

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



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。

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

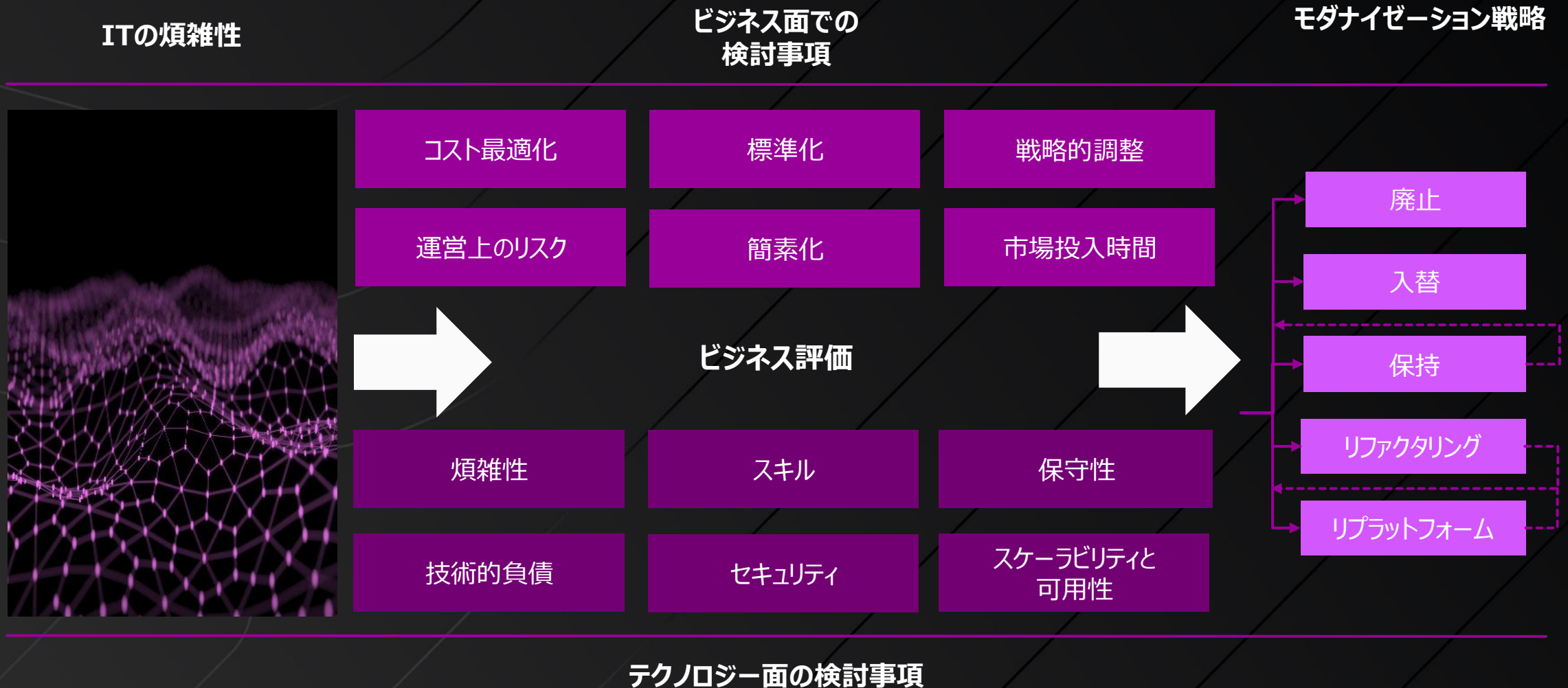
保持、リファクタリング

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

