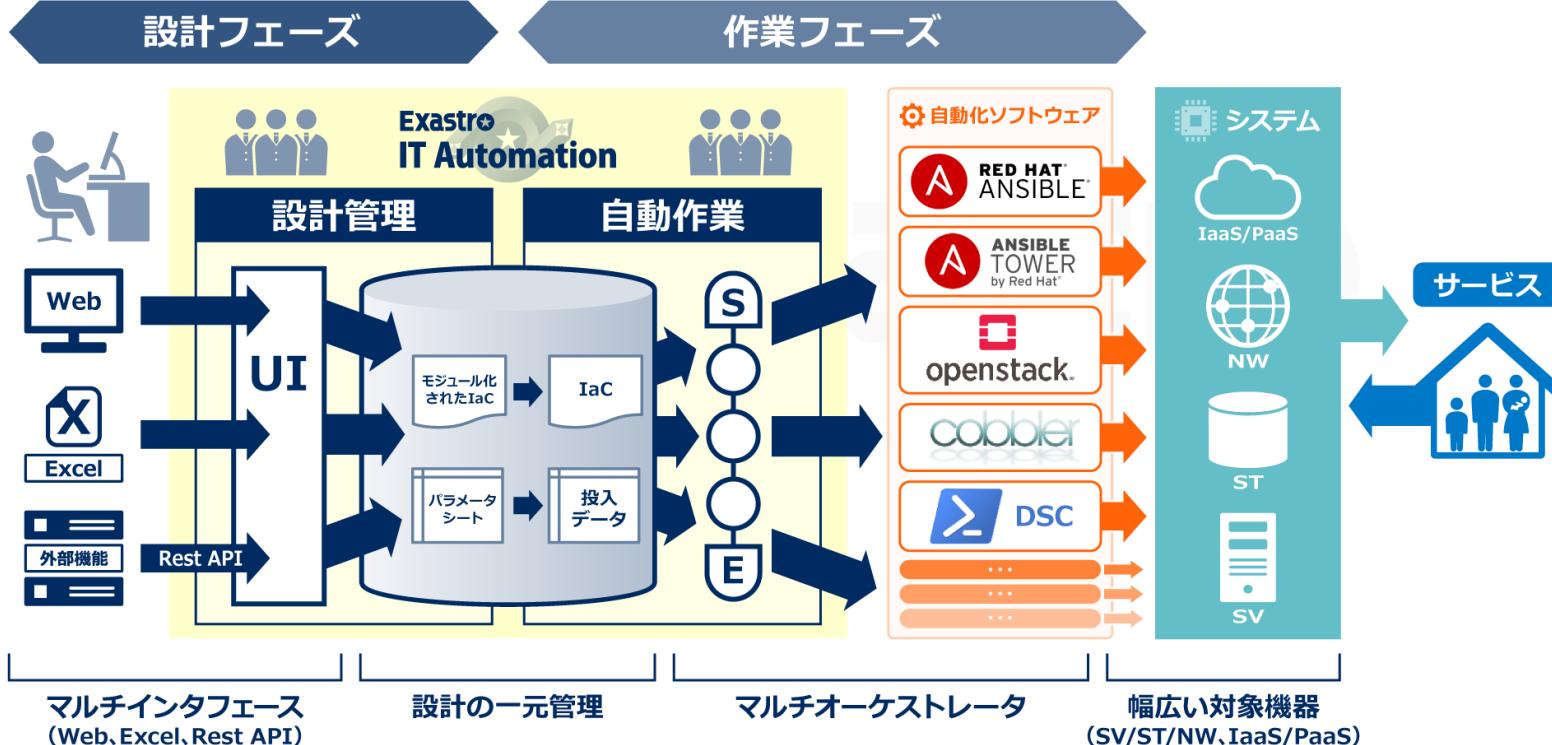


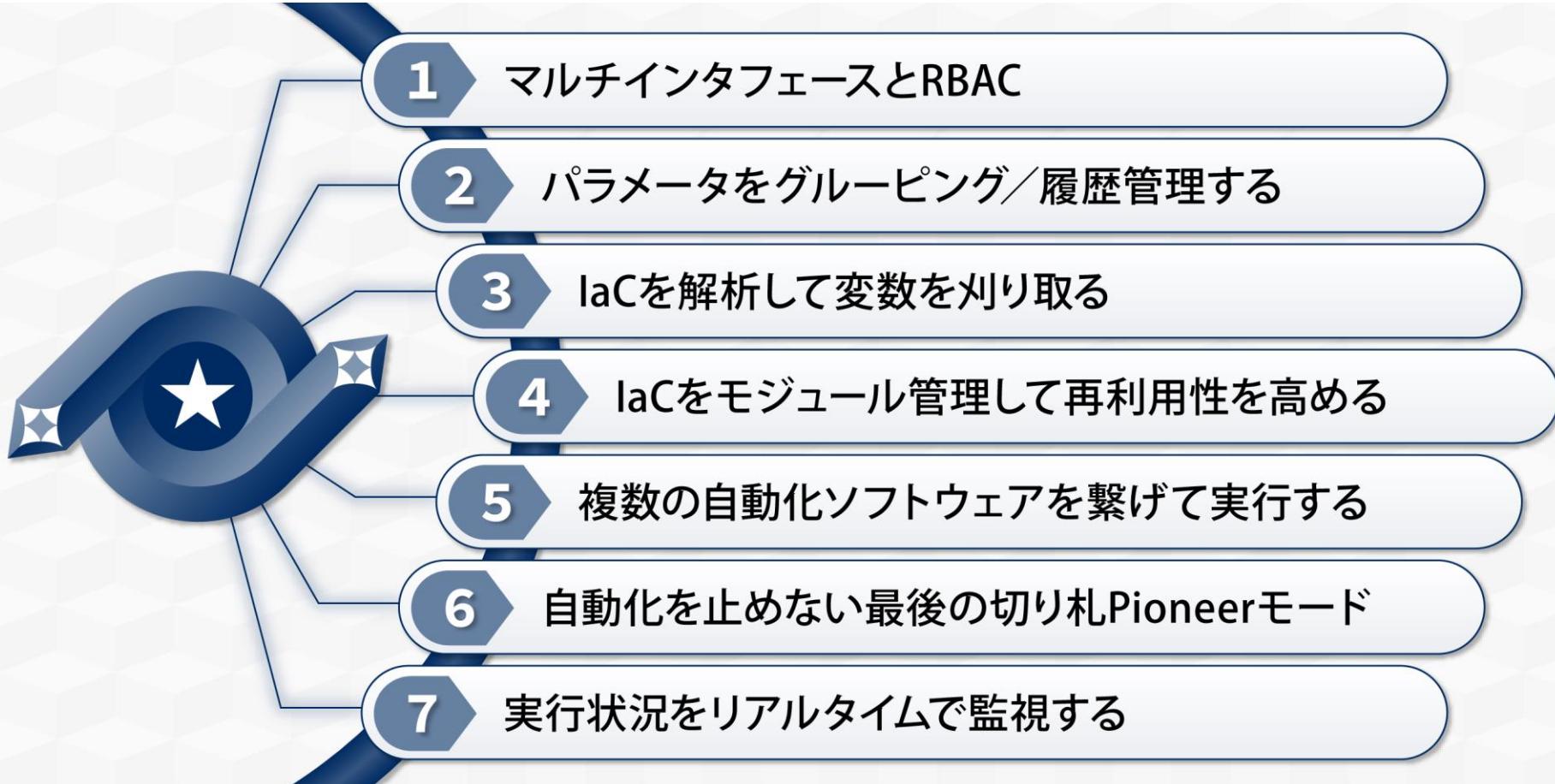
ITAは「システム情報をデジタル管理するためのフレームワーク」です



Exastro IT Automation (ITA) のご紹介

