攻めのITにつなげる COBOLモダナイゼーション

～IT資産の健全性の回復から、
API化によるクラウド移行まで～

ITモダナイゼーションSummit Web Live 2023

2023年4月13日（木）

TSH 東京システムハウス株式会社

目次

- **はじめに** … 会社紹介、サービス紹介
- **COBOLの現状とモダナイゼーションでやるべきこと**
  - COBOLシステムの課題、IT資産の健全性の回復について
- **攻めのITにつなげるCOBOLモダナイゼーション**
  - 守りのITから攻めのITへの転換について
- **まとめ**

はじめに

会社概要

商号	東京システムハウス株式会社（略称：TSH）
設立	1976年11月
資本金	1億7,990万円 （払込資本）
売上高	30.0億円（2022年10月期）
従業員数	174名（2022年10月現在）
本社	東京都品川区西五反田8-1-5 五反田光和ビル5F
特色	独自技術に特化したソリューション・サービス。 ニッチな業種向けの専用パッケージ。どの企業系列にも属さない独立系。 コンピュータ利用に関する総合サービスを提供。

TSH3.0
FOR THE 50TH ANNIVERSARY

マイグレーションの取り組み

MMS Mainframe Migration Service

1995年から蓄積された
経験・ノウハウと
230件以上の導入実績

Advanced Joint Tool AJTOOL

代替フレームワーク
「AJTOOL」の開発と提供

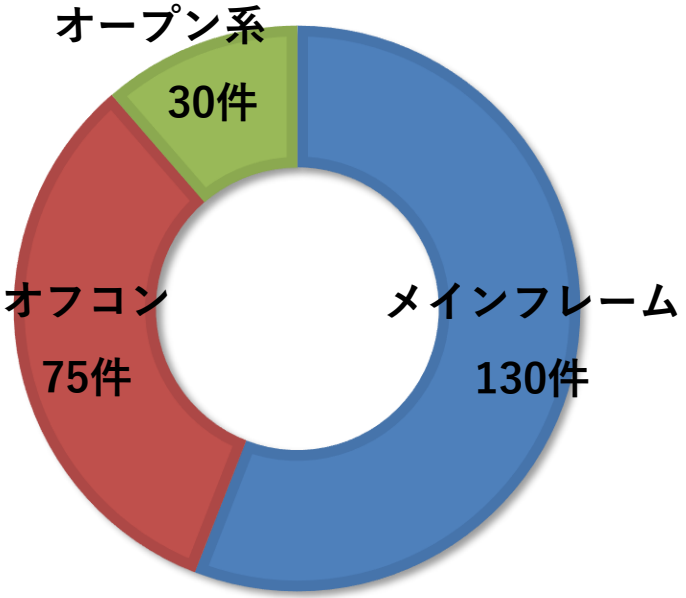
MMS+Cloud

COBOLシステムの
クラウドネイティブ移行
に対応

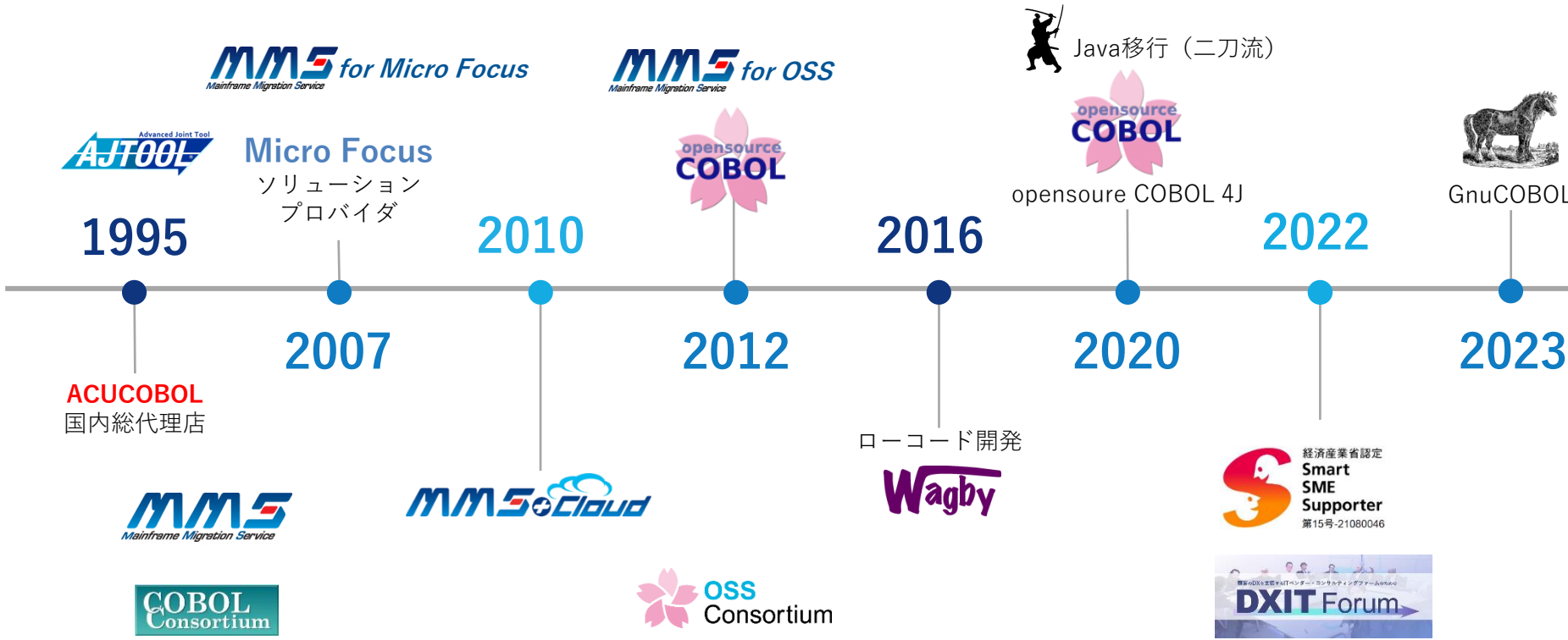
OSS Consortium

コミュニティでの
開発貢献と基幹系での
OSS利用の普及活動

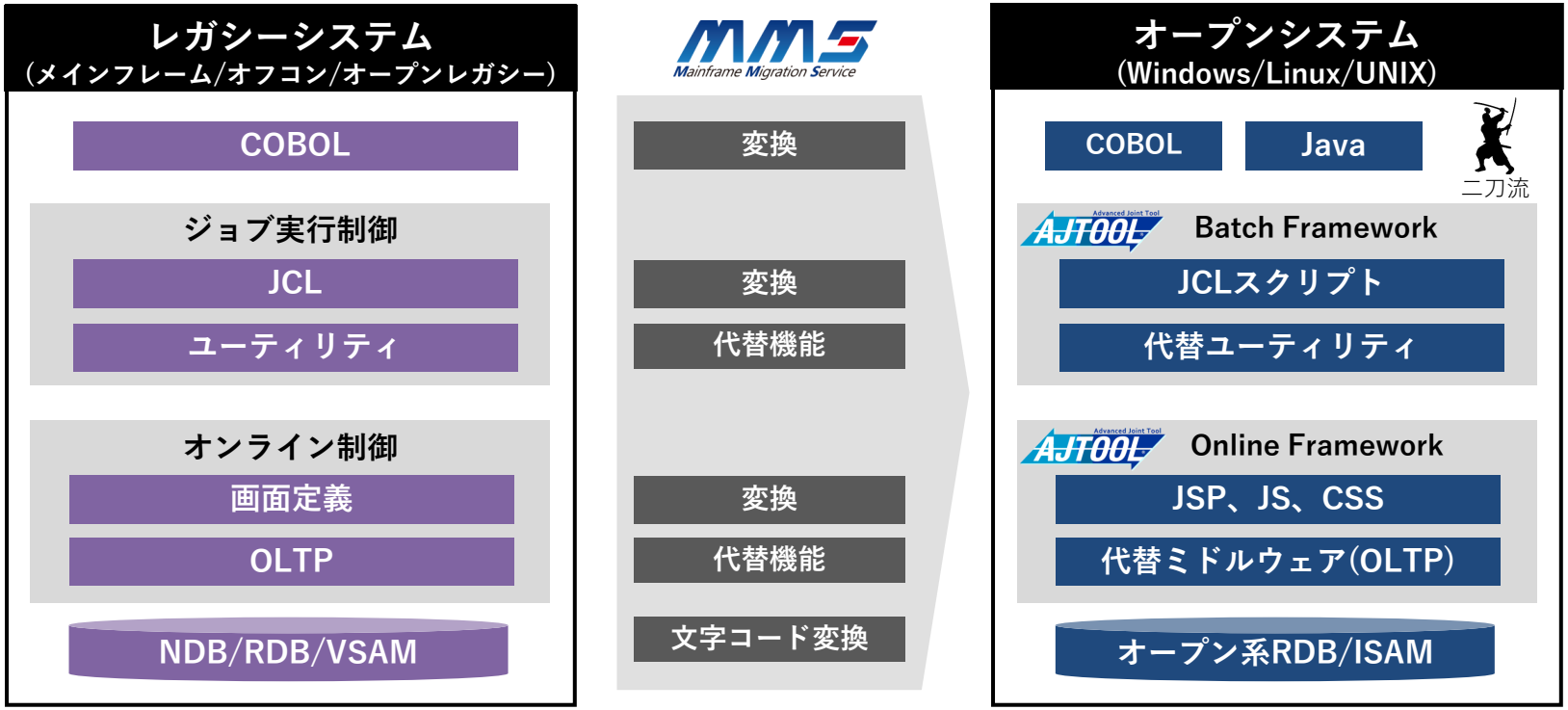
マイグレーション実績
(1995年～)