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

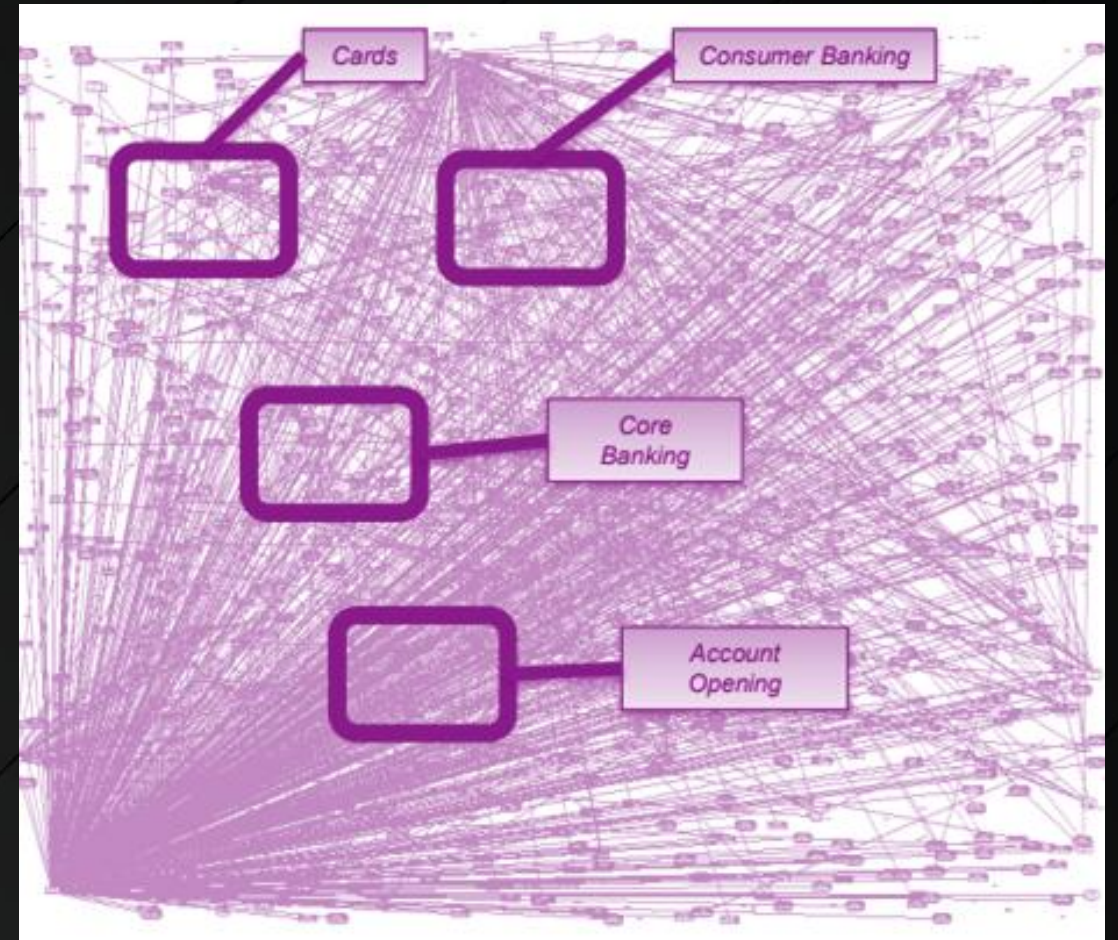


COBOLシステムの規模を決して過小評価しないこと

800 Billion Lines of COBOL

8x more lines of
COBOL than stars
in our galaxy

Get some relative comparisons of code bases here:
informationisbeautiful.net/visualizations/million-lines-of-code