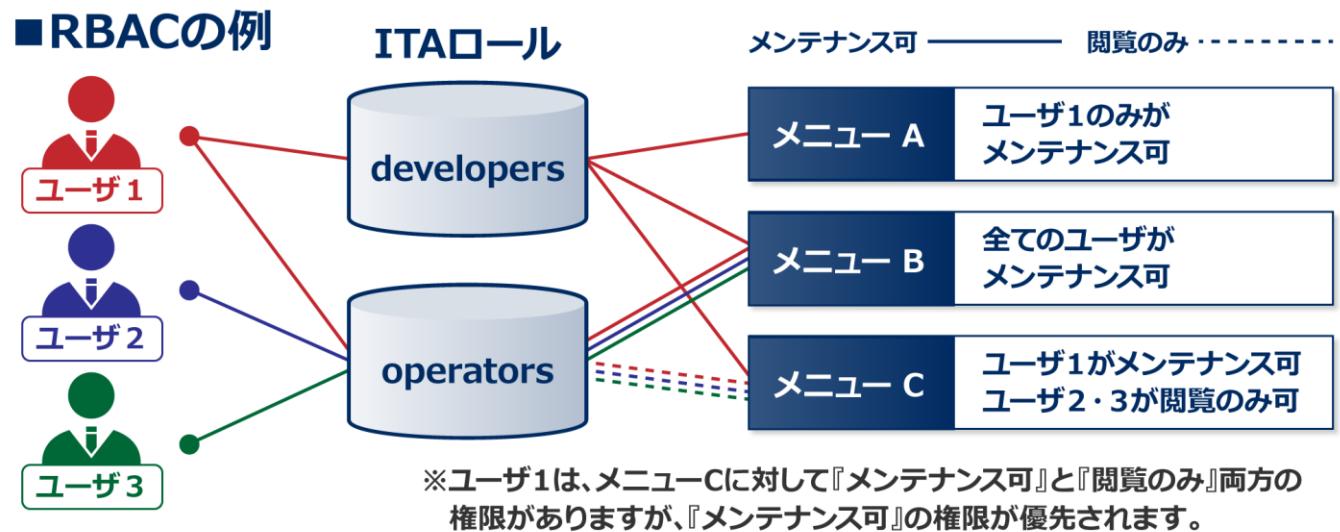
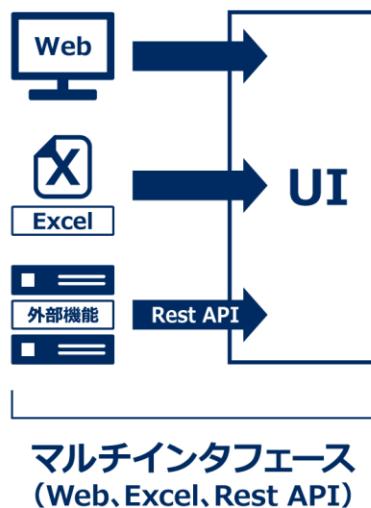
ITAは「システム情報をデジタル管理するためのフレームワーク」です





7つの特徴：①マルチインターフェースとRBAC

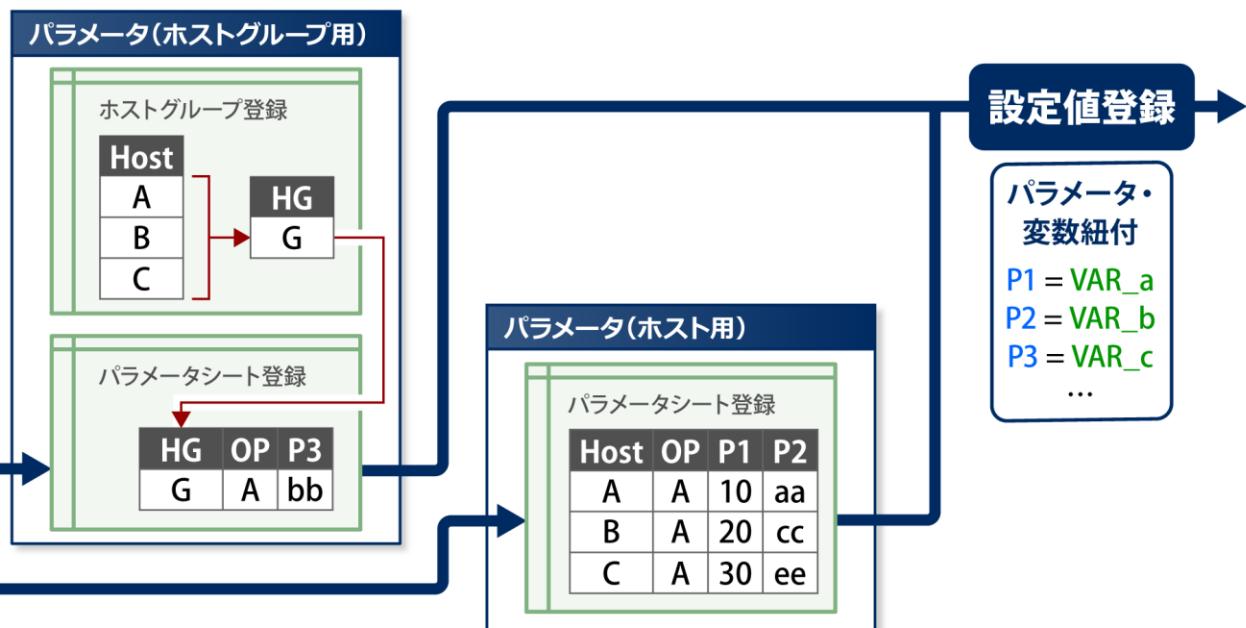
ユーザ操作を3種類のI/F(Web, Excel, RestAPI)から実行できます。
またどのI/Fからの操作でも「誰が・いつ・何をしたか？」を記録します。
RBACを備えており、開発者、作業者、運用者といった役割りを定義でき、
その役割りごとに出来ること(参照のみ、更新、実行)を制御できます。



