COBOLとモダナイゼーション：
世界の潮流とRocket Softwareの進化

Senior Vice President of Hybrid Cloud Sales,
Rocket Software
Stuart McGill

Rocket® software

Modernization.™
Without Disruption.™

お客様のモダナイゼーションがどの過程にあっても、そのニーズに応えます。

Rocket® software

メインフレームのモダナイゼーション分野で最も包括的なサービスポートフォリオ

モダナイゼーションの実施
保持、リファクタリング

ハイブリッド環境での実行
保持、リファクタリング

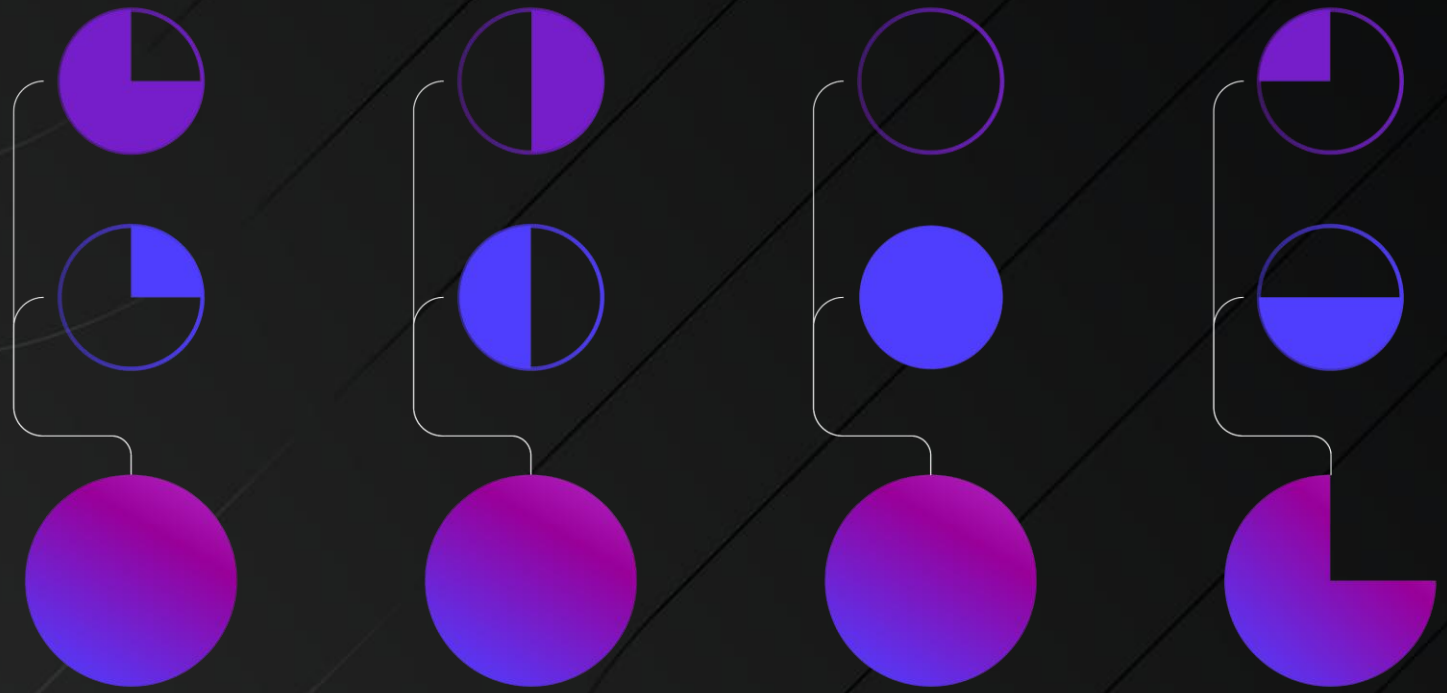
プラットフォームの変更
リプラットフォーム、廃止

アプリケーションのリファクタリング
保持、リファクタリング

ロケットソフトウェア

Enterprise Suite

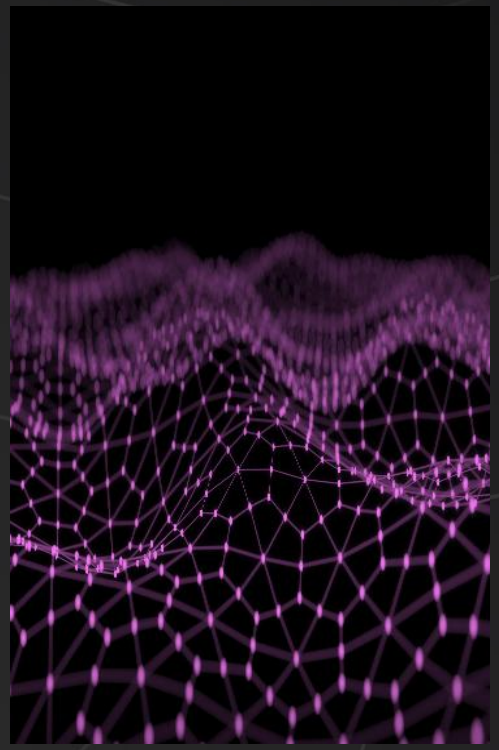
ロケットソフトウェア
+
Visual COBOL +
Enterprise Suite