マイグレーションの取り組み



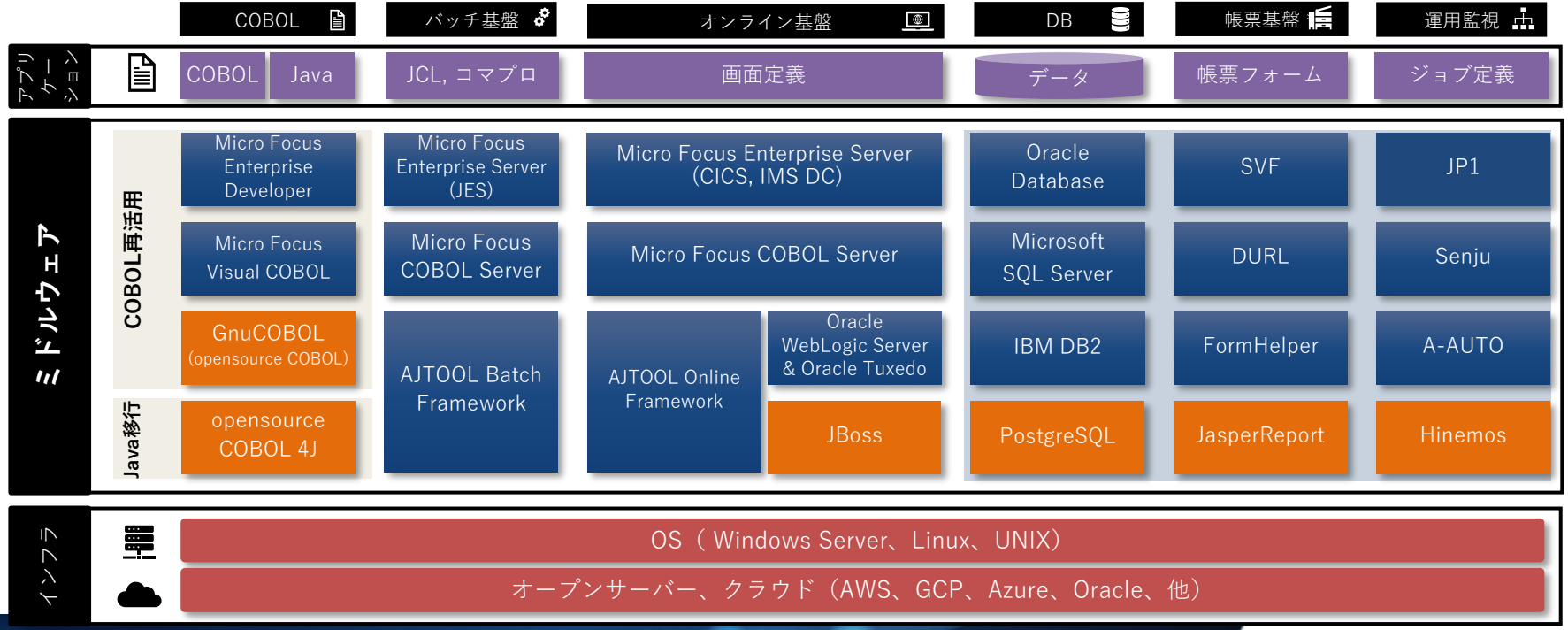
レガシーマイグレーション（資産移行）



レガシーマイグレーション（環境移行）

お客様に合わせた最適な処理方式とミドルウェアへ移行します。

商用製品　OSS



東京システムハウス株式会社

マイグレーションの工程

MMS Mainframe Migration Service

分析・設計フェーズ

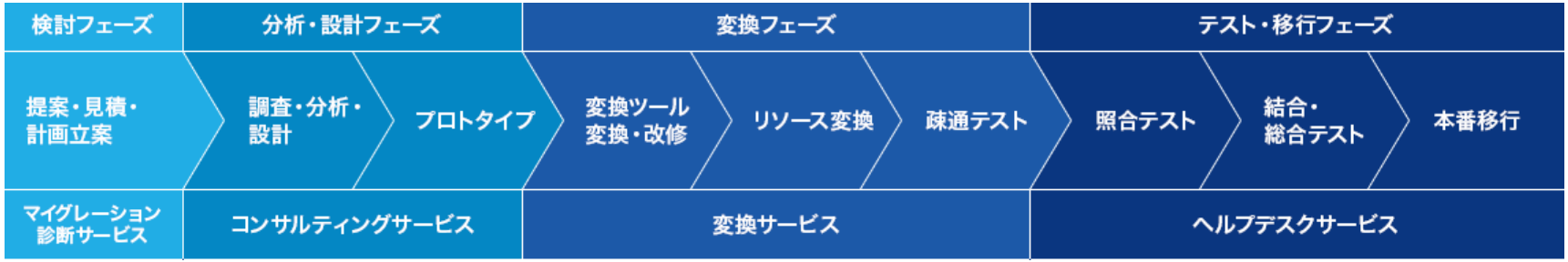
- **資産分析**から課題を抽出し、**網羅的解決**を検討する
- 一部資産でプロトタイプを作成し、移行性や機能性および性能の検証を実施

変換フェーズ

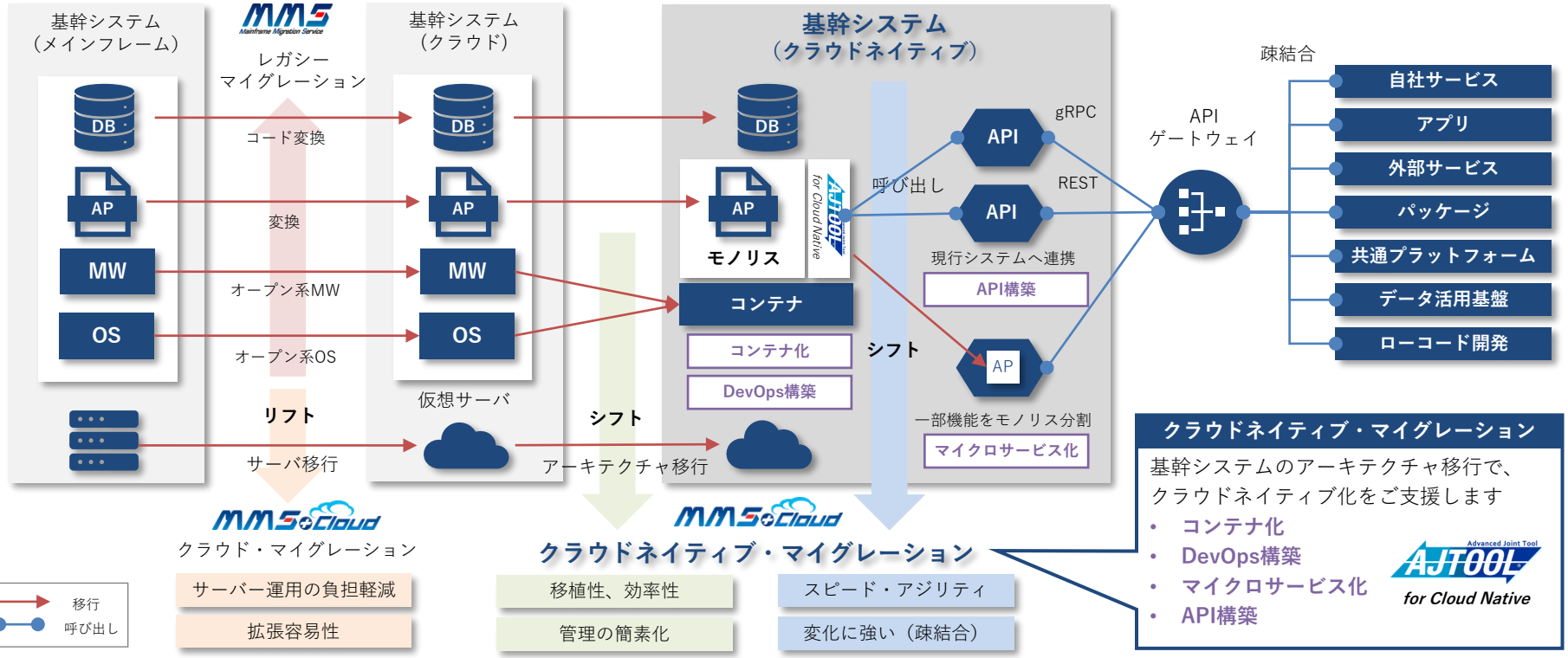
- **変換ツールでの自動変換**や一部の手直しと、代替機能の製造や単体テストを実施
- 疎通テストで変換結果の簡易検証を行う