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

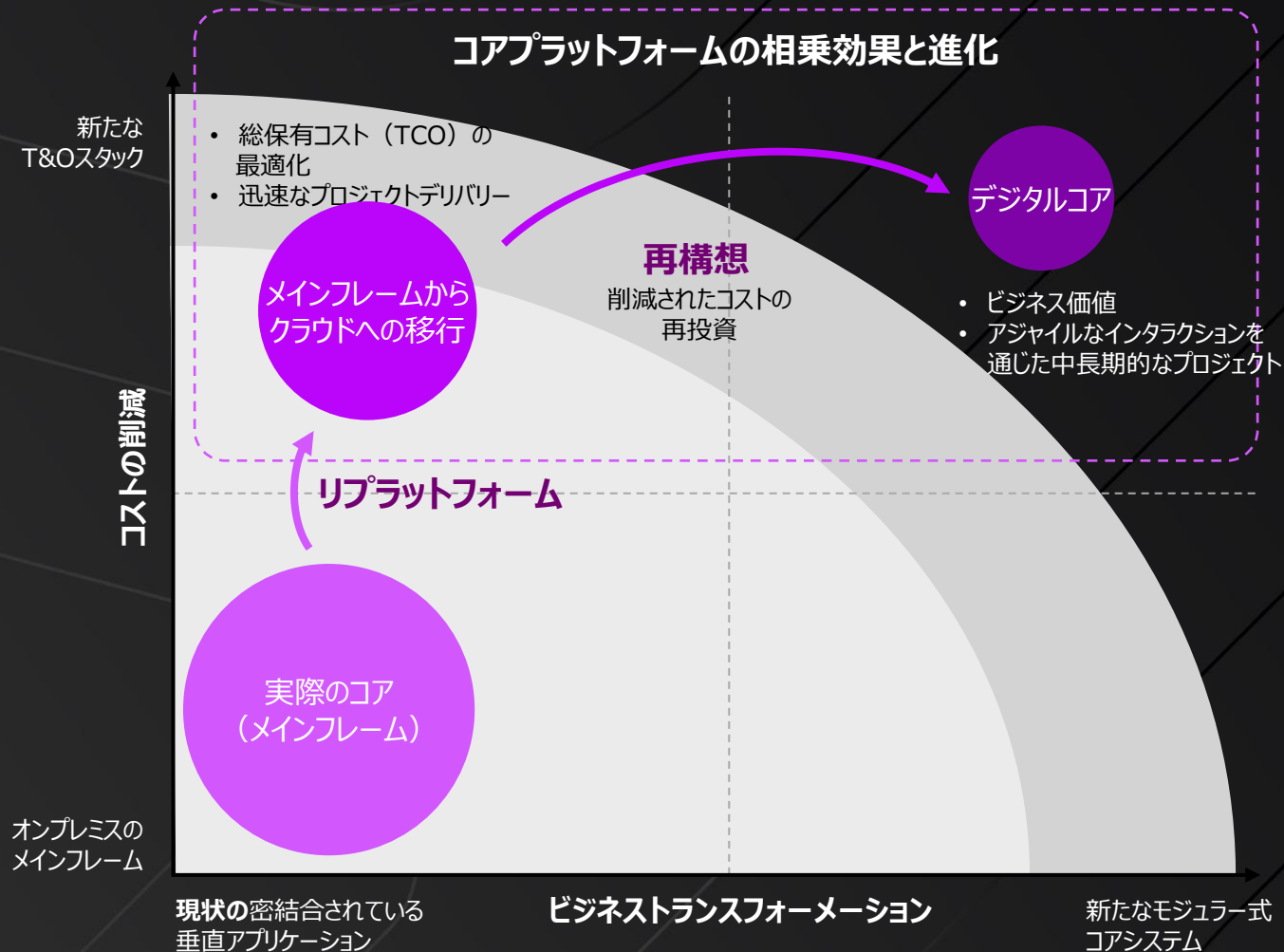
- 出力は正確に見える
- しかし、念入りにチェックする必要がある！



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%）

大多数のアプリケーションとサービス

コアをクラウドに移行することでTCOを削減

再構築 （約20%）

重要なアプリケーションとサービスのプロセスの変更

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：

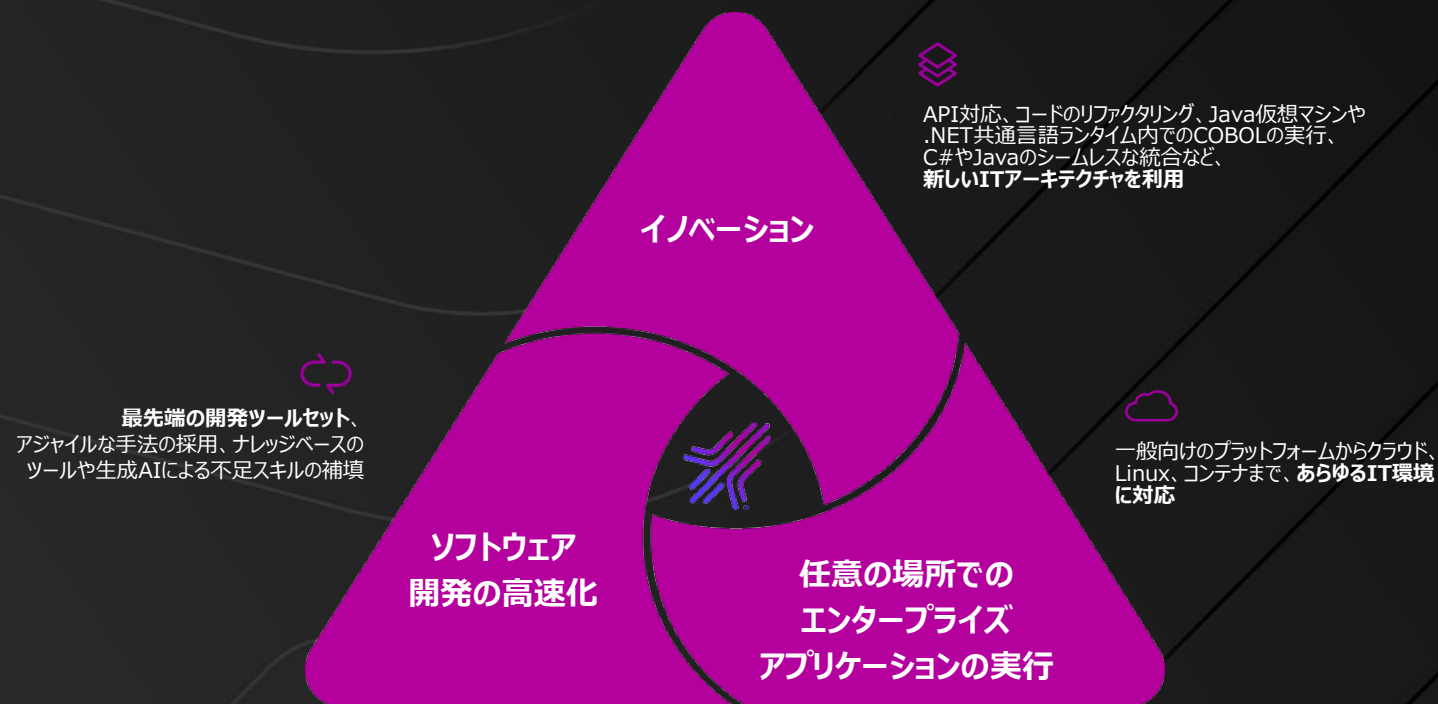
- コアビジネスモデルの再構築
- 新たな付加価値サービスの開発支援
- 主なデータエンティティ（顧客データ等）の資産化
- 新たなテクノロジーアーキテクチャを通じた再設計による最も価値の高いサービスの強化

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

インフラ	メインフレーム	分散	クラウド		
			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



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:

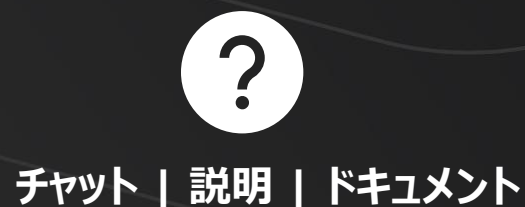
IBM®メインフレームアプリケーション向けの最新統合開発環境（IDE）で、アジャイル開発プロセスを容易にします。