Rocket software

ビジネスアプリケーションに最適なモダナイゼーション戦略の策定

ITの煩雑性



ビジネス面での検討事項

コスト最適化　標準化　戦略的調整

運営上のリスク　簡素化　市場投入時間

ビジネス評価

煩雑性　スキル　保守性

技術的負債　セキュリティ　スケーラビリティと可用性

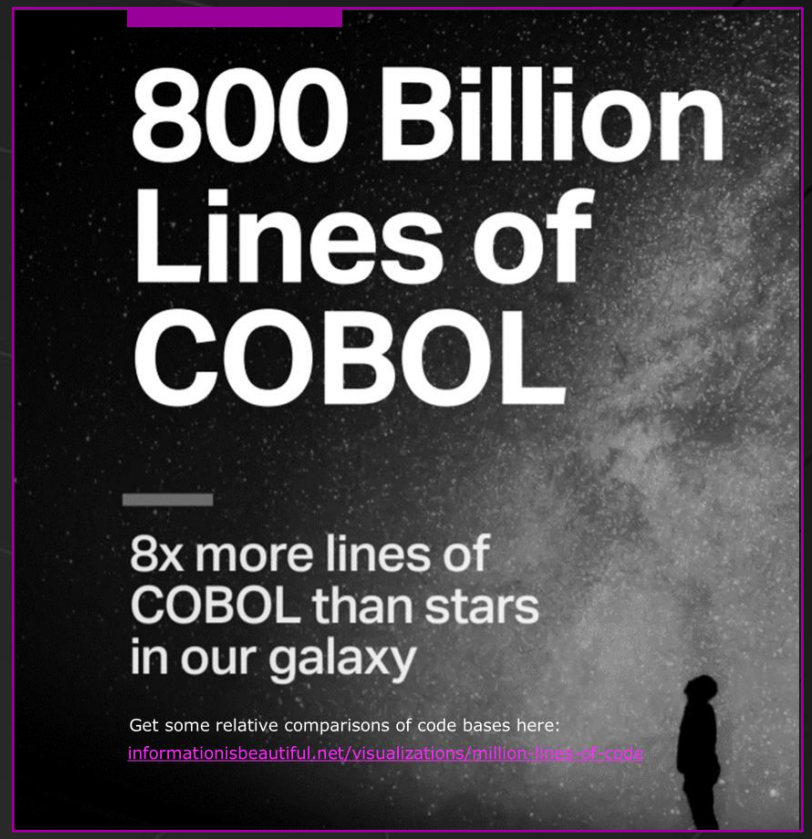
モダナイゼーション戦略

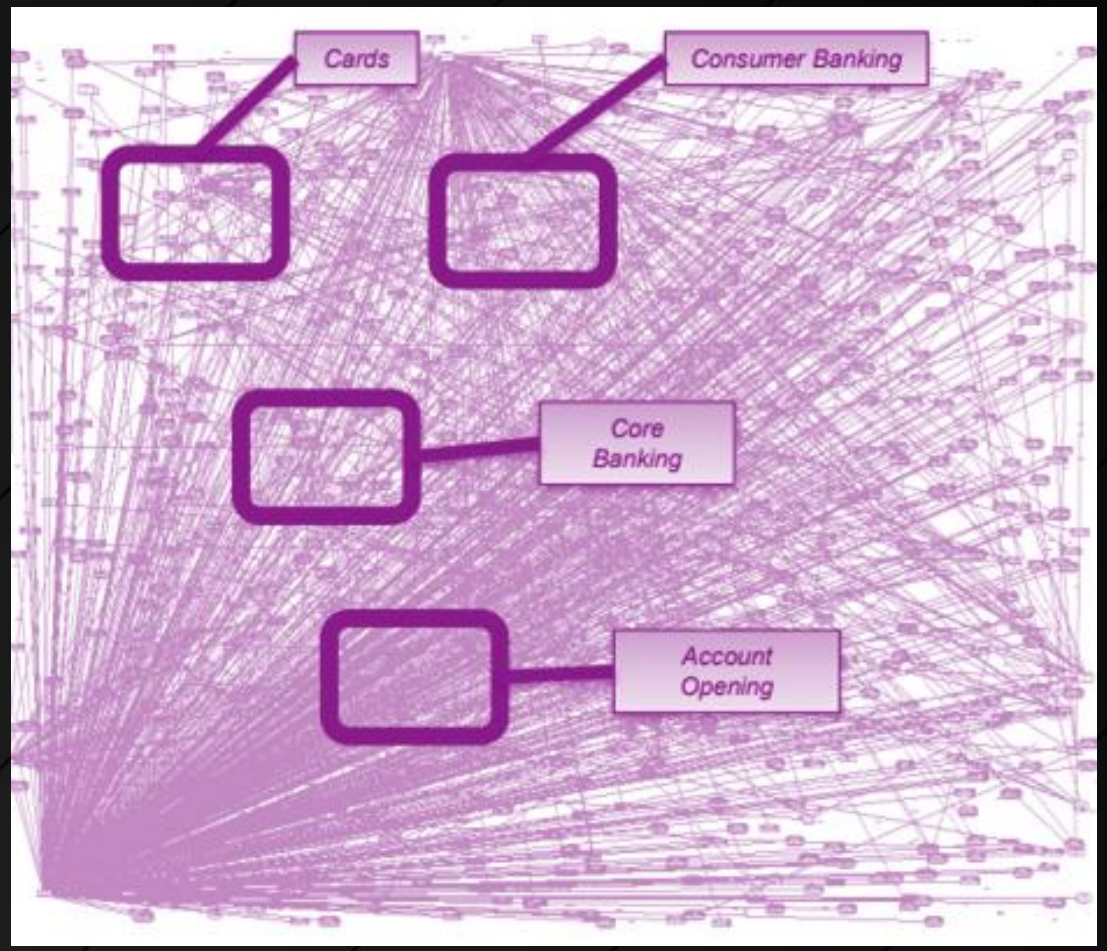
- 廃止
- 入替
- 保持
- リファクタリング
- リプラットフォーム

テクノロジー面の検討事項

Rocket software

COBOLシステムの規模を決して過小評価しないこと





Rocket software

現状確認：どこに生成AIを活用できるか？

課題

- 訓練データの制限と不完全性
- ハルシネーション（不正確な出力）
- 生成AIを圧倒する規模と煩雑性
- コンテキストウィンドウの制限

アナリスト向けガイダンス

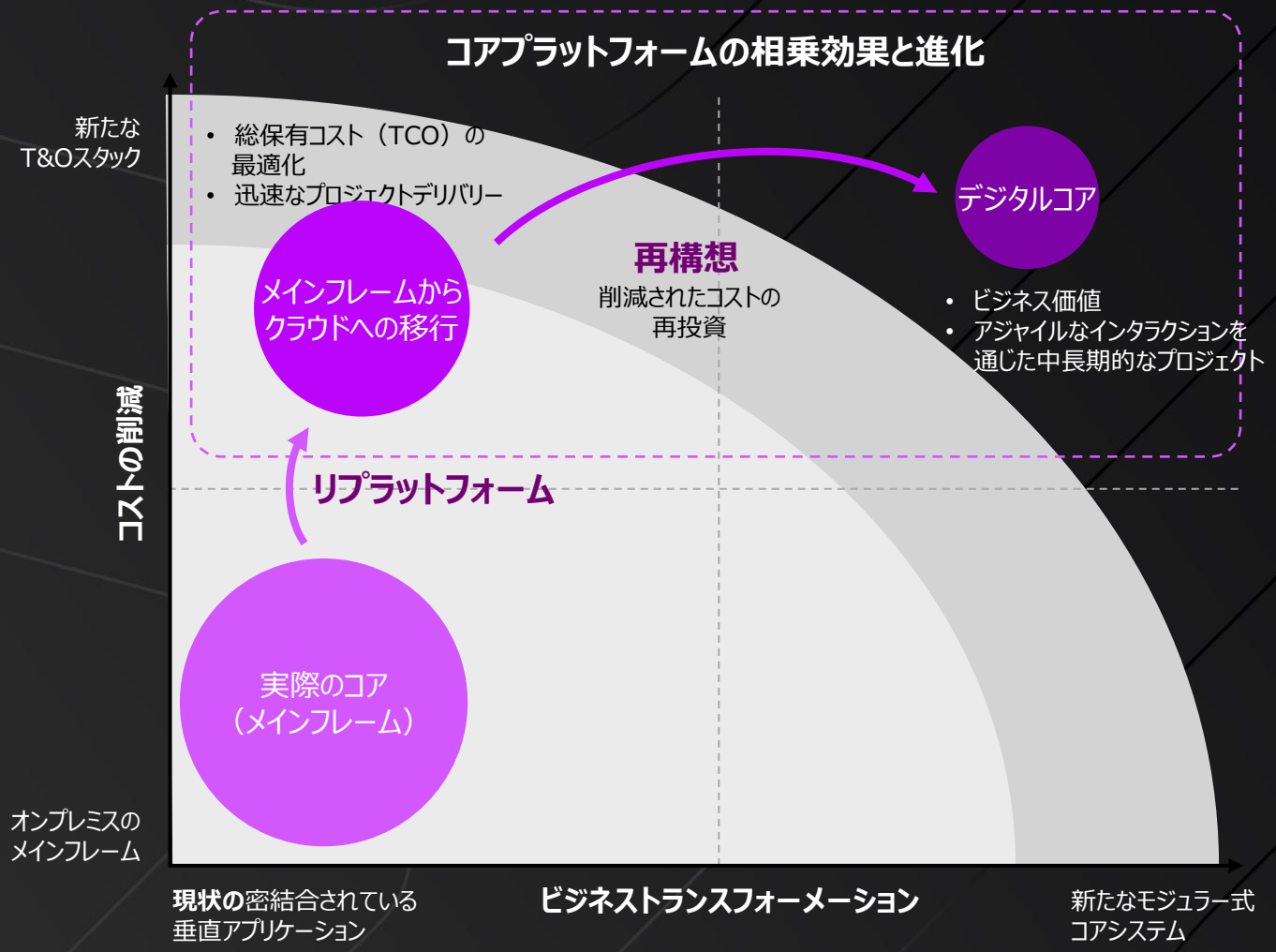
生成AIでコードのリライトを自動化しないこと

- 出力は正確に見える
- しかし、念入りにチェックする必要がある！

生成AIの適切な役割

- **開発ワークフローを強化**してモダナイゼーションを加速させる