テスト・移行フェーズ

- **照合テストで現新比較**を行い、移行の正当性を確認
- 総合テストで運用を見据えた動作確認を実施



クラウドネイティブ・マイグレーション



クラウドネイティブ・マイグレーション

基幹システムのアーキテクチャ移行で、
クラウドネイティブ化をご支援します

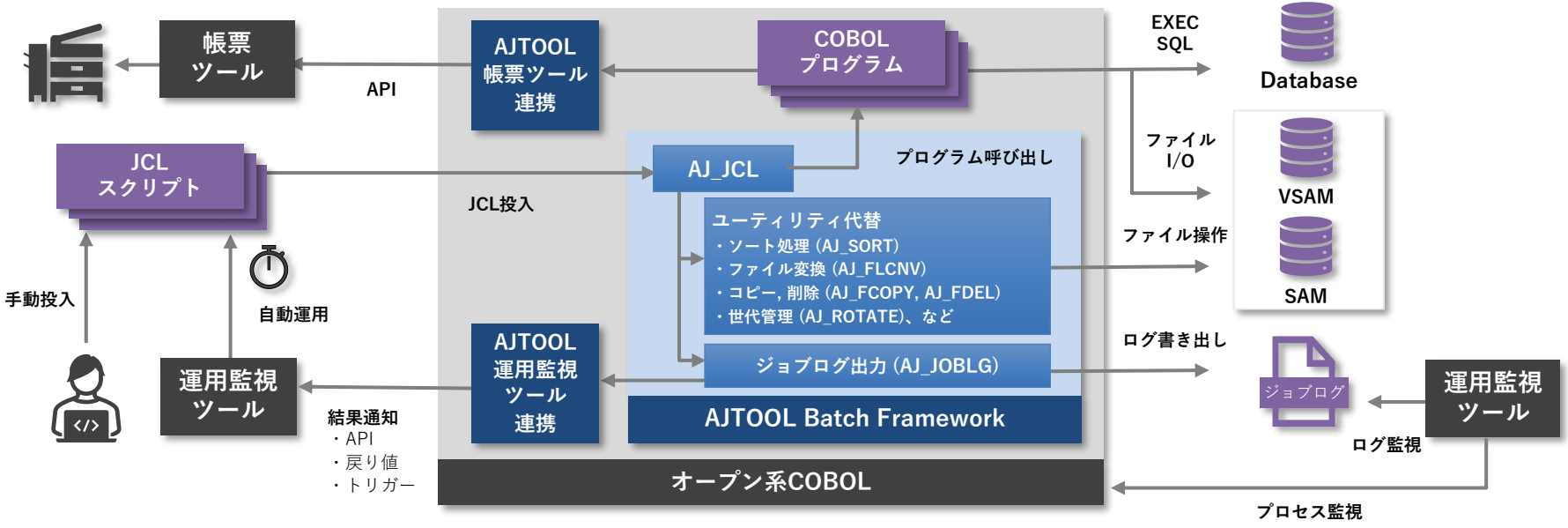
- コンテナ化
- DevOps構築
- マイクロサービス化
- API構築

AJTOOL Advanced Joint Tool for Cloud Native

AJTOOL Batch Framework

バッチ処理の実行フレームワーク

- JCLスクリプトの実行制御 AJ_JCL とユーティリティの代替機能を提供します



AJTOOL Batch Framework

Advanced Joint Tool AJTOOL

- JCLスクリプトの記述例

```
#-------------------------#
#  償還予定表  印刷処理
#-------------------------#
# ST001 最新データ抽出
>ST001
¥STEP LPPG01
@FILE SYS010 FN=dat/MCST
@FILE SYS020 FN=dat/ST001.OUT
¥EXEC

# ST002 ソート（出力対象）
>ST002
@JUMP RETURN-CODE /= 0 ST999
@PARAM
IF=(FN=ST001.OUT,RECL=64)
OF=(FN=ST002.OUT,RECL=64)
SEL=(17,6,N,EQ,@202212@)
KEY=(1,8,N,A)
@PEND
¥AJ_SORT
```

```
# ST003 償還予定表出力
>ST003
@JUMP RETURN-CODE /= 0 ST999
¥STEP LPPG02
@FILE SYS010 FN=dat/MCST
@FILE SYS020 FN=dat/ST002.OUT
@FILE SYS040 FN=dat/LPJ01C.TXT
¥EXEC
@IF RETURN-CODE /= 0
  @DISPLAY 印刷に失敗しました
  @GOTO ST999
@END-IF

# * ST900 終了処理
>ST900
¥AJ_FDEL ST002.OUT

>ST999
@END
```