システムのパラメータ情報をグルーピング／履歴管理できます。
 (履歴管理の重要性については後ほどディープダイブでご紹介します)

設計結果(パラメータ)

Host	OP	P1	P2	P3
A	A	10	aa	bb
B	A	20	cc	bb
C	A	30	ee	bb



7つの特徴：③IaCを解析して変数を刈り取る

ITAではIaCがアップロードされるとまずIaCに誤りが無いか解析します。
誤りがなければ、IaCの記述から変数名を刈り取って管理します。
変数名を選択式で利用するので誤植等のヒューマンエラーは起きません。

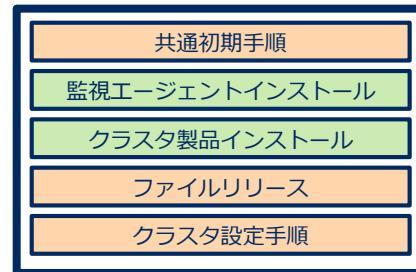


ITAではIaC(Playbook等)を一発モノで終わらせず再利用して利用し続けられるように、モジュール化して作業時に組み立てることが可能です。

Webサーバ構築手順



APサーバ構築手順



DBサーバ構築手順



共通の手順はモジュール化し再利用できるように管理する

共通初期手順

ファイルリリース

DB設定手順

クラスタ設定手順

監視エージェントインストール

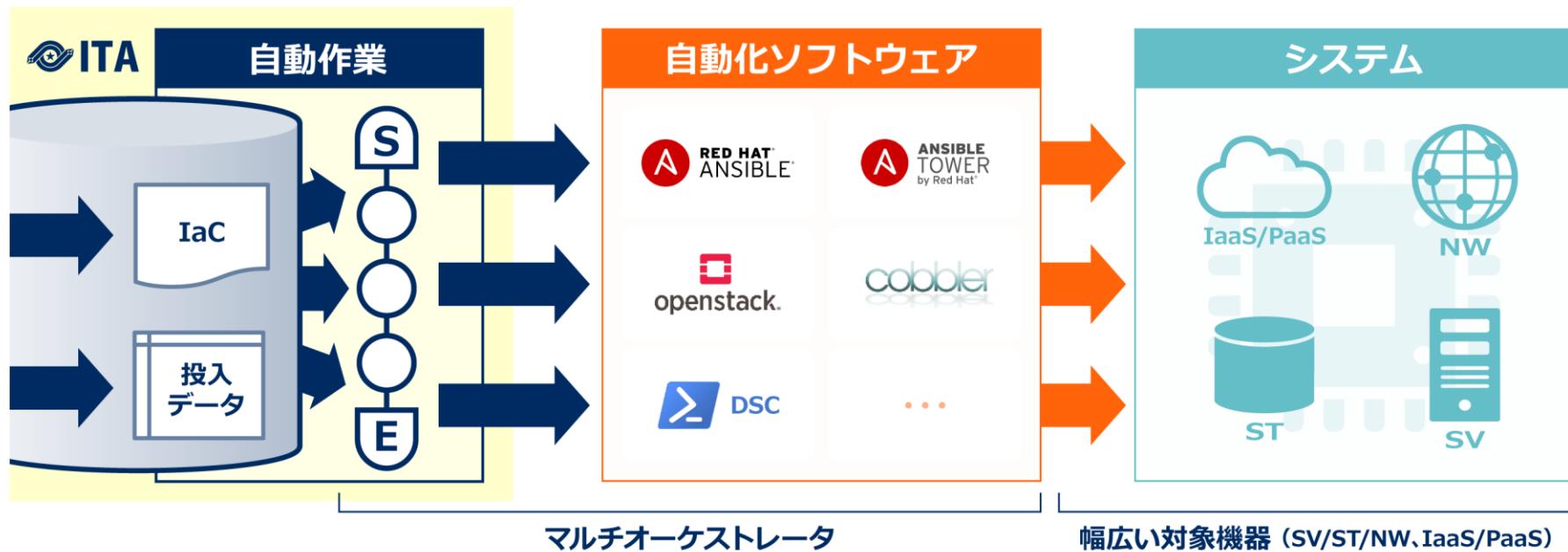
クラスタ製品インストール

HTTPサーバインストール

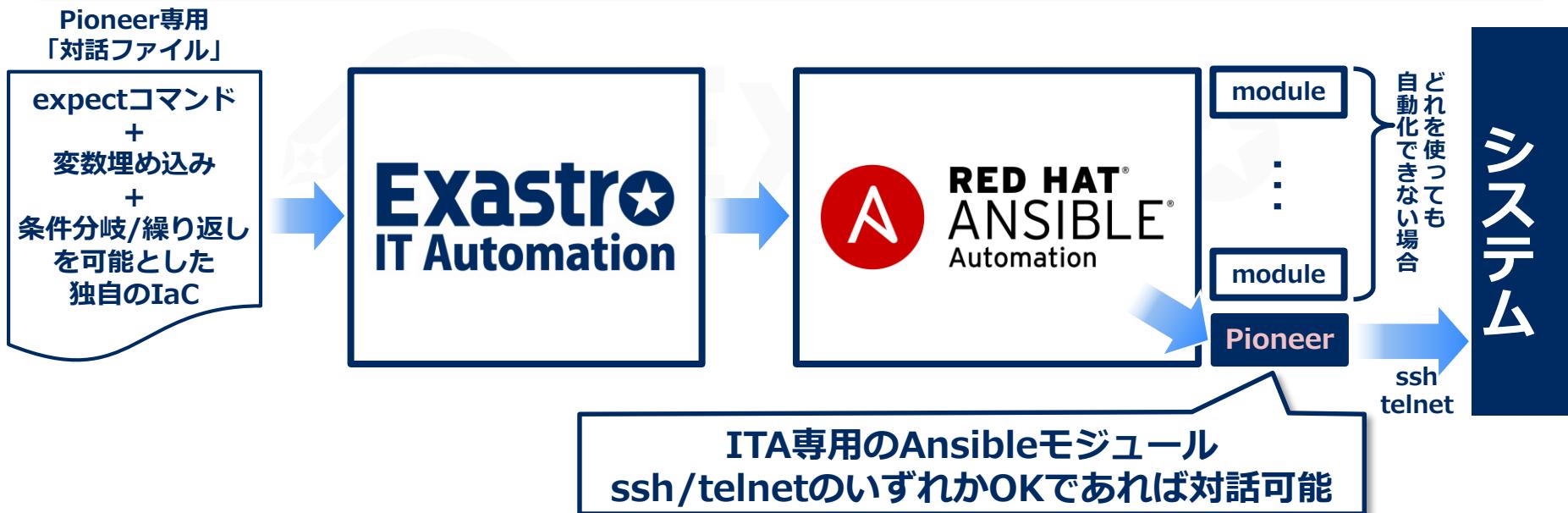
DBMSインストール

Exastro Playbook Collection

複数の自動化ソフトウェアを繋げて一本の作業フローを定義できます。
また自動化ソフトウェアの動作に必要な投入データを自動生成します。
例) (Ansibleの場合) 必要なPlaybookを集めて繋げ、ノード毎にパラメータからhost_varsを作る



Ansibleのどのモジュールを使っても自動化できない場合に、手動作業を挟んでしまうと自動化のメリットが半減します。そこで、**自動化を止めない最後の切り札として、ITAではPioneerモードをご用意しています。**