- **文書化、コードの解釈、統合開発環境（IDE）内アシスタント**に利用

最高の投資利益率を実現できる総合的なロードマップ



プラットフォーム変化のイメージ

コアアプリケーションを2つのプラットフォームに分散させつつ連携するように進化させることで、統合した形でビジネスを支援

リプラットフォーム（約80%）	再構築（約20%）
大多数のアプリケーションとサービス	重要なアプリケーションとサービスのプロセスの変更
コアをクラウドに移行することでTCOを削減	ビジネス価値の実現

メインフレームからクラウドへの移行では、アプリケーションを新たなロケットソフトウェアのプラットフォームへ移管することに集中：

- クラウドのアーキテクチャに移行することでメインフレーム以外のテクノロジーも利用可能になり、クラウドかオンプレミスかを問わず、TCOを40%削減可能
- コアアプリケーションへの投資ライフサイクルを延長
- フロントへのデータ移行
- DevOpsによるアプリケーションの構築と実行を支援

デジタルコアでは、以下を目的として価値の高いアプリケーションとサービスに集中：

- コアビジネスモデルの再構築
- 新たな付加価値サービスの開発支援
- 主なデータエンティティ（顧客データ等）の資産化
- 新たなテクノロジーアーキテクチャを通じた再設計による最も価値の高いサービスの強化