AJTOOL Batch Framework

Advanced Joint Tool AJTOOL

- JCL実行結果のジョブログ例

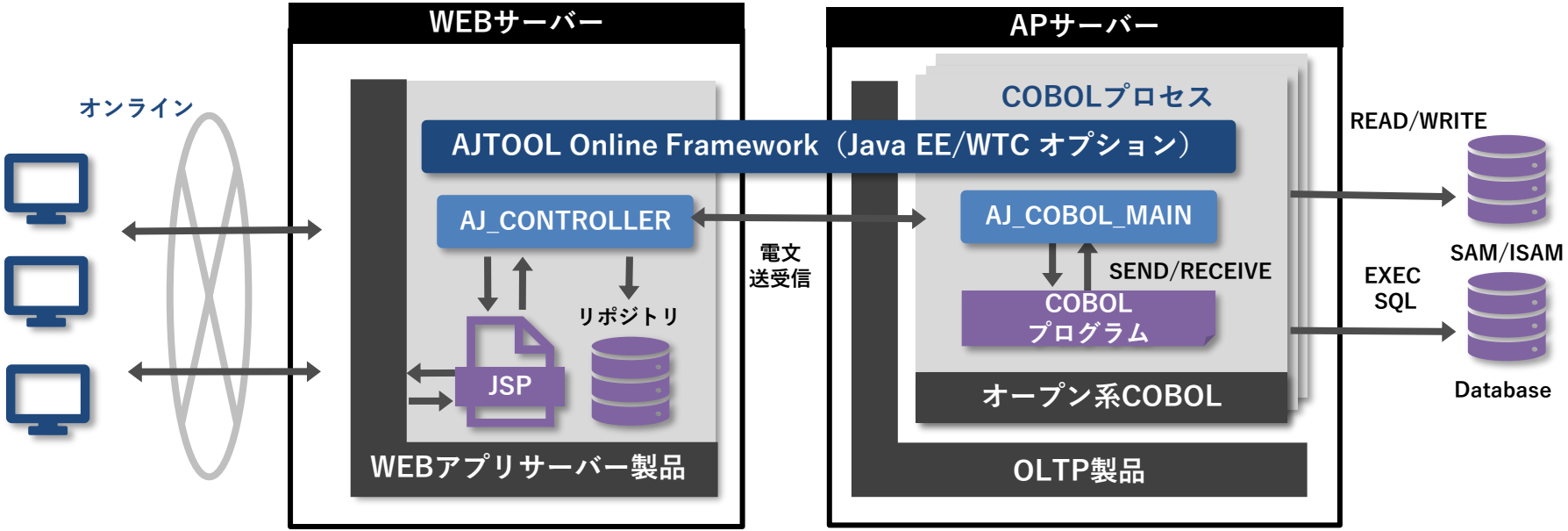
```
[~/cobol4j_batch_demo] ./gc_runjcl.sh LPJ01C.jcl
AJ_JCL_001 *** AJ_JCL 開始  ***:(jcl/LPJ01C.jcl)
AJ_JCL_007 00013:>ST001
AJ_JCL_007 00017:¥EXEC(¥STEP LPPG01)
AJ_JCL_004 STEP=000001----22年10月25日 : 22時35分19.46秒      開始
最新データ抽出処理完了：  9件
AJ_JCL_005 ---------------22年10月25日 : 22時35分19.46秒 正常終了
AJ_JCL_007 00022:>ST002
AJ_JCL_007 00030:¥AJ_SORT
AJ_JCL_004 STEP=000002----22年10月25日 : 22時35分19.46秒      開始
AJ_SORT_004 SORT入力ファイル：dat/ST001.OUT
AJ_SORT_005 SORT入力件数：  0000000009
AJ_SORT_006 SORT出力ファイル：dat/ST002.OUT
AJ_SORT_007 SORT出力件数：  0000000003
AJ_SORT_001 SORT正常終了
AJ_JCL_005 ---------------22年10月25日 : 22時35分19.46秒 正常終了
```

```
AJ_JCL_007 00035:>ST003
AJ_JCL_007 00041:¥EXEC(¥STEP LPPG02)
AJ_JCL_004 STEP=000003----22年10月25日 : 22時35分19.46秒      開始
AJ_JCL_005 ---------------22年10月25日 : 22時35分19.47秒 正常終了
AJ_JCL_007 00050:>ST900
AJ_JCL_007 00051:¥AJ_FDEL ST002.OUT
AJ_JCL_004 STEP=000004----22年10月25日 : 22時35分19.47秒      開始
AJ_JCL_005 ---------------22年10月25日 : 22時35分19.47秒 正常終了
AJ_JCL_007 00053:>ST999
AJ_JCL_002 *** AJ_JCL 終了  ***
AJ_JCL_006 ***** 開始時間   22年10月25日 : 22時35分19.45秒
AJ_JCL_006 ***** 終了時間   22年10月25日 : 22時35分19.47秒
```

AJTOOL Online Framework

メインフレーム型オンラインのフレームワーク（EJB/WTC オプション）

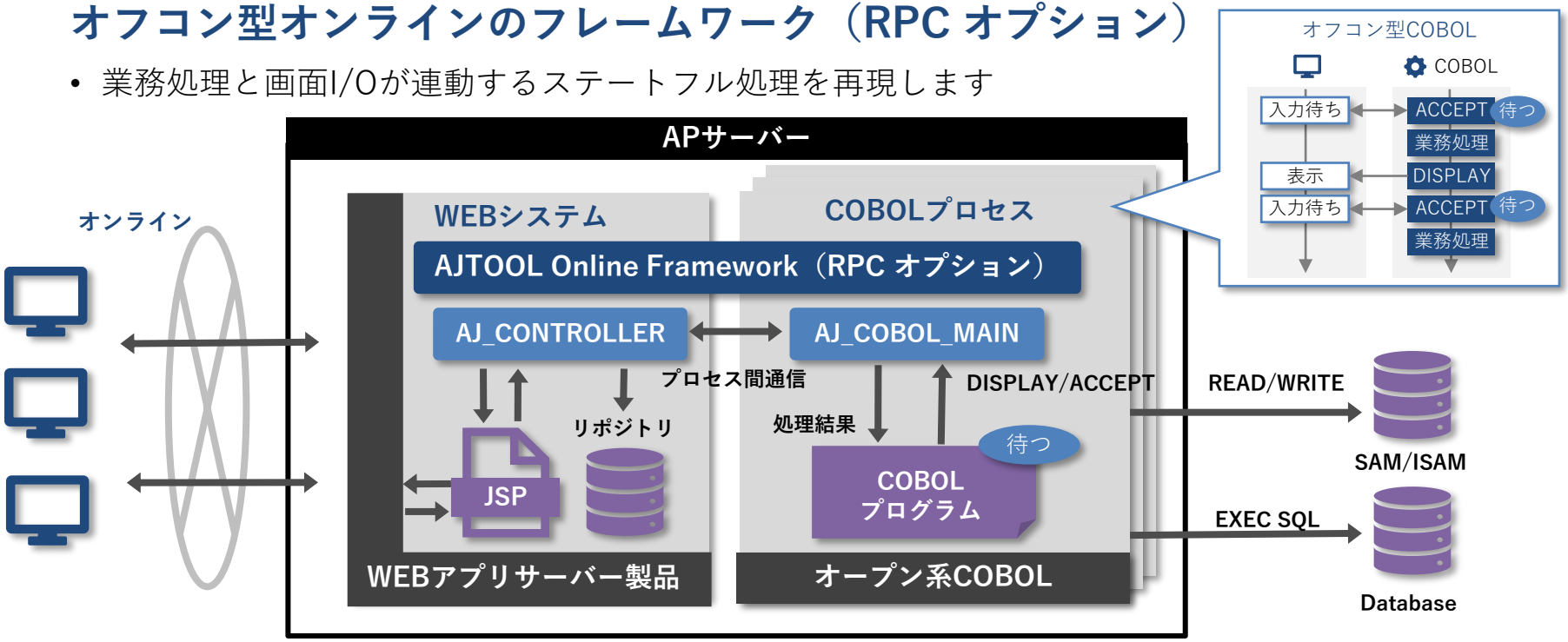
- WEB⇒OLTP⇒COBOLが連動するステートレス処理を実行します



AJTOOL Online Framework

オフコン型オンラインのフレームワーク（RPC オプション）

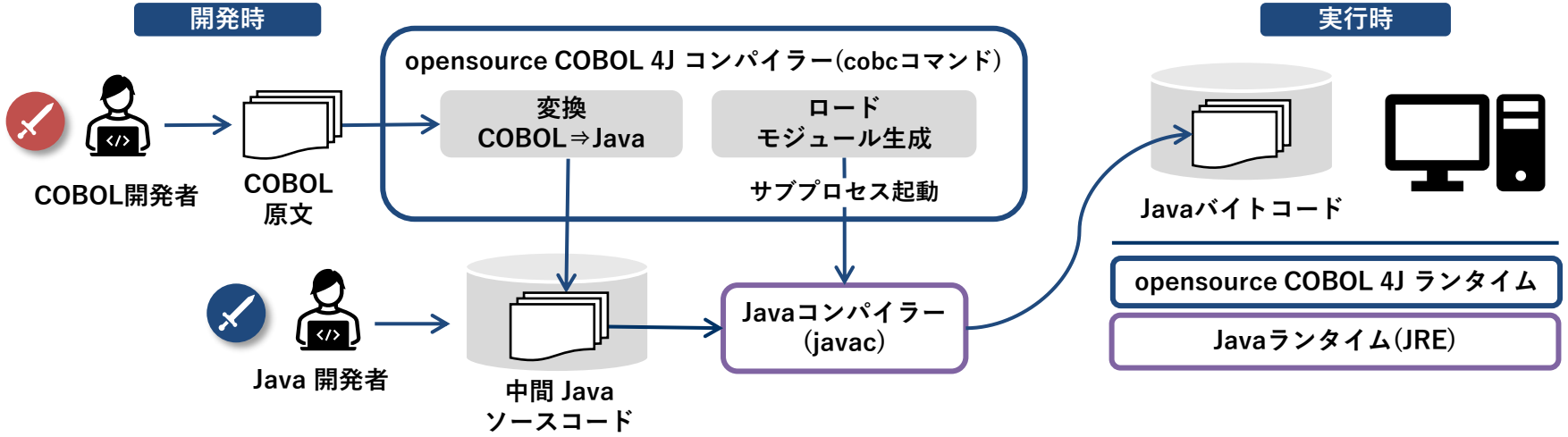
- 業務処理と画面I/Oが連動するステートフル処理を再現します



opensource COBOL 4J

https://github.com/opensourcecobol

- opensource COBOL シリーズの新しいプロジェクト（OSSコンソーシアム）
- COBOLをトランスレートして Javaを生成、javacでバイトコードを生成
- 元祖「opensource COBOL」を Java にマイグレーション



