デジタル変革にはOSSをこう使ってくれ！

第1部
レガシー刷新と攻めの変革の両立を目指して

OSSを活用したITモダナイゼーション

2023年3月10日（金）

TSH 東京システムハウス株式会社

自己紹介

比毛　寛之
ひもう　　ひろゆき

石巻市ご当地ヒーロー「シージェッター海斗」

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　部長

- レガシーマイグレーション屋さん。COBOL資産のモダナイズや
  基幹系でのOSSの活用など、既存資産を活用した基幹系システムの刷新が得意。
- 活動：OSSコンソーシアム(理事、オープンCOBOLソリューション部会リーダー)
  COBOLコンソーシアム(セミナ分科会)
  DXITフォーラム
- 出身：宮城県石巻市

今日お話すること

- **OSSを活用したITモダナイゼーション**
  - ITモダナイゼーションについて
  - opensource COBOL、opensource COBOL 4J、二刀流の移行手法
- **プラットフォームデジタル化指標**
  - ITモダナイゼーションで気になる事項
  - IT資産の健全性、データ活用性、アジリティ（ユーザ要件への対応）
- **まとめ**
  - レガシー刷新と攻めの変革の両立を目指して

OSSを活用したITモダナイゼーション

東京システムハウスの取り組み

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

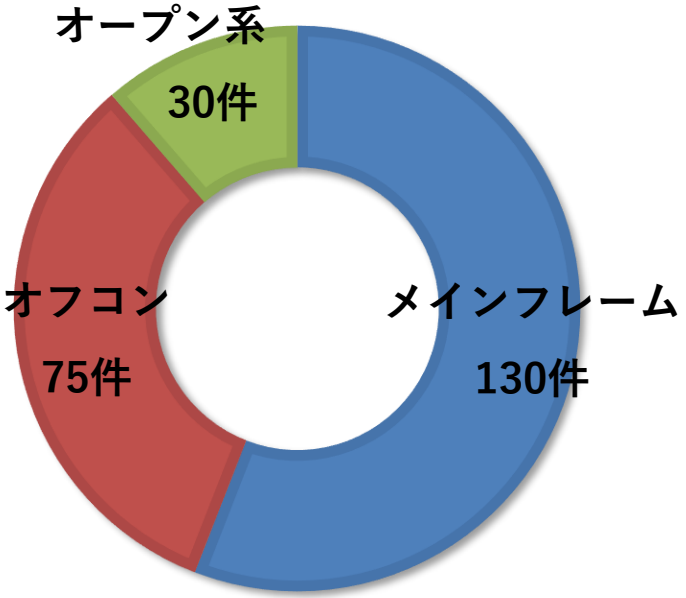
MMS+Cloud

COBOLシステムの
クラウドネイティブ移行
に対応

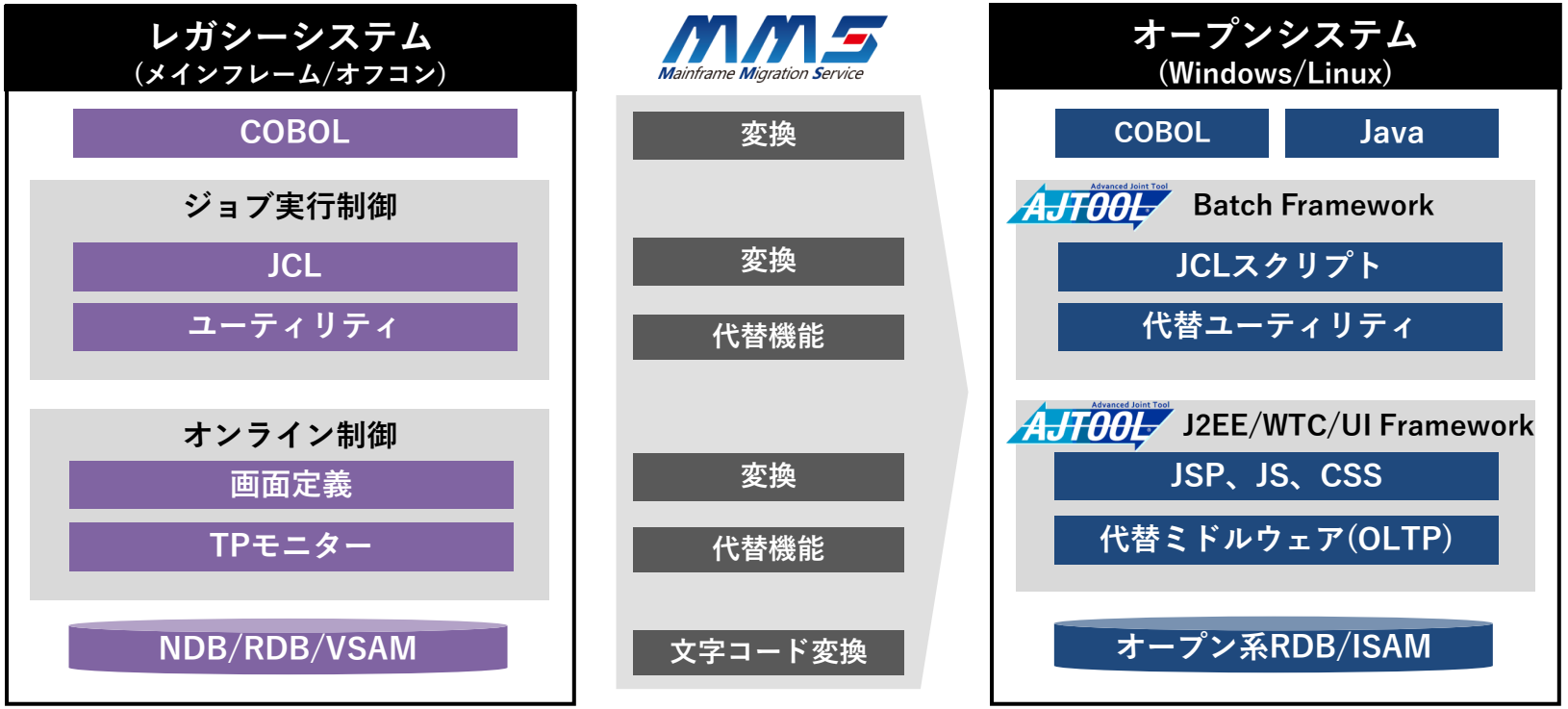
OSS Consortium

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～)