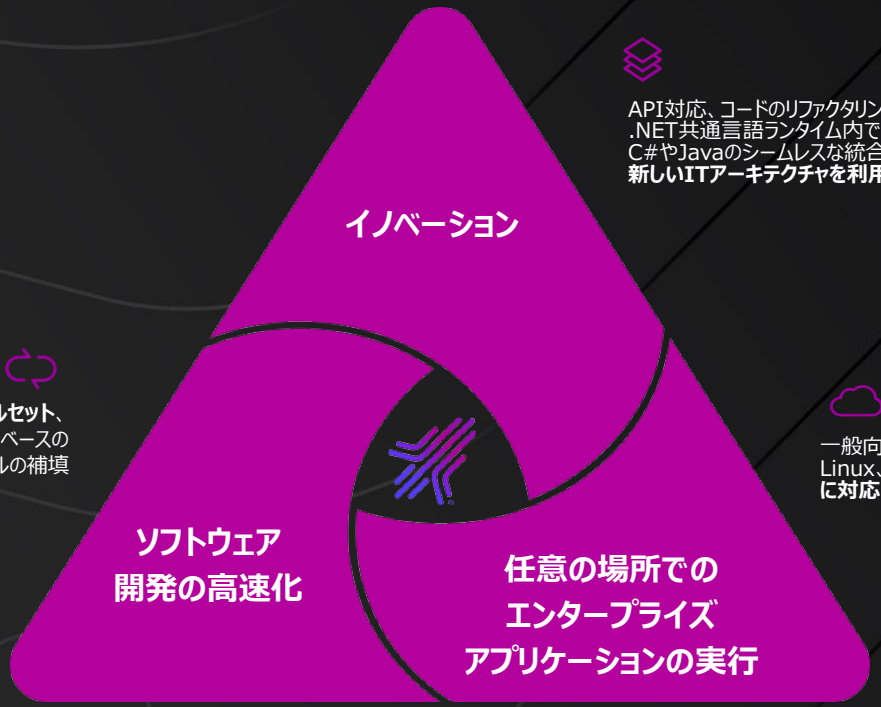
Rocket software

モダナイゼーション成熟度モデル

アプリケーションのモダナイゼーションを次の段階へ進める際の判断基準

			クラウド		
インフラ	**メインフレーム**	**分散**	**クラウドへ移行可能**	**クラウド向けに最適化**	**クラウドネイティブ**
アプリケーション	• モノリス • 自社専用 • ACIDトランザクション	• N層アーキテクチャ • 移動可能 • 仮想	• 疎結合 • リレーショナル • 「マクロ」コンテナ	• コンポーネント • APIサービス • コンテナ	• マイクロサービス • NoSQL • BASEトランザクション
ITプロセス	ウォーターフォール型	反復的	アジャイル	DevOps	DevSecOps
管理	初期	適切な管理下	定義されている	計測されている	最適化が進行中
組織文化	部門別	トップダウン	協調	高い信頼	パフォーマンス

よりスマートなモダナイゼーション



API対応、コードのリファクタリング、Java仮想マシンや.NET共通言語ランタイム内でのCOBOLの実行、C#やJavaのシームレスな統合など、**新しいITアーキテクチャを利用**

最先端の開発ツールセット、アジャイルな手法の採用、ナレッジベースのツールや生成AIによる不足スキルの補填

一般向けのプラットフォームからクラウド、Linux、コンテナまで、**あらゆるIT環境に対応**

Rocket® COBOL/Enterprise Server:
メインフレームアプリケーション向けの柔軟でスケーラブルなデプロイメント環境を提供し、クラウドサービスを含むさまざまなプラットフォームをサポートします。

Rocket® Visual COBOL/Enterprise Developer:
IBM®メインフレームアプリケーション向けの最新統合開発環境（IDE）で、アジャイル開発プロセスを容易にします。

Rocket® Enterprise Analyzer:
包括的なアプリケーションインテリジェンスと分析ツールを提供し、意思決定を支援します。

Rocket® software

アプリケーション全域のナレッジと生成AIの融合
説明：LLMの回答精度を向上させるためのロケットソフトウェアのアプローチ

Visual COBOL
Enterprise Developer

チャット | 説明 | ドキュメント

精度と生産性

RAGモデルへの入力

キュレーションされた
コード

アプリケーション
全域のナレッジ

製品ドキュメント

Rocket COBOL

自社システムが基盤

生成された
プロンプト

モデルの選択
OpenAI、Gemini、Graniteなど

*バージョン11.0でVisual Studio Code（EclipseおよびVisual Studio）に対応

生成AIにCOBOL コードを説明してもらう

「これは何をする
コードですか？」

```
    perform find-airport
    if airport-found = 1
        move f-rec to lnk-rec
    end-if
.
1 references | Explain | Generate Doc Comment
distance-between-airports section.
*> finds airports and distance between them
    initialize lnk-distance-result
    move lnk-airport1 to airport
    perform find-airport
    if airport-found = 1
        move f-rec to a1-rec
        move lnk-airport2 to airport
        perform find-airport
        if airport-found = 1
            move f-rec to a2-rec
            perform calculate-airport-distar
            move distance to distance-km
            move distance-m to distance-mil
        end-if
    end-if
.

calculate-airport-distance section.
    move a1-latitude to file-angle
    perform convert-angle
```

生成AIの説明

```
* This COBOL code initializes variables for storing airport names
* and distances. It searches for two airports using the
* find-airport procedure. If both airports are found, it
* calculates the distance between them. The final distances (in
* kilometers and miles) are stored in appropriate variables.
1 references | Explain | Generate Doc Comment
distance-between-airports section.
*> finds airports and distance between them
    initialize lnk-distance-result
    move lnk-airport1 to airport
    perform find-airport
    if airport-found = 1
        move f-rec to a1-rec
        move lnk-airport2 to airport
        perform find-airport
        if airport-found = 1
            move f-rec to a2-rec
            perform calculate-airport-distance
            move distance to distance-km
            move distance-m to distance-miles
        end-if
    end-if
.
```

生成AIに複雑なコード文の解釈を手伝ってもらう

```
calculate-airport-distance section.

    move a1-longitude to file-angle
    perform convert-angle
    move out-angle to long1

    move a2-latitude to file-angle
    perform convert-angle
    move out-angle to lat2

    move out-angle to long2

    compute distance = function acos(function sin(lat1) *
                       function sin(lat2) + function cos(lat1) *
                       function cos(lat2) *
                       function cos (long2 - long1))
                       * radius-of-earth

    compute distance-m = distance / km-per-mile
.
```

このコードスニペットを説明

生成AIの説明

ROCKET COBOL EXPERT

EXPLANATION OF AIRCODE.CBL[145,11-147,33]

This COBOL code is performing a calculation to determine the distance between two geographic locations (latitude 1 - lat1, longitude 1 - long1 and latitude 2 - lat2, longitude 2 - long2) on the Earth's surface using the spherical law of cosines.