二刀流移行手法

https://github.com/opensourcecobol

opensource COBOL 4J は Java変換 と COBOL継続 の 二刀流 が可能

Java変換

opensource COBOL

opensource COBOL 4J

COBOL継続

Java変換の実現

COBOLコンパイル過程で中間Javaソースを生成する。
Javaエンジニアによるメンテも可能。

OSSのCOBOLコンパイラ

NISTのCOBOL85テストを通過したCOBOLコンパイラである。
OSSで公開しており(GPL3)、誰でも取得して利用が可能。

ブラックボックスなし

中間Javaが利用するランタイムライブラリは
OSSで公開されている(LGPL3)。

COBOL文化の継承

COBOLの業務ロジックを100%再現。Javaでは手間がかかる
演算や固定長などのCOBOL文化をJava環境で実装できる。

Java移行方式の比較

https://github.com/opensourcecobol

- opensource COBOL 4Jはコンパイラとして提供される。
- 生成される中間Javaソースは可読性がある。
- 他のJava移行方式と異なり、COBOL継続とJava変換の二刀流が可能。

	Java移行方式	手法	出力	Java変換	Java可読性	COBOL継続	実績	コスト	技術者確保
1	変換ツール型 マイグレ時にJavaソース（Javaソース） に変換をする	リライト	Javaソース	○ 可能	◎ 良い	× 不可	◎ 多数	× 変換費用	○ Java多い
2	コンパイラ型 COBOLコンパイラが（バイトコード） バイトコードを生成する	リホスト	バイトコード	× 不可	ー 対象外	○ 可能	○ 安定	△ 製品費用	△ COBOL減少
3	コンパイラ型 COBOLコンパイラが（Javaソース） Javaソースを生成する	リホスト＋リライト	Javaソース	○ 可能	○ あり	○ 可能	△ これから	◎ OSS公開	◎ Java+COBOL

opensource COBOL 4J お試しください

ダウンロードはこちら
https://github.com/opensourcecobol/opensourcecobol4j

Qiita記事『openource COBOL4J はじめの一歩』
https://qiita.com/Hiroimon/items/a73e8bdec4b376916ca0

最新情報はTwitterでも発信中
https://twitter.com/tsh_mms

マイグレーション事例

- ITモダナイゼーションSummit 2022（昨年）
  - レビュー記事：
    日経クロステック Special
    『独自ツールの活用から“二刀流”まで顧客に合った“脱レガシー”を提供』
    https://special.nikkeibp.co.jp/atclh/NXT/22/it_modernization_summit0531/p7/
  - 講演資料：http://www.cobol.gr.jp/knowledge/material/220414_report/01.pdf
    - 日本製紙様（富士通ホストからのマイグレーション事例）
    - 北陸電機商会様（東芝TPCareからのマイグレーション事例）

COBOLシステムの現状と
これからするべきこと

社会基盤のアトラス「COBOL」

最近のCOBOL言語の話題

- 基本情報処理技術者試験からCOBOLの出題が終了
- COBOLは古い、新しいことできない･･･
- COBOLのシステムは複雑、やりたくない･･･
- COBOL技術者の引退、技術者不足
- 空前の脱COBOLブーム（Javaリライト）
- 今もCOBOL資産は世界に8000億行ある
  （Micro Focus社 2022年2月調査）

黙々と処理をこなして社会基盤を支える巨人COBOL
その姿はギリシャ神話で天空を背負うアトラス
COBOLが手を離せば天空が落ちてきて社会が大混乱する
（フランクフルト中央駅のアトラス像）

COBOL言語の問題ではなく「IT資産の健全性」の問題

社会基盤のアトラス「COBOL」

- 本来COBOLは保守性や変更容易性が高い言語
  - 国際規格で標準化、この1月に新規格公開（ISO/IEC 1989:2023） NEW
  - 英語を用いた分かり易い構文、事務処理計算の業務を抽象的に記述
  - 構造化プログラミング、コード再利用の仕組み
- 手入れ不足で不健全になり、維持・運用にコストと人手がかかる
  - **長年の増改築**（コピペ量産、念のため残した処理、その場しのぎのバイパス処理）
  - **標準でない記述**（ベンダー独自機能、特定機器向け制御コード）
  - 仕様書や**テストケースの消失**（元々は綺麗に整っていたはず）
  - COBOL技術者不足よりも深刻な**仕様理解者不足**（ベテラン引退、引継ぎ不足）

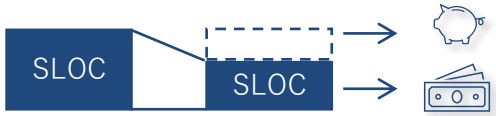
モダナイゼーションは「IT資産の健全性」の回復のチャンス

IT資産の健全性の回復

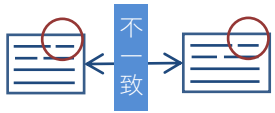
モダナイゼーションでのIT資産の健全性の回復のポイントは4点

1. プロジェクト開始前のリファクタリング
2. テクノロジとビジネスロジックの分離（再レガシー回避）
3. テストケースの再構築
4. 仕様理解者を増やす

- **無駄なコードは移行コストに直結する**



- **モダナイ中のリファクタリング・改修はNG。現新比較の不一致原因が増加してしまう。**



オリジナルバグ？
変換誤り？
データ誤り？
環境非互換?

＋リファクタ誤り？
改修誤り？

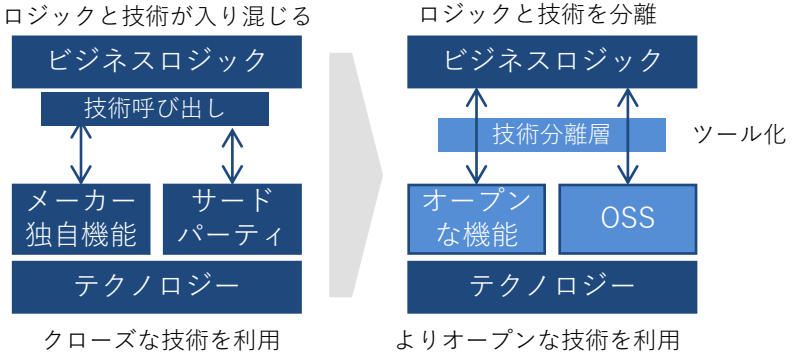
- **リファクタリングの要点**（保守性・変更容易性）
  - デッドコード、なぞ変数名、バイパス見直し
  - スパゲティは仕様理解者または解析ツール活用
  - 現行システム上でしっかり動作確認

IT資産の健全性の回復

モダナイゼーションでのIT資産の健全性の回復のポイントは4点

1. プロジェクト開始前のリファクタリング
2. **テクノロジとビジネスロジックの分離（再レガシー回避）**
3. テストケースの再構築
4. 仕様理解者を増やす

- **技術変化に左右されない設計**
  - ビジネスロジックがテクノロジーを自由に乗り換えられる仕組みにする



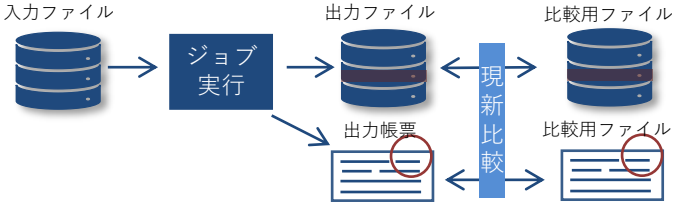
- **再レガシーの回避**
  - メーカー独自機能を互換製品で回避すると将来の再レガシーの原因となり得る

IT資産の健全性の回復

モダナイゼーションでのIT資産の健全性の回復のポイントは4点

1. プロジェクト開始前のリファクタリング
2. テクノロジとビジネスロジックの分離（再レガシー回避）
3. **テストケースの再構築**
4. 仕様理解者を増やす

- **現新比較テストはモダナイ後も維持すべき**
  - 構築当初のテストケースが手入れ不足 or 消失
  - モダナイは現新比較テストを必ず作る
  - 廃棄せず、せっかく作ったのだから維持をする