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。

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



RAGモデルへの入力



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

生成AIにCOBOL コードを説明してもらおう

「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

.
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

.
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

生成AIの説明

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
```

このコードスニペットを説明

```
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
```

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, longitude 2 - 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

ROCKET EA EXPL... Configure EA Connection x Analyzer Summary: SBANK20P.cbl Enterprise Analyzer Summary: SBANK10P.cbl Enterprise Analyzer Summary: SBANK20P.cbl x

bankdemo

- COBOL
- DBANKS1P.cbl
- DBANKS2P.cbl
- DBANKS2P.cbl
- DBANKIOP.cbl
- DCASH01P.cbl
- DCASH02P.cbl
- DFHUIANS.cbl
- DFHZCSGM.cbl
- DHELP01P.cbl
- DHELP01P.cbl
- OPENFIL.cbl
- SBANK00P.cbl
- SBANK10P.cbl
- SBANK20P.cbl
- SBANK30P.cbl
- SBANK35P.cbl
- SBANK40P.cbl
- SBANK50P.cbl
- SBANK60P.cbl
- SBANK70P.cbl
- SBANK80P.cbl
- SBANK90P.cbl
- SBANK99P.cbl
- SBANKZZP.cbl
- SCASH00P.cbl
- SCASHDRV.cbl
- SCHAR00P.cbl
- SCUSTOMP.cbl