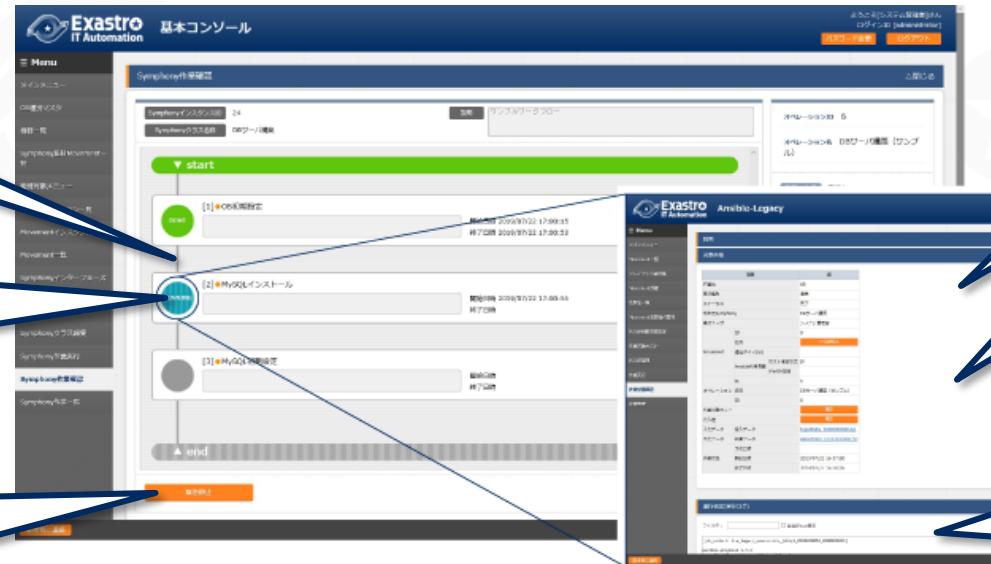
7つの特徴：⑦実行状況をリアルタイムで監視する

ITAは手動作業と比較して遜色なく実行状況をリアルタイム把握できることを重視しています。また実行結果(作業エビデンス)を欲しい時にダウンロードできるなど、作業記録をしっかりと管理します。