Here is a breakdown of the code:

- "*> spherical law of cosines..." This comment indicates that this section of code implements the spherical law of cosines to compute distances on the sphere.
- "compute distance = ..." The COMPUTE statement in COBOL assigns the value of an arithmetic expression to a data item (in this case, 'distance').
- "function acos(..." The ACOS function computes the arccosine of its argument in

ROCKET COBOL EXPERT

EXPLANATION OF AIRCODE.CBL[145,11-147,33]

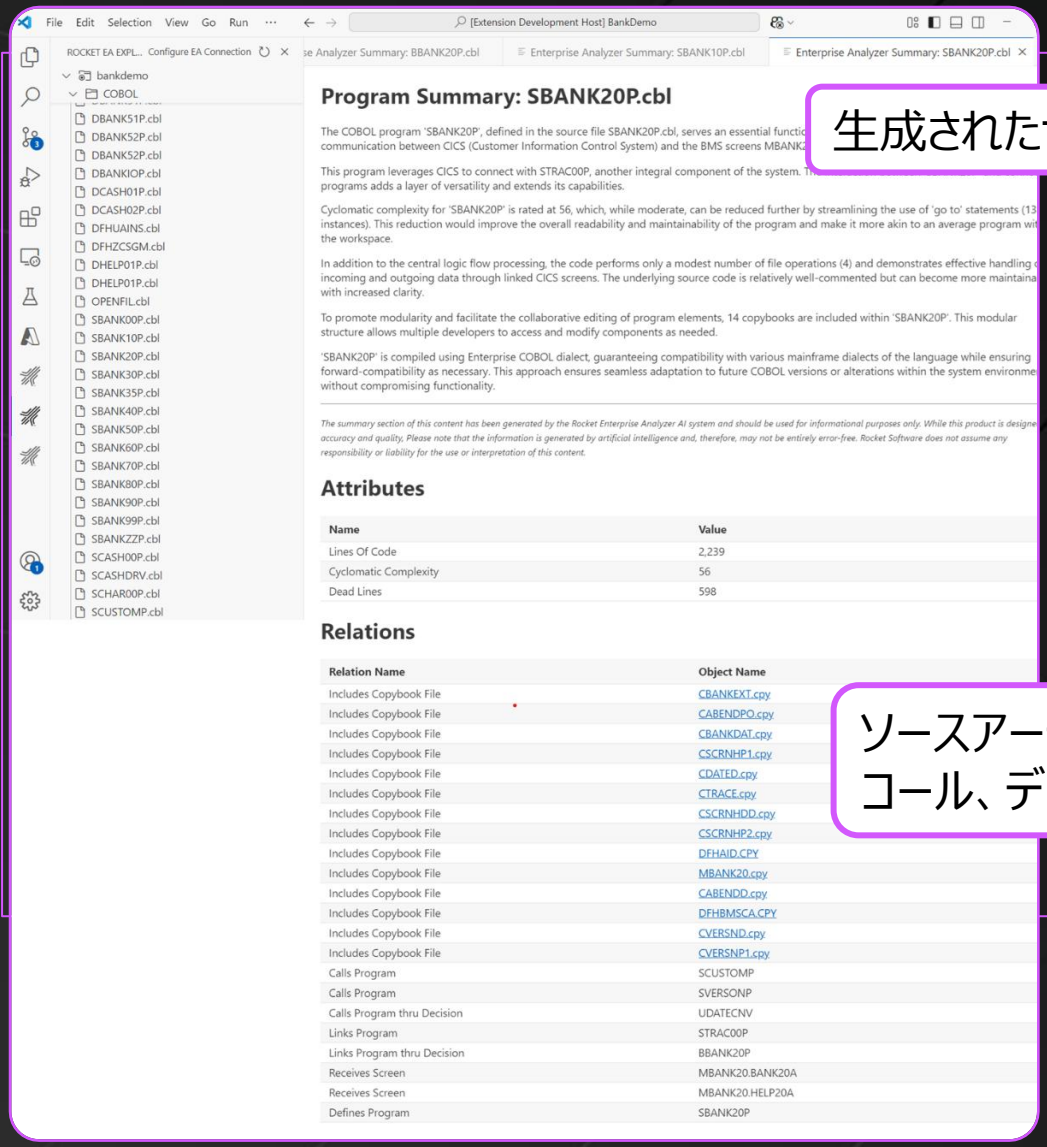
This COBOL code is performing a calculation to determine the distance between two geographic locations (latitude 1 - lat1, longitude 1 - long1 and latitude 2 - lat2, longitu long2) on the Earth's surface using the spherical law of cosines.

expression representing the angular difference between longitude 1 and longitude 2.

- "* radius-of-earth" This factor in the formula is the Earth's radius and the distance on the sphere will be the calculated angle times this value.