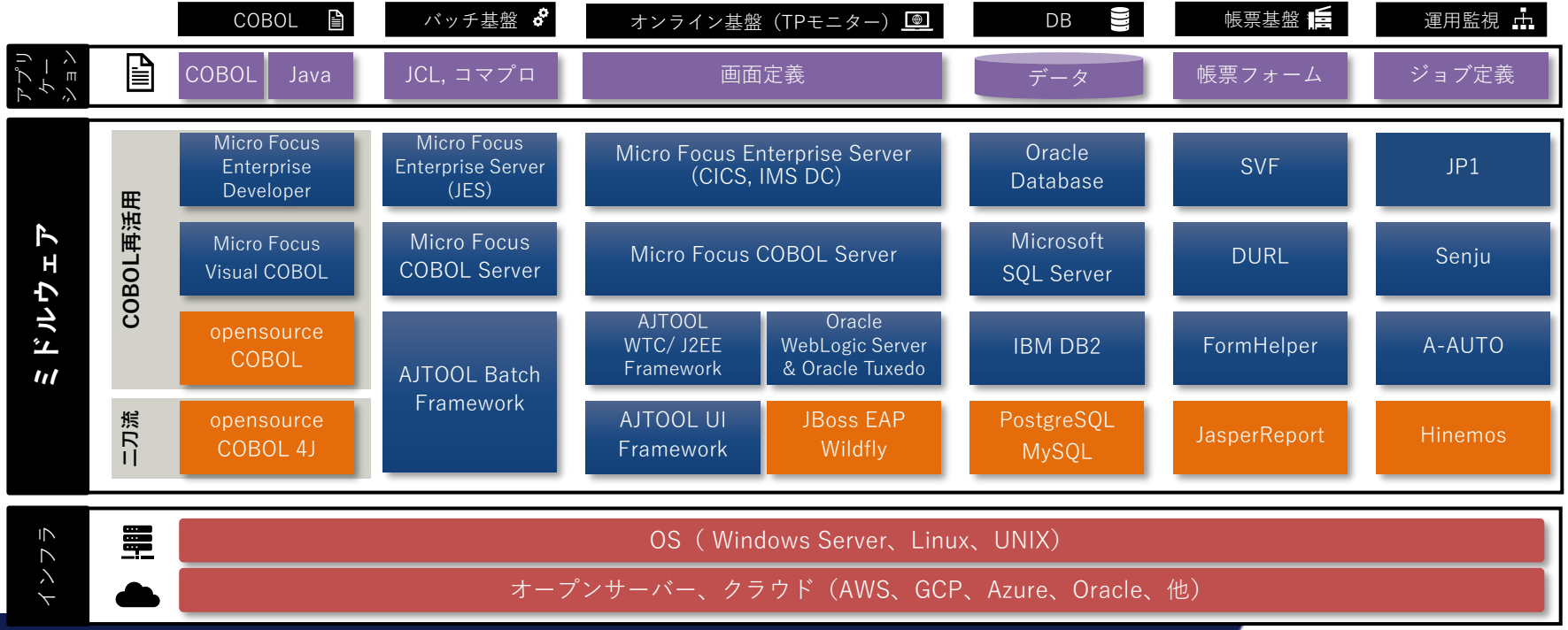
ITモダナイゼーション（資産移行）



ITモダナイゼーション（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



OSSコンソーシアム
オープンCOBOLソリューション部会

目的

基幹システムでのOSS普及を背景として、プロプライエタリな環境が一般的なCOBOLの開発においてもオープンソースのメリットを活かすため、OSS COBOLを技術・ビジネスの両面からサポートできるように整備していき、基幹システムにおけるOSS化の普及・促進に貢献する。

参加企業

- 株式会社アックス
- 株式会社エネルギア・コミュニケーションズ
- サン情報サービス株式会社
- 株式会社ＣＩＪ
- OVOL ICTソリューションズ株式会社
- 株式会社ソフトテックス
- 東京システムハウス株式会社
- 株式会社ビイガコーポレーション
- 株式会社日立ソリューションズ
- 有限会社ランカードコム
- 飯島 裕一

活動内容

国内でも実績のあるOSS COBOLのOpenCOBOL 1.1 pre-releaseをベースに、処理系自身の既知のバグや未実装機能および有用と思われる拡張機能などの情報を共有する。