- **CI/CDパイプラインで現新比較を再活用する**
  - バージョン管理 + CIツールを導入する
  - クラウドのAWS code pipeline、Github Actionsなどで基幹系向けのCIを構築する

IT資産の健全性の回復

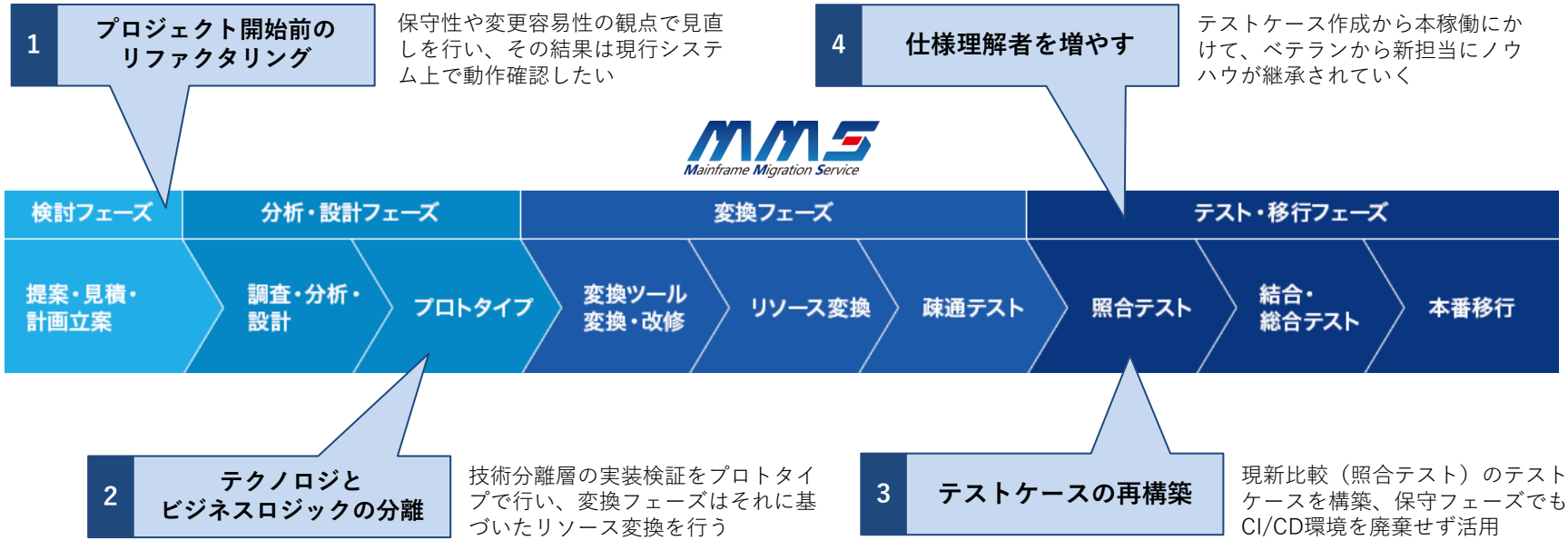
モダナイゼーションでのIT資産の健全性の回復のポイントは4点

1. プロジェクト開始前のリファクタリング
2. テクノロジとビジネスロジックの分離（再レガシー回避）
3. テストケースの再構築
4. **仕様理解者を増やす**

- **仕様理解のチャンスをベンダに丸投げしない**
  - 仕様理解とスキルセットへの投資と考える
  - スキルセットを回復して将来構想の足掛かりへ
- **ブラックボックスを紐解く ⇒ 結果が明白**
  - **分析設計**：棚卸、機能整理、サードパーティ整理
  - **現新比較**：シナリオ/データ作成、不一致の調査
  - **総合テスト**：運用・保守のテスト、シナリオの作成、ジョブスケジュール、外部連携の検証
- **その過程でベテランから新担当へのノウハウ継承**
  - 新担当はテストシナリオ/データ作成で現行を理解、次第に不一致の詳細調査で深部の仕様も把握
  - 若手の方がオープン系技術が得意なので、終盤にかけてウエイトは若手に移っていく

IT資産の健全性の回復

モダナイゼーションの各工程とポイント4点の発生個所



プラットフォームデジタル化指標（IPA）

DX対応に求められる要件を満たすかを評価するIPAが作成・公開した指標

PFデジタル化指標は大きく分けて ITシステム全体 と 機能システムごと から構成される。
さらにこれらは 属性情報 と 評価項目 から構成される。

対象	種別	大分類			項目数
ITシステム全体	属性情報	財務			5
	評価項目	機能システム間の独立性			12
		データ活用の仕組み			
		運用の標準化			
		ガバナンス		プロジェクトマネジメント、品質	
				セキュリティ、プライバシー	
				CIO、デジタル人材	
機能システムごと	属性情報	事業特性			13
		影響度			
		システム特性			
		保有リソース			
		IT資産の状況			
	評価項目	①DX対応に求められる要件		データ活用性	46
				アジリティ（ユーザ要件への対応）	
				アジリティ（非機能要件への対応）	
				スピード	
		②基礎的な要件	ITシステム品質	利用品質	
				開発品質	
			IT資産の健全性		

評価する上で把握しておくべき特性・状況など。

企業のITシステム全体を評価する。

目指すべき状態に対して現状がどのような状態であるかを評価。

企業のITシステム全体を構成する「機能システム」を評価。

DX実現に必要な変化への柔軟かつ迅速な対応、データ活用の度合を評価。

DX対応上で前提となる、ソフトウェア品質および技術的負債の状況を評価

引用元：DXIT Forum Open Seminar 「レガシー刷新と攻めの変革の両立を目指して(1)」
https://www.ipa.go.jp/event/2022/digital/20230222.html

COBOLシステムの精密検査をやってみると、
IT資産の健全性に課題が多い傾向

IT資産の健全性
1 ソフトウェア資産の最適化
2 不要なソフトウェア資産を増やさない
3 組織的な対応、設計内容の把握
4 適切な箇所での対応
5 再構築に必要な設計情報の維持・管理
6 ハードウェア製品のサポート継続性
7 ソフトウェア製品のサポート継続性
8 利用サービスの継続性

**PFD指標で精密検査をして
モダナイの過程で適切に対策**

まとめ

- COBOLを取り巻く課題は **IT資産の健全性** の問題
- モダナイゼーションはIT資産の健全性を**回復するチャンス**
- PFD指標で精密検査をして個別の課題を把握し適切に対策する

- IT資産の健全性の回復により、基幹系システムは適切なコストと人手で維持・運用され、既存事業を安定して支えていく

攻めのITにつなげる
COBOLモダナイゼーション

攻めのITと守りのIT

	定義	例えば
守りのIT 	・既存ビジネスの運用・維持に関わるIT投資（ラン・ザ・ビジネス） ・ITによる作業効率化やコスト削減を目指すことで、安定性や品質向上などを目標とする	・基幹系システム ・クラウド化 ・セキュリティ強化 ・業務自動化
攻めのIT 	・利益拡大や新規顧客獲得に関わる戦略的なIT投資（バリューアップ） ・ITを活用して新ビジネス創出や既存ビジネスの付加価値向上を目標とする	・ECサイト ・データ分析の活用 ・AI技術の活用 ・IoT技術の利用

基幹系システムは守りのIT・・・ 守りのITの中心で予算の比重も大きい、既存ビジネスの運用・維持が目的のため、ビジネス創出や付加価値向上に直接的な貢献はしない。

攻めのITと守りのIT

DXレポート（2018年9月）

『**ランザビジネス**に予算や人員が多く費やされており、
新しいビジネスや価値創出にむけた**バリューアップ**に
振り向ける余裕がなくなってしまう』

DXレポート2.2（2022年7月）

『DX推進に取り組むことの重要性は広がる一方で、
デジタル投資の内訳はDXレポート発出後も変化がなく、
既存ビジネスの維持・運営に約８割が占められている
状況が継続』

攻めのIT投資（バリューアップ） **約20%**

vs.