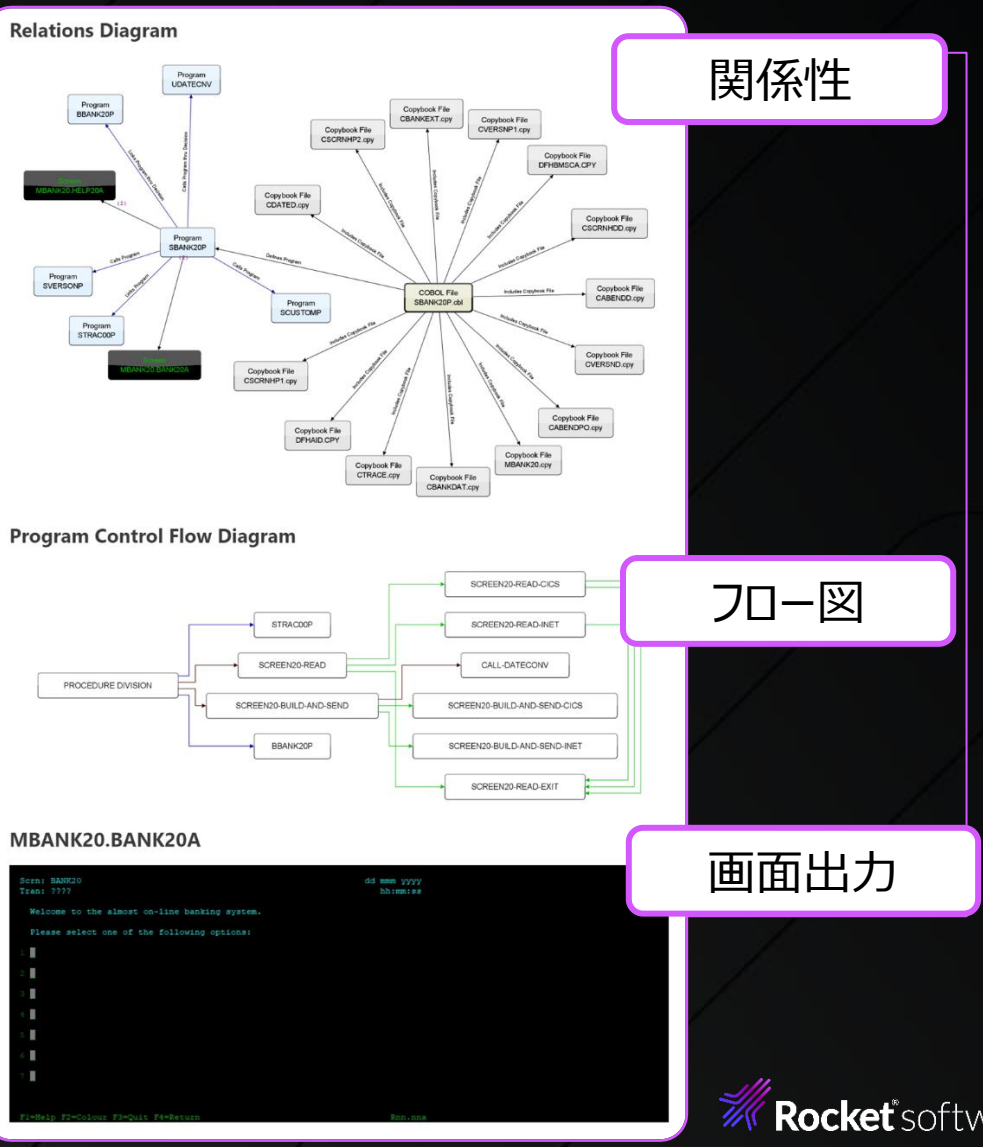
In summary, within a COBOL program context, this code snippet uses the spherical law of cosines to calculate a geographical distance (distance) based on latitude 1 (lat1), longitude 1 (long1), latitude 2 (lat2), and longitude 2 (long2). By performing various trigonometric operations, it calculates the angle between the points on the sphere, then multiplies that by the Earth's radius to determine the distance between those coordinates in a spherical sense.