http://www.osscons.jp/osscobol

https://github.com/opensourcecobol/

一緒に開発しませんか？

OSS Consortium オープン COBOL ソリューション部会

opensource COBOL

OpenCOBOL 1.1
pre-release

ORACAプロジェクトで開発されたオープンソースのCOBOLコンパイラ（原作者 西田圭介さん）
（2009年2月）

1.1
Community Edition

コミュニティ有志によるバグfix等反映版

GnuCOBOL

GnuCOBOL 1.1

FSFに移管、GNUに登録、名称変更（2013年9月）

GnuCOBOL 3.1.2

REPORT SECTION, XML/JSON, COBOL2014（2020年12月）

opensource COBOL

日本コミュニティによるメンテナンス版

opensource
COBOL
v1.2J

（2012年7月）

v1.5.2J

（2019年4月）

日本の商習慣に応じた拡張が必要だったので、
OSSコンソーシアムで開発・メンテナンス
（1.1 pre-release を Jul. 2012 に fork）

OSS Consortium

opensource COBOL

Javaトランスレート版

opensource
COBOL 4J
v1.0 DEV

（2020年7月）

v1.0.8

（2023年2月）

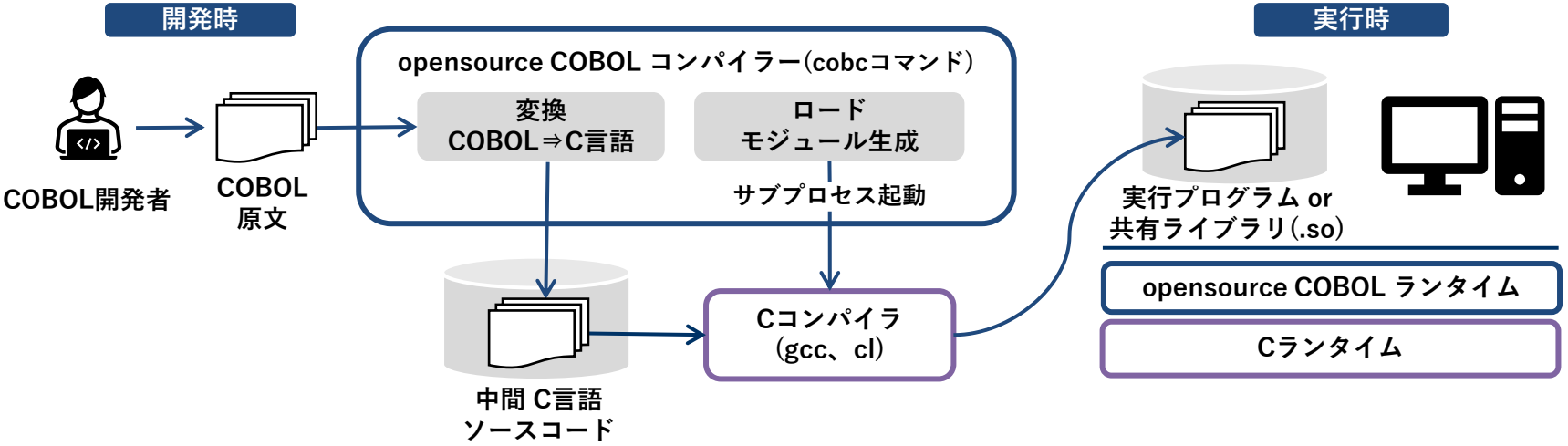
COBOLをJavaへ
トランスレートする
方式に移行

※4J=for Java

opensource COBOL

https://github.com/opensourcecobol

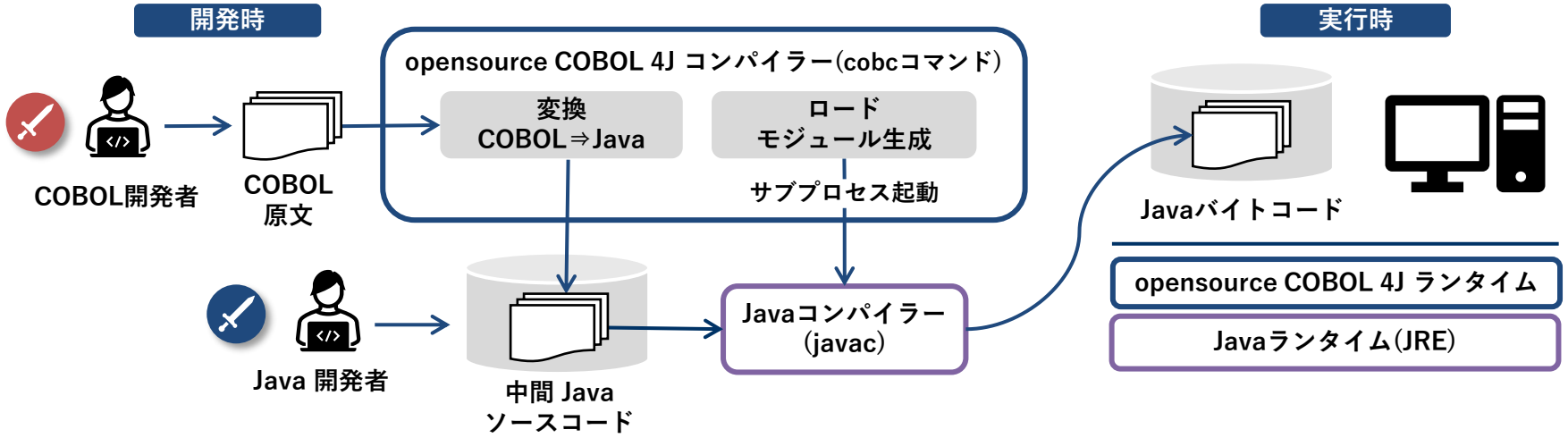
- OSSコンソーシアムで開発・公開しているCOBOLコンパイラ
- COBOLをC言語にトランスレート、Cコンパイラでバイナリを生成
- Linuxは『gcc』、Windowsは『cl』やLinuxエミュレータ(CygWin/MinGW)を利用