守りのIT投資（ランザビジネス） **約80%**

**4年が経過したが
変わっていない**

守りのIT投資から、攻めのIT投資への転換が必要

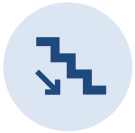
攻めのITと守りのIT

守りのITから攻めのITへの転換

IT資産の健全性の回復

- モダナイゼーションを通じて**IT資産の健全性を回復**させる
- 保守性・変更容易性の改善
- 開発アジリティの向上
- 仕様理解とスキルセットの回復

守りのITのコスト・工数削減 その分を攻めのITへ



- モダナイゼーションを完遂し、**基幹系システムを安定稼働**
- ITインフラを最新化・クラウド化
- **守りのITのコスト削減と工数削減**、その分を**攻めのIT投資へ**

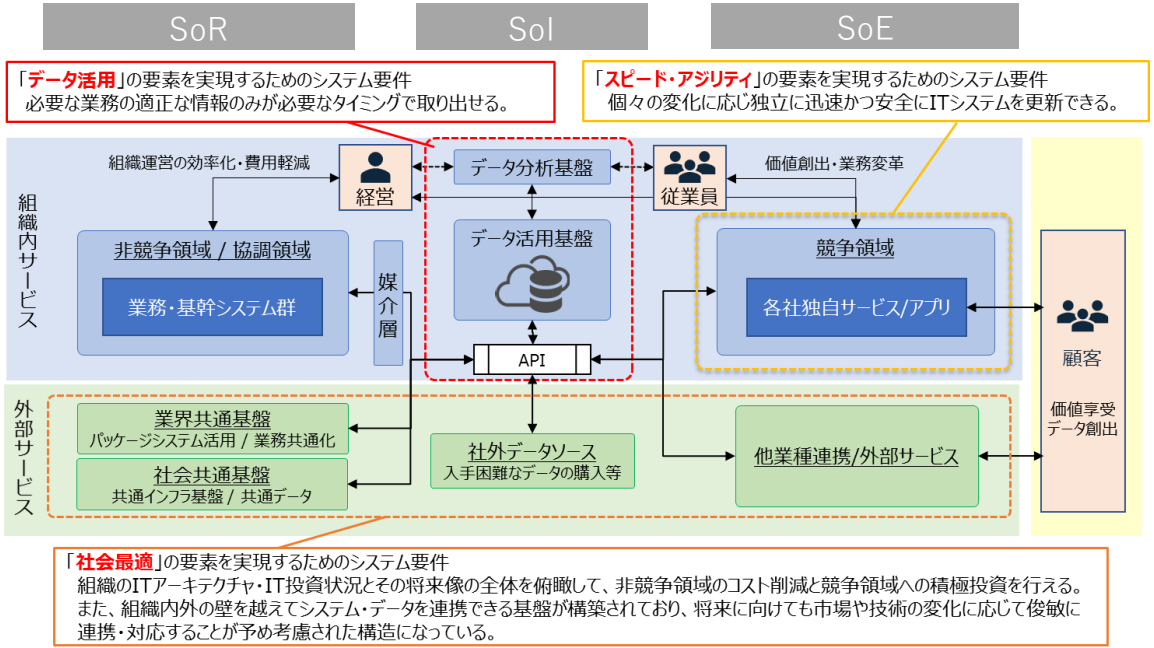
攻めのITと基幹系システムをつなげる

- モダナイゼーションは通過点
- 既存事業で長年蓄積したデータや業務ロジックなど、**基幹系システムの機能を攻めのITから利用可能に**することで**新しいビジネスや価値創出に貢献**

攻めのIT投資へ転換しつつ、基幹系とつなげて新しい価値を創出

攻めのITにつなげるには

スサノオフレームワーク（IPA『DX 実践手引書 IT システム構築編』）



引用元：DXIT Forum Open Seminar「レガシー刷新と攻めの変革の両立を目指して(1)」
https://www.ipa.go.jp/event/2022/digital/20230222.html