生成AIに各プログラムのサマリーを作成してもらい、全体像を把握する



生成されたサマリー文

ソースアーティファクト、
コール、データアクセス



関係性

フロー図

画面出力

Rocket software

AIロードマップ：今後12～18か月の展望
より迅速な発見、よりスマートなインサイト、より安全な大規模モダナイゼーション

10.0

コーディング支援

- 平易な英語での質問
- デッドコードの特定

11.0

コードの説明

- 説明のインライン挿入
- プログラムのサマリー
- ビジネス名

Copilotの統合

- IDEのCOBOLにおけるCopilotの利用

ビジネスルールの生成

- テクノロジーとビジネスの架け橋
- 意味を備えたラベルの作成

スマート検索

- 重要点を即座に発見
- 自然言語クエリ
- 「銀行口座を作るコードを探して」

オペレーションとトラブルシューティング

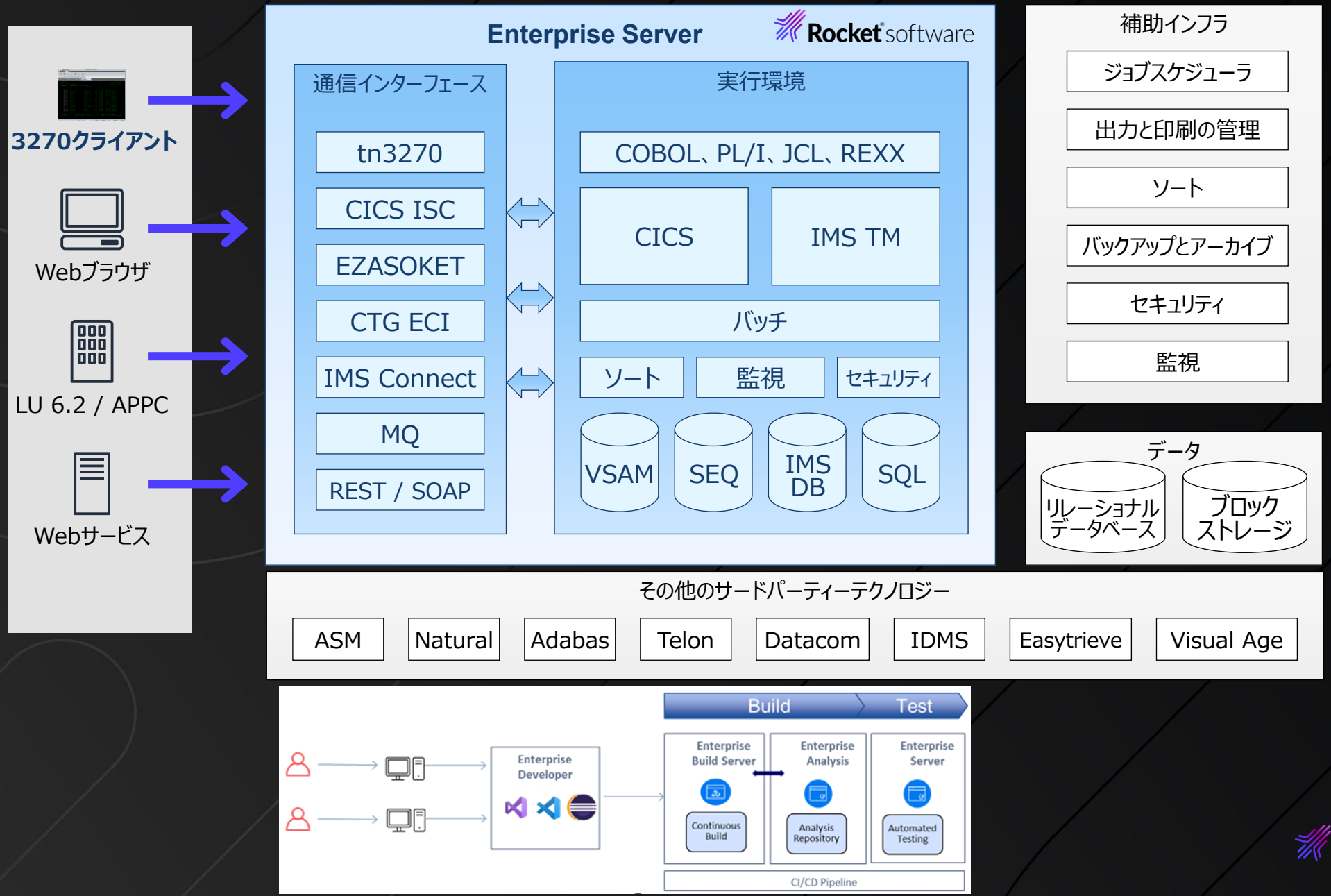
- トラブルシューティング（なぜこのジョブは失敗したのですか？）
- 非構造化データの検索