Program Summary: SBANK20P.cbl

The COBOL program 'SBANK20P', defined in the source file SBANK20P.cbl, serves an essential function in facilitating communication between CICS (Customer Information Control System) and the BMS screens MBANK20 and MBANK20A.

This program leverages CICS to connect with STRAC00P, another integral component of the system. The program adds a layer of versatility and extends its capabilities.

Cyclomatic complexity for 'SBANK20P' is rated at 56, which, while moderate, can be reduced further by streamlining the use of 'go to' statements (13 instances). This reduction would improve the overall readability and maintainability of the program and make it more akin to an average program within the workspace.

In addition to the central logic flow processing, the code performs only a modest number of file operations (4) and demonstrates effective handling of incoming and outgoing data through linked CICS screens. The underlying source code is relatively well-commented but can become more maintainable with increased clarity.

To promote modularity and facilitate the collaborative editing of program elements, 14 copybooks are included within 'SBANK20P'. This modular structure allows multiple developers to access and modify components as needed.

'SBANK20P' is compiled using Enterprise COBOL dialect, guaranteeing compatibility with various mainframe dialects of the language while ensuring forward-compatibility as necessary. This approach ensures seamless adaptation to future COBOL versions or alterations within the system environment without compromising functionality.

The summary section of this content has been generated by the Rocket Enterprise Analyzer AI system and should be used for informational purposes only. While this product is designed for accuracy and quality, please note that the information is generated by artificial intelligence and, therefore, may not be entirely error-free. Rocket Software does not assume any responsibility or liability for the use or interpretation of this content.

Attributes

Name	Value
Lines Of Code	2,239
Cyclomatic Complexity	56
Dead Lines	598

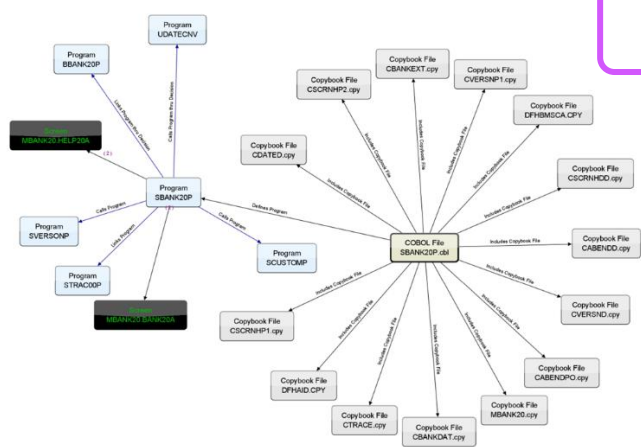
Relations

Relation Name	Object Name
Includes Copybook File	CBANKEXT.cpy
Includes Copybook File	CABENDPO.cpy
Includes Copybook File	CBANKDAT.cpy
Includes Copybook File	CSCRNHP1.cpy
Includes Copybook File	CDATED.cpy
Includes Copybook File	CTRACE.cpy
Includes Copybook File	CSCRNHDD.cpy
Includes Copybook File	CSCRNHP2.cpy