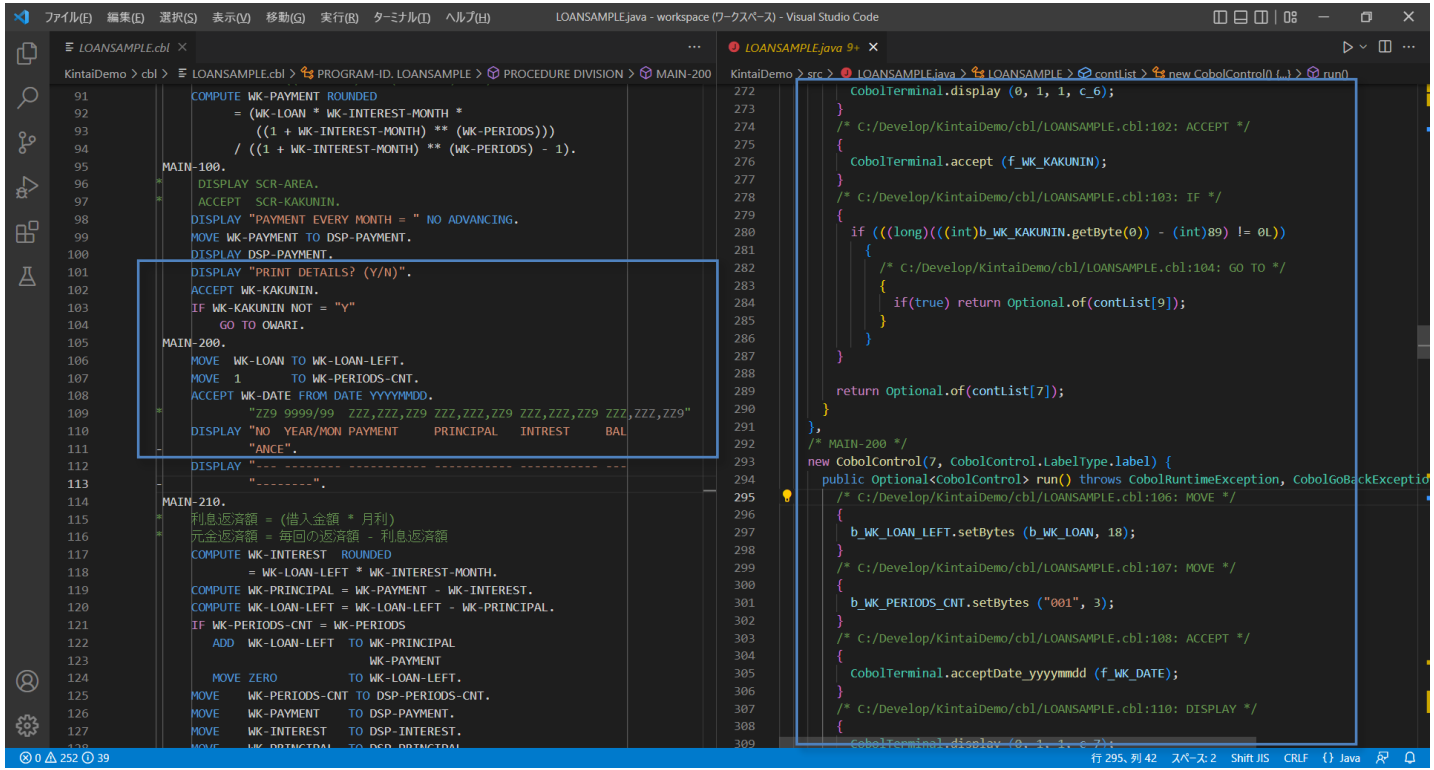
opensource COBOL 4J

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



opensource COBOL 4J



opensource COBOL 4J

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

Java変換 

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。

お試しください

ダウンロードはこちら & 一緒に開発しませんか？
https://github.com/opensourcecobol

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

最新情報は弊社Twitterで発信中（@tsh_mms）
https://twitter.com/tsh_mms

プラットフォームデジタル化指標

ITモダナイゼーションで気になる事項

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		

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。

ITモダナイゼーションで
気になる事項3点

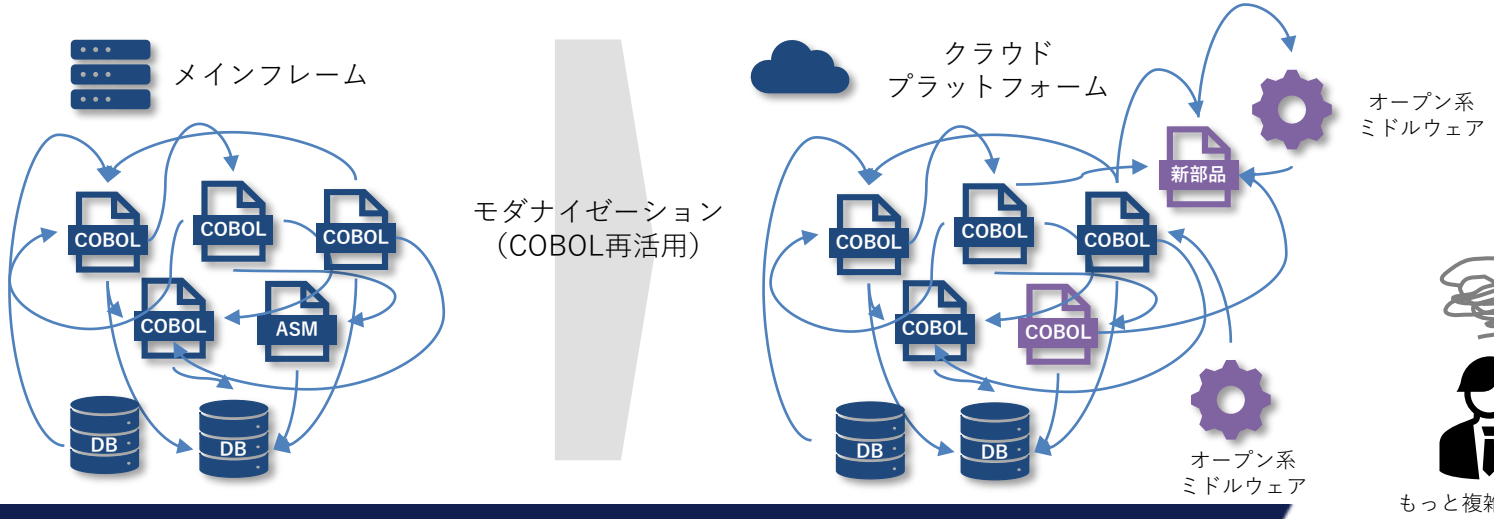
①IT資産の健全性

指標の分類・項目	評価項目の説明	モダナイゼーション観点で気になる点
機能システムごと／評価項目(基礎的な要件)		
IT資産の健全性		
ソフトウェア資産の最適化	品質管理標準に従って、最適な設計・ロジックの追加や修正が継続的に実施された結果、ソフトウェア資産は最適な状態になっているか。 ※目的:ロジックを簡潔にして、スパゲッティ化を防ぐため ※問題となる例: 難解で、修正しにくい、デグレし易い、コーディング不適切なモジュール構成・分割	**元のアプリが複雑化・スパゲティ化している場合は，そのままで移行することにならないだろうか。同様に，使われないコード部分もそっくり移すことにならないか。 保守性・変更可能性を確保するために，ソフトウェア資産を最適化するための工夫はないだろうか。**