攻めのITにつなげるポイント3点

クラウドネイティブ移行

業務・基幹システム群もコンテナ化やCI/CD構築など、攻めのIT投資領域のシステムと同じ基盤で接続可能とすること

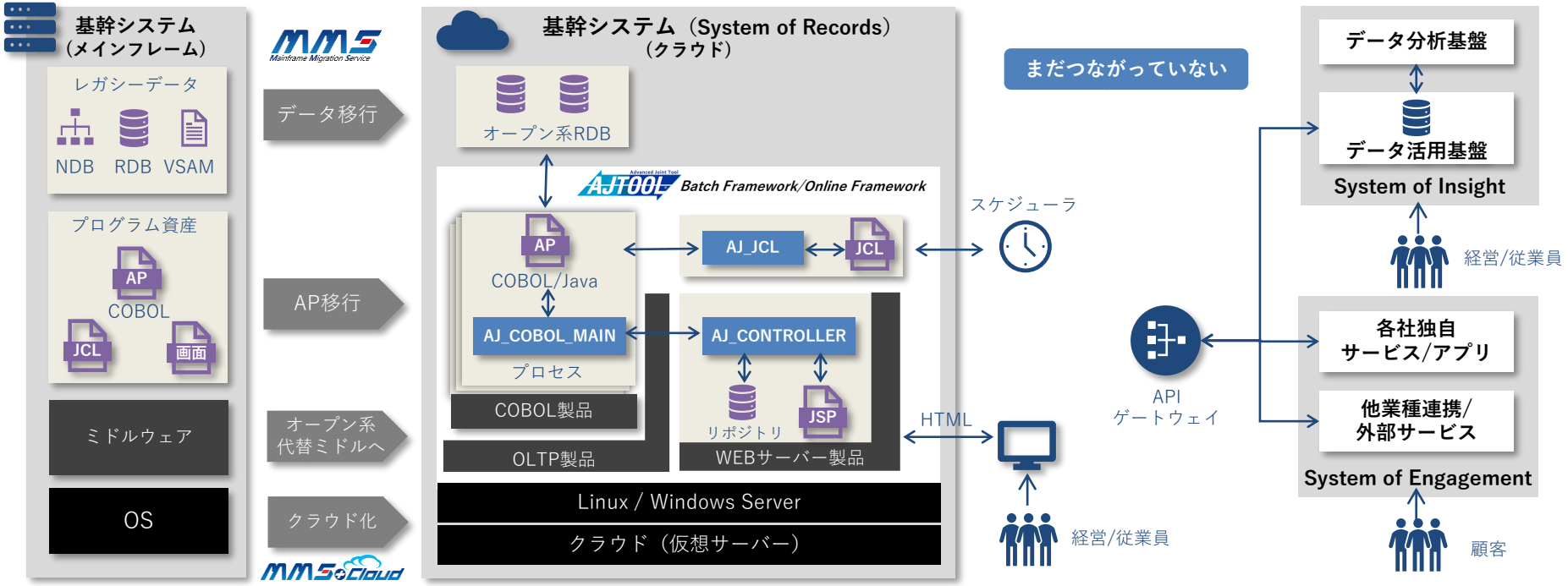
API構築

業務・基幹システム群へのAPI、および、媒介層（COBOLモノリスの入口）を提供すること

データ活用

業務・基幹システム群内のレガシーデータを、リアルタイムまたは業務処理を経由でアクセス可能とすること

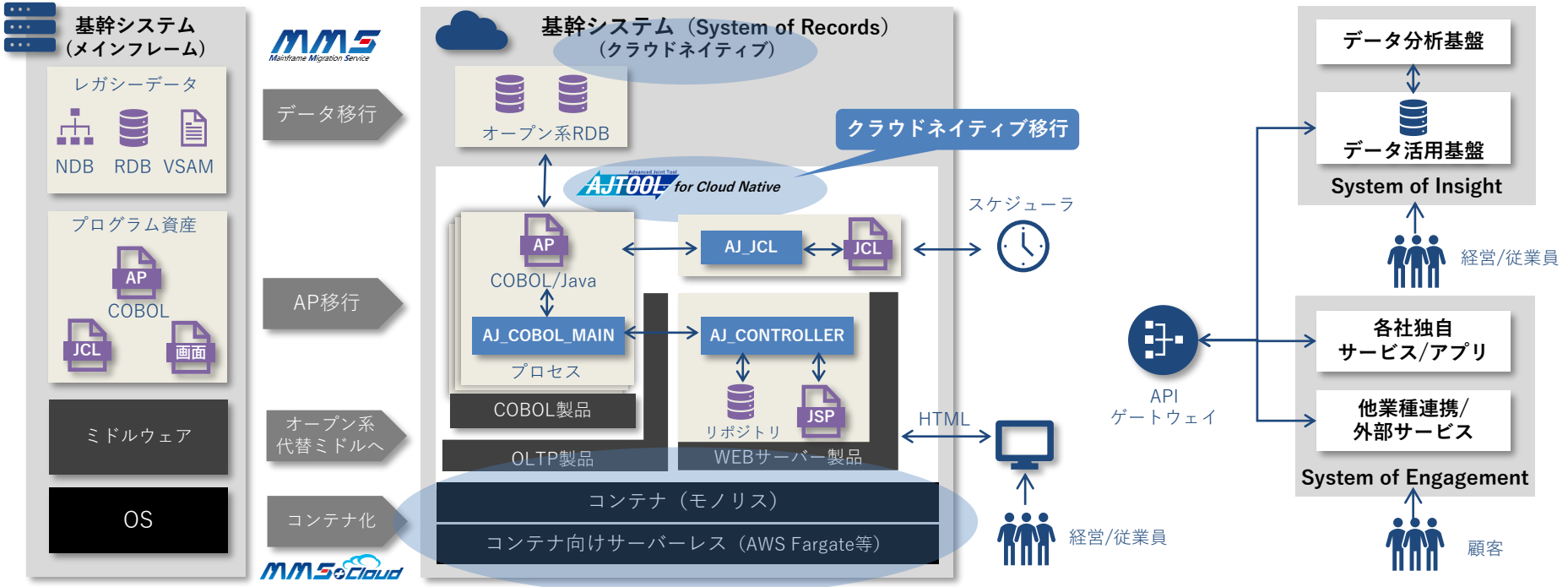
攻めのITにつなげるには　～ モダナイゼーション後



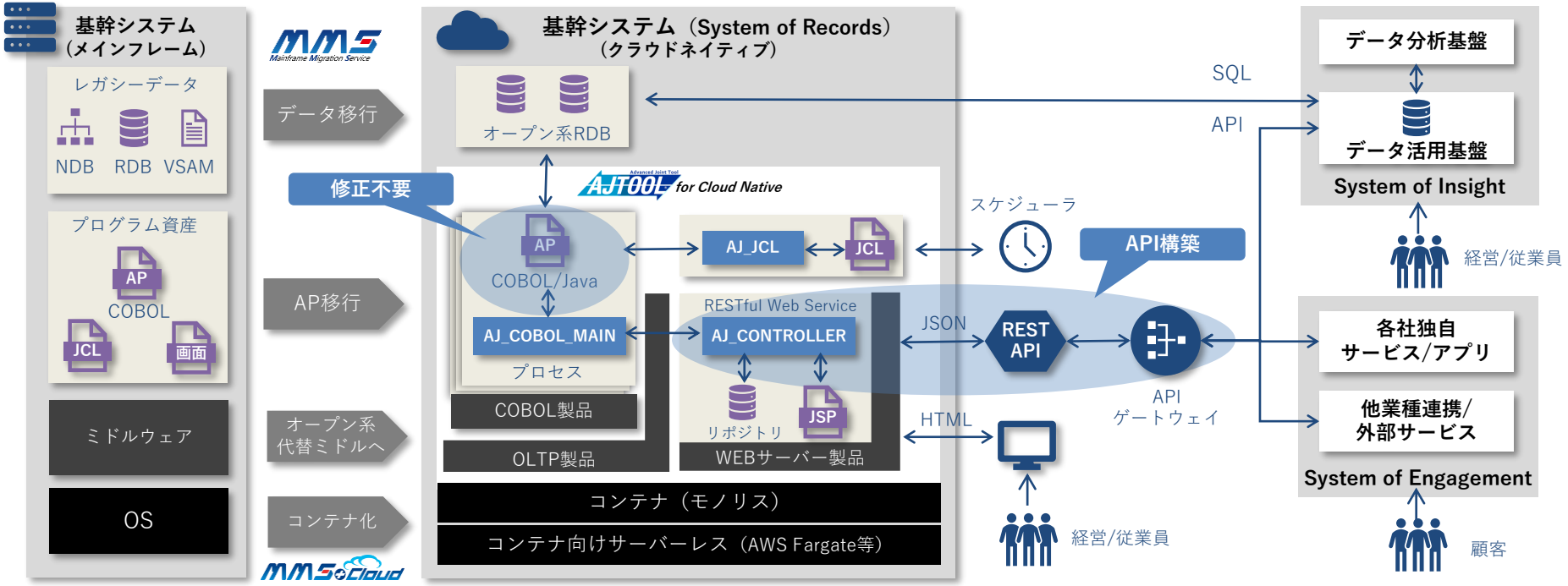
守りのIT（既存ビジネスの運用・維持）

攻めのIT（新ビジネス・価値創出）

攻めのITにつなげるには　～ クラウドネイティブ移行



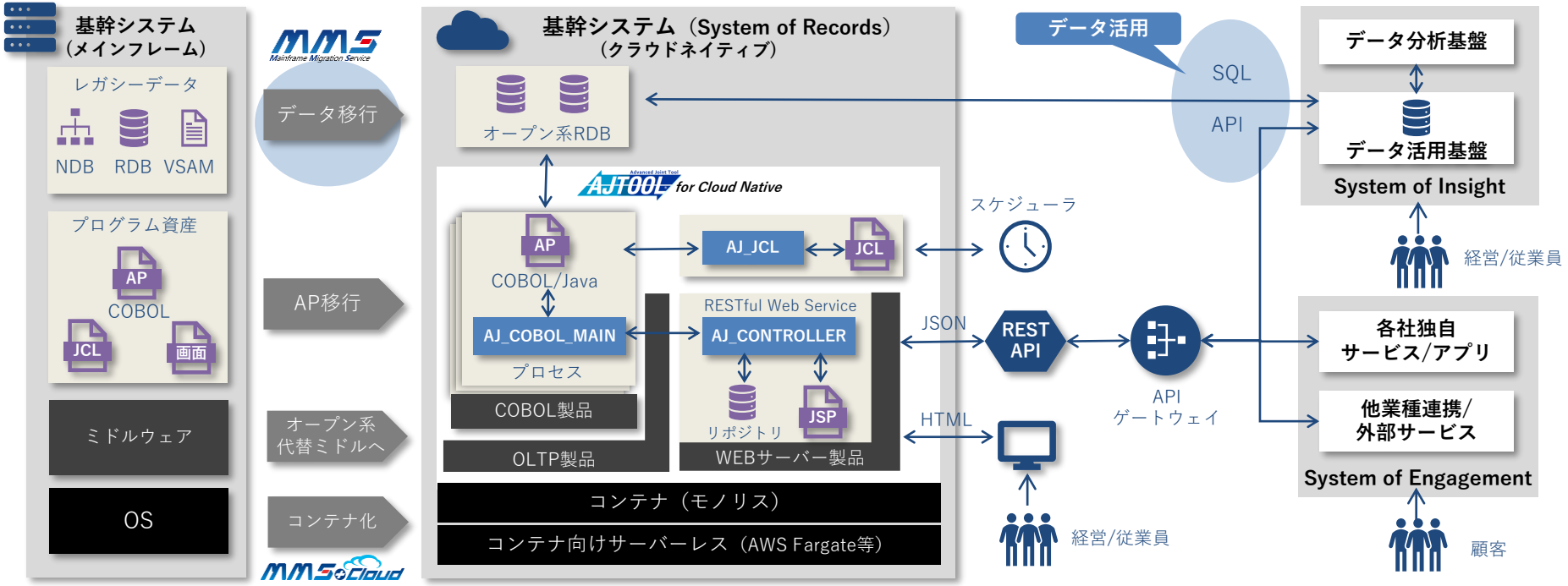
攻めのITにつなげるには　～ API構築



守りのIT（既存ビジネスの運用・維持）

攻めのIT（新ビジネス・価値創出）

攻めのITにつなげるには　～ データ活用



守りのIT（既存ビジネスの運用・維持）

攻めのIT（新ビジネス・価値創出）

まとめ

- COBOLを取り巻く課題は **IT資産の健全性** の問題
- モダナイゼーションはIT資産の健全性を**回復するチャンス**
- COBOLシステムには**長年蓄積したデータ**や**業務ロジック**が含まれる
- **MMS**と**AJTOOL for CloudNative**により、**COBOLシステム**は安定稼働しつつ**攻めのITとつながり**、**新ビジネスや価値創出に貢献**できる

- **今後もIT資産を未来に継承するソリューションを提供してまいります**

お問い合わせ

東京システムハウス株式会社
マイグレーションソリューション部

☎ 03-3493-4601
mms@tsh-world.co.jp
@tsh_mms