作業フローの途中に
「保留ポイント」
を設定可能

実行状況をクリック
すればドリルダウン
が可能

非常時には「緊急停
止」で作業をストッ
プすることが可能

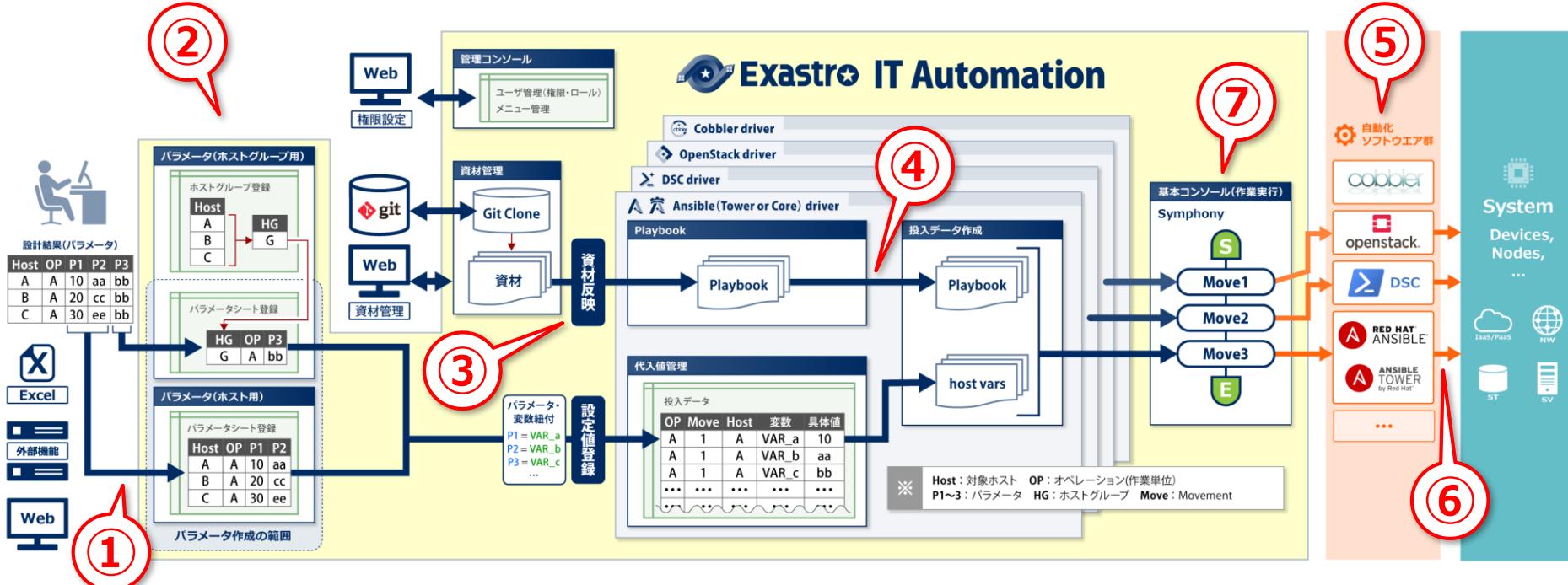


投入データ(自動生成)
がダウンロード可能
(zip)

実行結果(作業エビデ
ンス)がダウンロード
可能(zip)

自動化ソフトウェア
の実行状況をリアル
タイムで表示

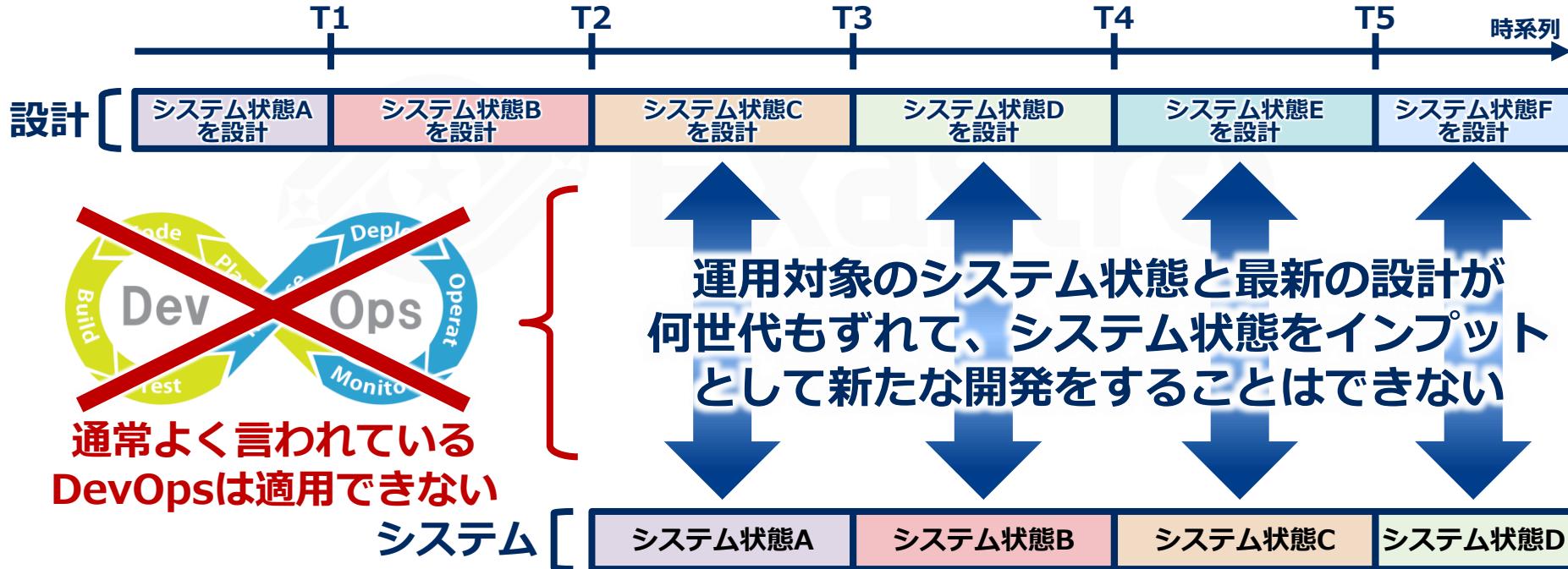
ご紹介した「7つの特徴」の他にも様々な工夫を凝らして、システム情報をデジタル管理できるようにしています。