不要なソフトウェア資産を増やさない	使われないコーディング、不要なコーディングを組み込まない、かつ、共通的な処理は部品化され、活用するよう徹底した結果、不要なソフトウェア資産がない状態になっているか ※目的:ソフトウェア資産の肥大化を防ぐため ※問題となる例:使われない、または無駄なコーディングがソースコードに含まれているあちこちに似たようなコーディングをしている機能が少ない割に規	

①IT資産の健全性

回答

・はい、現行のスパゲッティ化・複雑化の状態はそのまま移行されます。
そこにホスト機能代替やミドル連携が加わり、もっと複雑になります。
・システムが長年の手入れ不足の場合、 IT資産の健全性を取り戻し、
保守性や変更容易性の改善と再レガシー化しない方式を検討します。



①IT資産の健全性

ところで、IT資産にCOBOLを残すのか？

- 最近のCOBOL言語の話題
  - 基本情報処理技術者試験でCOBOLの出題終了
  - 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ースのようだ。
COBOLが手を離せば天空が落ちてくる。
（画像 Wikipediaより）

COBOL言語の問題ではなく「IT資産の健全性」の問題

①IT資産の健全性

- 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ってしまった
  - **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）

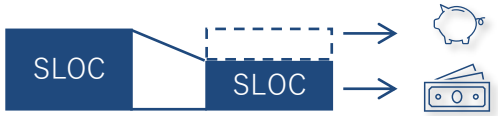
COBOL再活用 でも 脱COBOL でも良いが、まず健全性の確保を

①IT資産の健全性

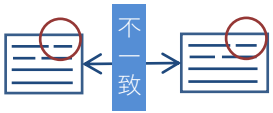
モダナイゼーションにおけるIT資産の健全性のポイントは4点

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？

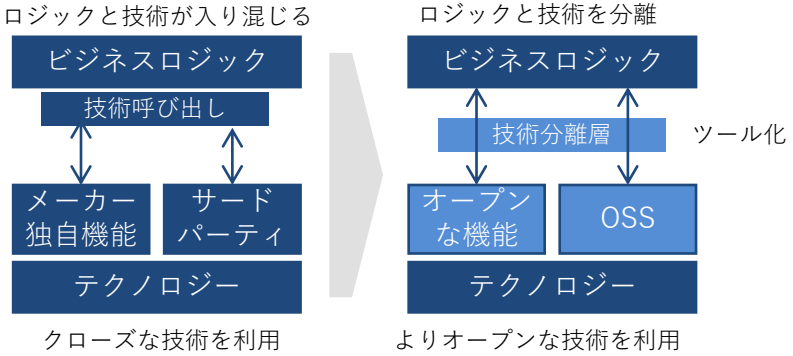
- **保守性・変更容易性の観点で対策**
  - デッドコード、なぞ変数名、バイパス見直し
  - スパゲティは仕様理解者または解析ツール活用
  - 現行システム上でしっかり動作確認