エンタープライズAIの統合

- アナライザーのインテリジェンスをエンタープライズAI・モダナイゼーションプラットフォームに統合
- より迅速で安全な大規模モダナイゼーション

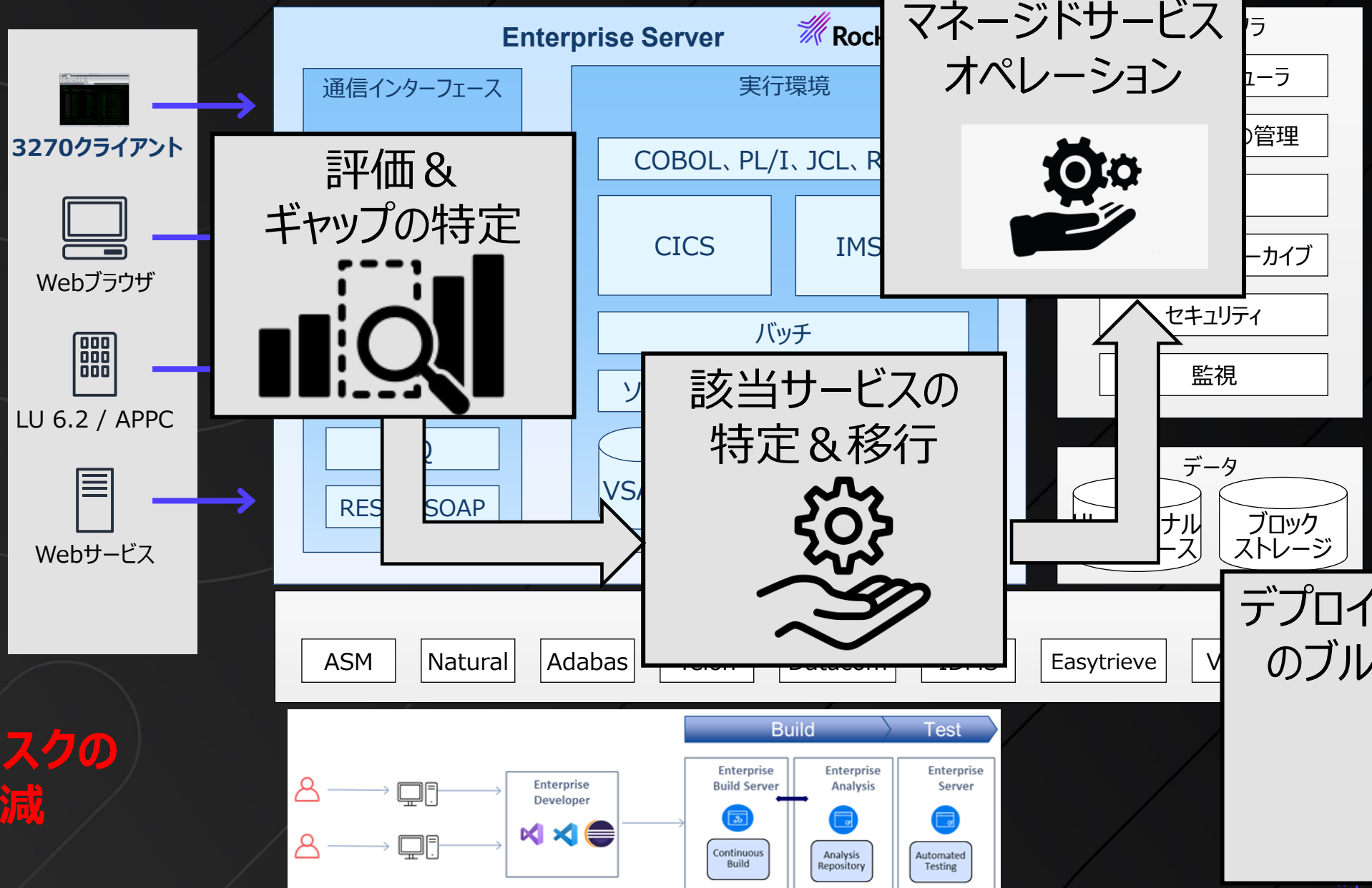
ISO42001準拠 – 責任あるAIの組み込み

ロケットソフトウェアの分散／クラウドプラットフォームアーキテクチャ参考図



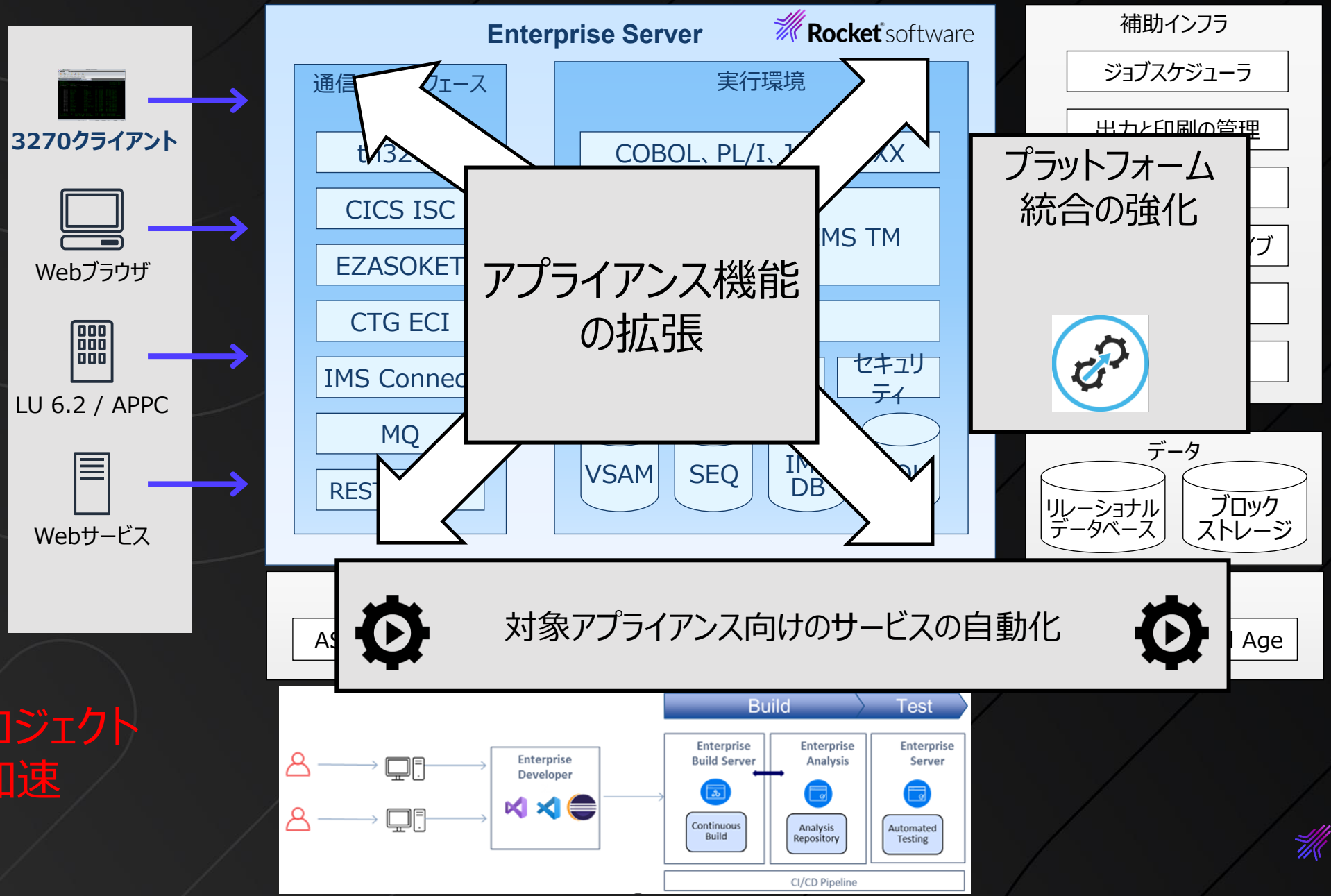
Rocket software

ロケットソフトウェアのアーキテクチャ参考図



移行リスクの軽減

ロケットソフトウェアの拡張アーキテクチャ参考図



Rocket software

ロケットソフトウェアによるアプリケーションモダナイゼーション

中断のないモダナイゼーション - 信頼性の高い迅速なアプリケーションリプラットフォーム	2010年以降、コードのリライトを必要としないリプラットフォームプロジェクトを数百件成功させた実績
レジリエンス、要求に応じたスケーリング、セキュア	ハイブリッド環境における弾力的なコンピューティングによるエンタープライズグレードのパフォーマンス。 セキュア・バイ・デザインかつISO 27001認証を取得済み
デジタル変革の加速	COBOLパイプライン、高度な開発ツールセット、継続的インテグレーション（CI）・継続的デリバリー（CD）
組織のナレッジの保護	AIツールでコアシステムのナレッジを保持・拡張
自由なアーキテクチャ	メインフレーム、オンプレミス、クラウド、コンテナのすべての環境に対応
将来に備えたモダナイゼーション	シームレスなAPI対応、Java、.NET、統合を含む
実績あるグローバルサポートモデル	経験豊富なパートナーとコンサルタントが成功を保証

Rocket® software

これぞテクノロジーの逆説

Thank you

Rocket® software