Includes Copybook File	DFHAID.CPY
Includes Copybook File	MBANK20.cpy
Includes Copybook File	CABENDDD.cpy
Includes Copybook File	DFHBMSCA.CPY
Includes Copybook File	CVERSND.cpy
Includes Copybook File	QVERSNP1.cpy
Calls Program	SCUSTOMP
Calls Program	SVERSONP
Calls Program thru Decision	UDATECNV
Links Program	STRAC00P
Links Program thru Decision	BBANK20P
Receives Screen	MBANK20.BANK20A
Receives Screen	MBANK20.HELP20A
Defines Program	SBANK20P

生成されたサマリー文

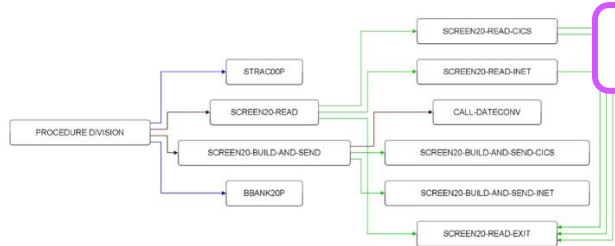
ソースアーティファクト、
コール、データアクセス

Relations Diagram



関係性

Program Control Flow Diagram



フロー図

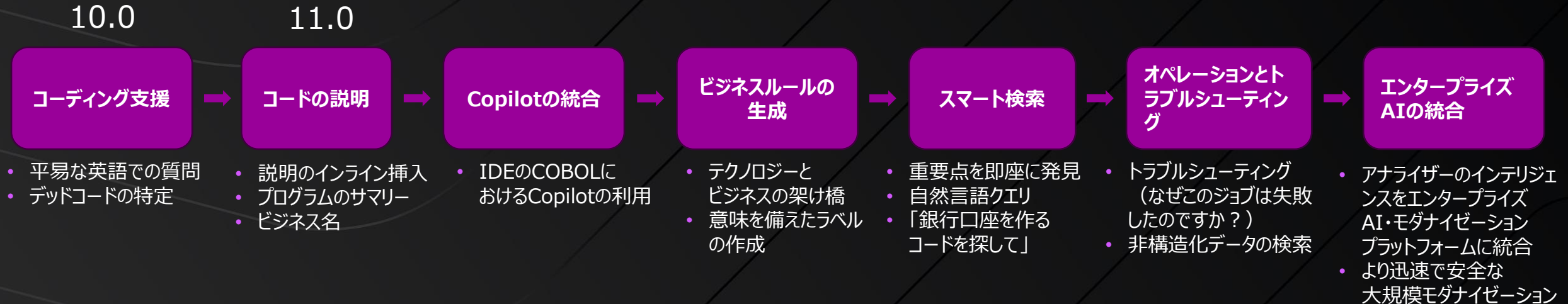
MBANK20.BANK20A



画面出力

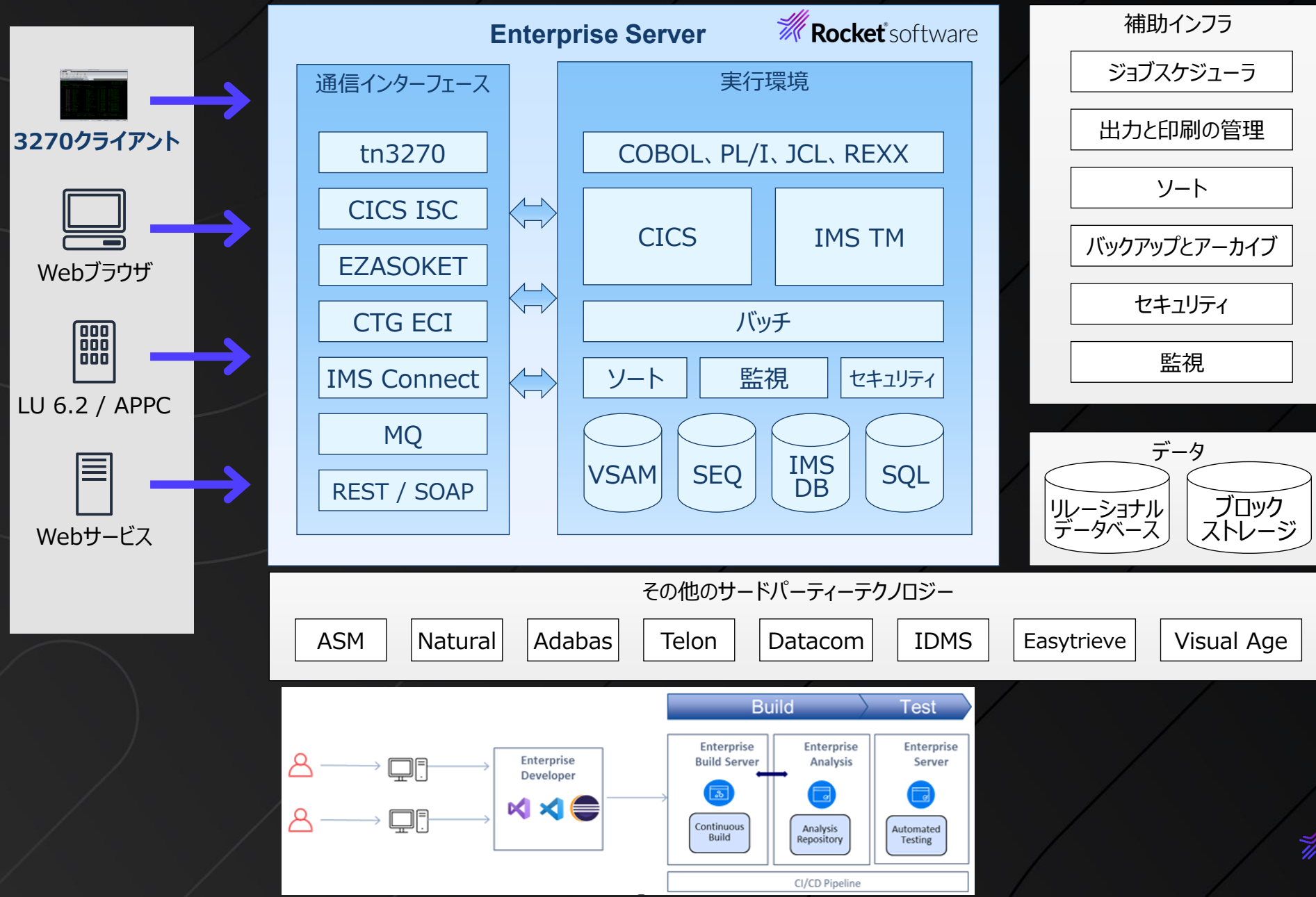
AIロードマップ：今後12～18か月の展望

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

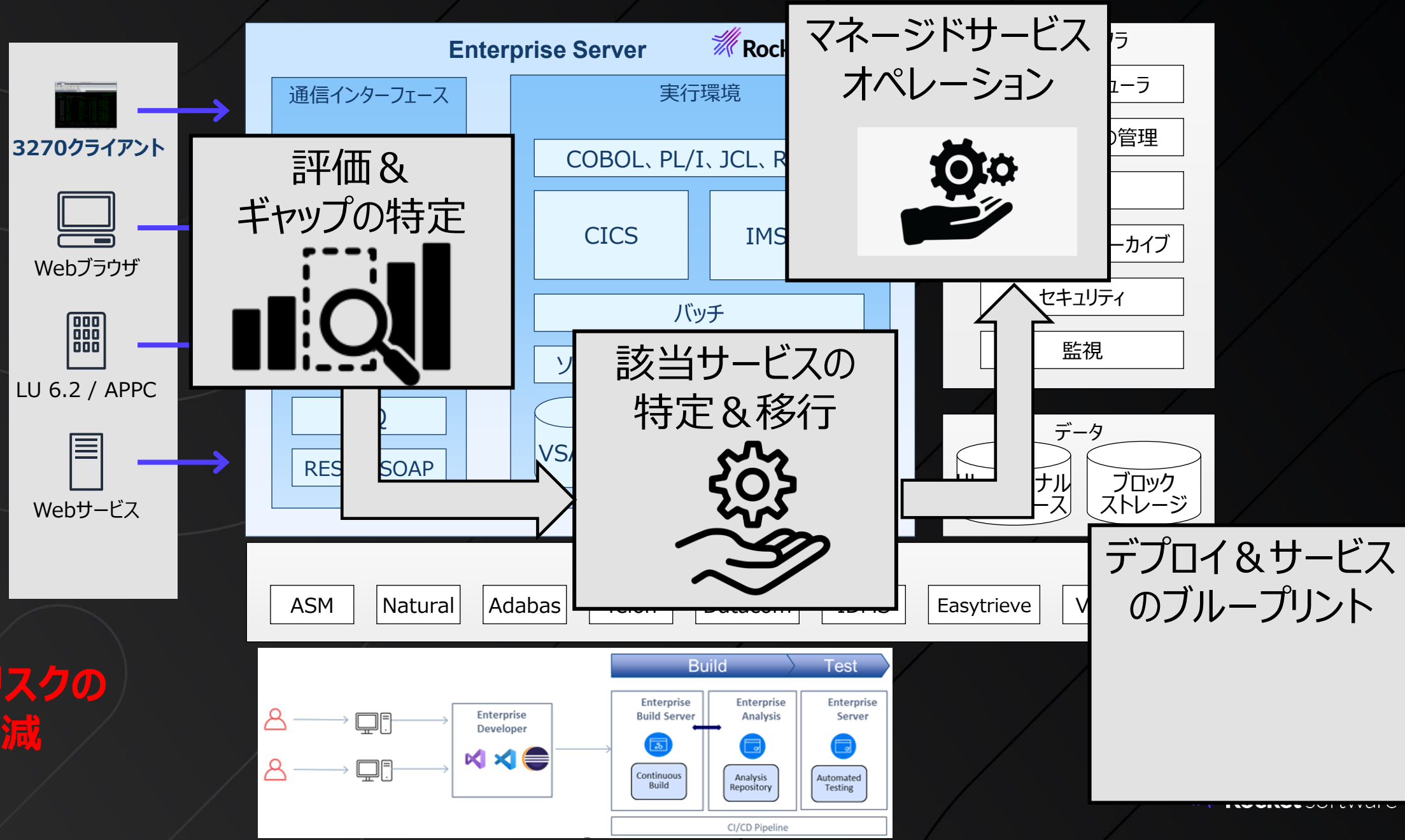


ISO42001準拠 – 責任あるAIの組み込み

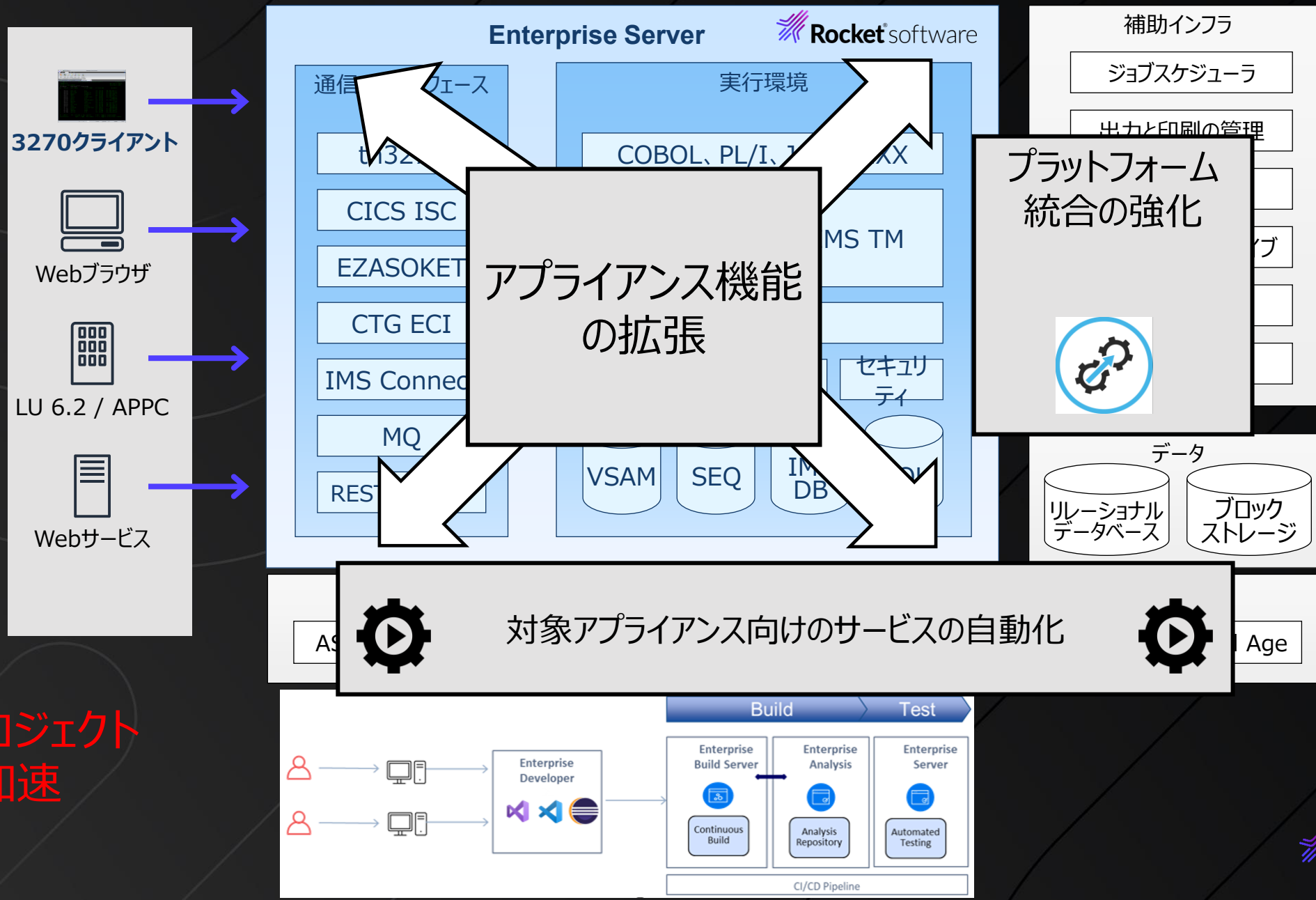
ロケットソフトウェアの分散／クラウドプラットフォームアーキテクチャ参考図



ロケットソフトウェアのアーキテクチャ参考図



ロケットソフトウェアの拡張アーキテクチャ参考図



移行プロジェクト
の加速

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

これぞテクノロジーの逆説



Post



Andrej Karpathy ✓

@karpathy

Follow



The hottest new programming language is English

(今、最もホットな新しいプログラミング言語は英語です)

8:14 PM · Jan 24, 2023 · **5.5M** Views

930

5.8K

36K

3.2K



Thank you