①IT資産の健全性

モダナイゼーションにおけるIT資産の健全性のポイントは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



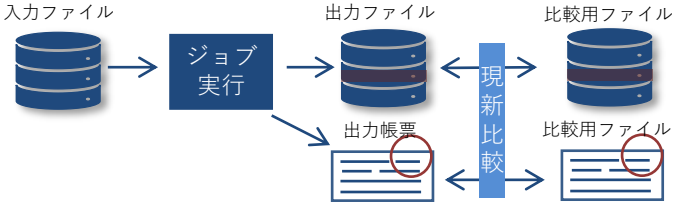
- **再レガシーの回避**
  - メーカー独自機能を互換製品で回避すると将来の再レガシーの原因となり得る

①IT資産の健全性

モダナイゼーションにおけるIT資産の健全性のポイントは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

①IT資産の健全性

モダナイゼーションにおけるIT資産の健全性のポイントは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若手へのノウハウ継承**
  - 若手はシナリオ/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

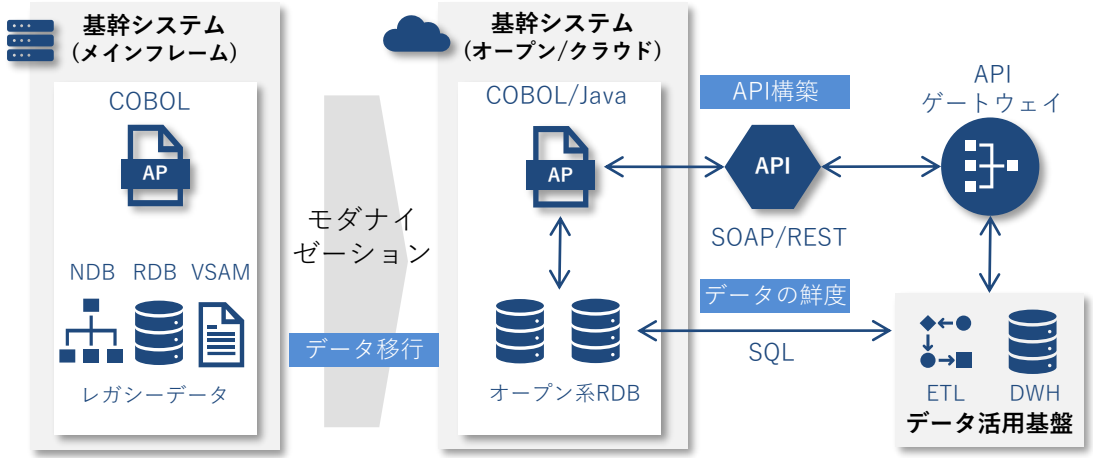
②データ活用性

指標の分類・項目	評価項目の説明	モダナイゼーション観点で気になる点
機能システムごと／評価項目(DX対応に求められる要件)		
データ活用性 ※SoR/SoEともに		
データの鮮度	活用すべきデータをリアルタイムに取得できるか。 ※リアルタイムに取得するデータは、リアルタイムにも、日次や月次などにも活用可能	**DXのキモのひとつはデータ活用。そのためには活用できるようにする「仕組み」も必要で，従来は特定アプリだけが使っていたデータを活用できるかどうか。そのようなオープンさを持たせられるかどうかも，モダナイゼーションへの期待のひとつではないか。しかし，そこまで期待をするのは無茶ぶりだろうか。**
データの量の変化への対応	定義済データ項目について、必要十分なだけのデータ量を取得しているか。また、データ量を柔軟に拡張可能か。	
データ分析へのインプット方法	取得データを、AI（機械学習／深層学習など）や、データ分析のシステムに容易にインプットできる仕組みになっているか	