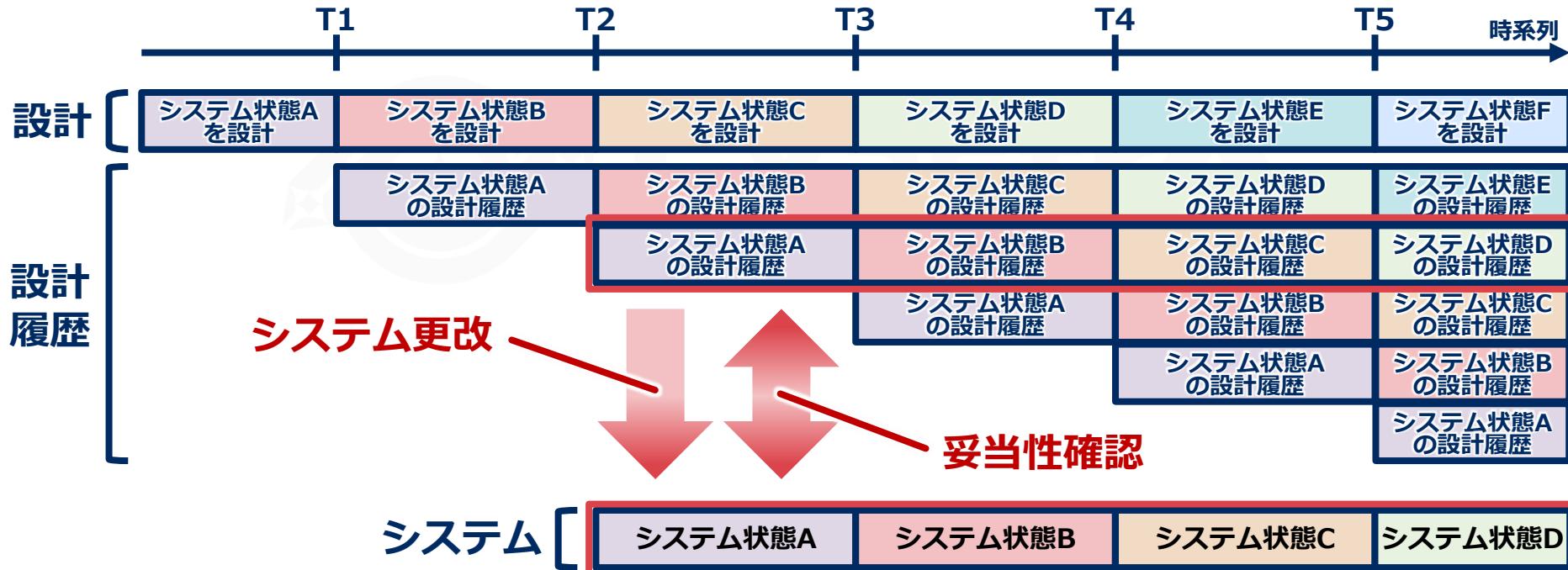
Exastro IT Automation (ITA) を少しディープダイブ

プラットフォームのDevOpsとパラメータの履歴管理

プラットフォームの設計者は数ヶ月先のシステムを設計しています。
設計と適用の時期が何ヶ月も空くので通常のDevOpsは適用できません。



プラットフォームの設計履歴の中には現在のシステム構成と同じ状態があり
これを使って「システム更改」や「システムの妥当性確認」ができます。



ITAで提供しているパラメータシートは履歴管理機能が付いています。
履歴から抽出したシステムの期待値を使ってシステム更改する仕組みです。

ITAの履歴管理機能つきパラメータシート

ホスト	オペレーション		パラメータ				設計日
	日時	作業名	P1	P2	P3	...	
hostA	12/20	クリスマス対応	1024	512	2048	...	10/1
hostA	10/9	hostB増設	512	256	1024	...	8/3
hostA	9/3	システムリリース	256	128	512	...	7/7
hostB	12/20	クリスマス対応	16	32	64	...	10/1
hostB	10/9	hostB増設	32	64	128	...	8/3

設計者は設計に集中できる

今日(10/9)で
パラメータを
抽出すると運用者は
運用に
集中できる

本日のシステムの期待値

ホスト	パラメータ				設計日
	P1	P2	P3	...	
hostA	512	256	1024	...	8/3
hostB	32	64	128	...	8/3

システム更改 妥当性確認

システム

つまり、ITAで実現するプラットフォームのDevOpsとは？