②データ活用性

回答

・無茶ぶりではありません。オープン系技術でモダナイズすることで、
基幹系データのリアルタイム提供やAPIを通じた柔軟な提供ができます。
・データの鮮度は、論理構造や排他条件の精査、締め前提の業務仕様など、
新鮮であれば良いのかについて十分な検討が必要です。



・データ移行
・レガシーデータをオープン系RDBに移行
・活用し易い列の検討（階層キー、マルチレイアウト、集団項目、OCCURS）

・API構築
・モノリスへの媒介層として、OLTP構築と同じ方法で元COBOLへのAPIを構築

・データの鮮度
・リアルタイムのデータ提供が可能だが、論理構造、排他条件、締め処理など新鮮さ vs 業務仕様を十分に検討

③アジリティ(ユーザ要件への対応)

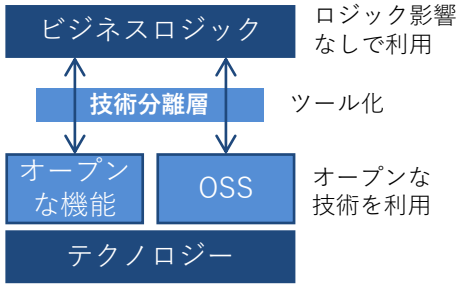
指標の分類・項目	評価項目の説明	モダナイゼーション観点で気になる点
機能システムごと／評価項目(DX対応に求められる要件)		
アジリティ(ユーザ要件への対応)		
要件変更し易い実装	小さい業務機能などの単位で、独立して開発できるような作りになっているか(適切なデータ分離、機能分割、構造化、カプセル化、重複や矛盾のないデータ、必要十分な設計情報の記述、など)	**外部エコシステムと連携するには，旧システムとは異なるシステム間連携やミドルウェア（比較的新しい技術）に対応したい場合もあるだろう。その場合，採用したモダナイゼーション手法やツールが事前に想定されているものでないと対応困難と推察する。希望する連携を実現するならば結局は新規開発をせざるを得ないだろうか。**
迅速な対応のための組織・体制	事業責任者、業務担当、システム担当が三位一体となって、製品/サービスのシステム対応を迅速に実施できる組織・体制を取っているか ※システム対応の度に、社内調整、承認などで時間をかけないため	
エコシステムの活用、連携の容易さ	外部のエコシステムの活用・連携が容易な方式を取っているか 例:サーバレスなビジネスロジック実行基盤、NoSQLデータベース	

③アジリティ(ユーザ要件への対応)

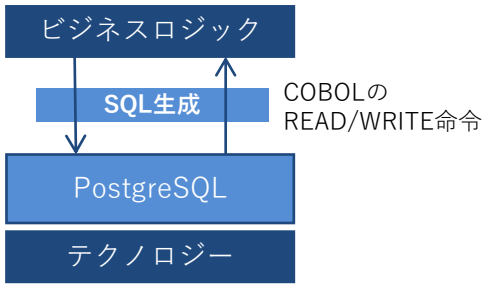
回答

・新ミドルや新技術ではテクノロジーとビジネスロジックの分離が重要です。
COBOL再活用の場合は、COBOL命令のまま利用できるようにします。
・個別対応をしないようにツール化し、開発者が向き合うビジネスロジックには
影響しません。将来、テクノロジーを変更する際の修正を最小限にします。

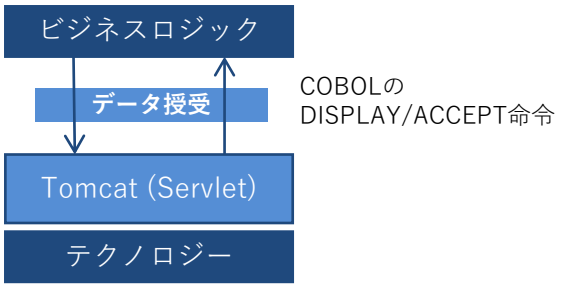
＜基本方針＞



＜対応例1＞



＜対応例2＞



テクノロジーの変更は、技術分離層を変更すればOK、ビジネスロジックの変更は最小限
外部のエコシステムと連携する場合も同様の方式を取る

まとめ

- **OSSを活用したITモダナイゼーション**
- **プラットフォームデジタル化指標（PFD指標）**
- **レガシー刷新と攻めの変革の両立に向けて**
  - PFD指標でシステムを精密検査して最適な対策を検討
  - ITモダナイはデジタル変革の第一歩、IT資産の健全性の回復が重要
  - OSS活用と自社スキルの回復で、攻めの変革の足掛かりに

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

Mainframe Migration Service