

**UNITIKA**  
We Realize It!

# JUASスクエア2020 オンラインペアセッション

## ユニチカ事例から見るITトレンドの選別・導入と業務 革新 – システムリフォームの進化

ユニチカ株式会社  
管理本部 情報システム部  
2020年7月29日

# Contents

1. 会社概要
2. プロジェクト概要（基幹システム・リブレース）
  - i. プロジェクト背景
  - ii. プロジェクト概要
3. システムリフォーム・ユニチカモデルとは
  - i. システムリフォームとは
  - ii. ユニチカモデルとは
  - iii. システムリフォーム（ユニチカモデル）を実現するためのアプリケーション基盤
4. 今後の予定（DX化 2025年の崖に向けて）
  - i. アーキテクチャ
  - ii. 保守・改善・運用

# 会社概要

**UNITIKA LTD.**

# ユニチカ株式会社

創立	1889年（明治22年）6月19日
資本金	100,450,000円
主な事業内容	<ul style="list-style-type: none"><li>•高分子事業 フィルム（ナイロン・ポリエステル） 樹脂（ナイロン・ポリエステル・ポリアリレート）</li><li>•機能材事業 機能材（ガラス繊維、ICクロス、 ガラスビーズ、活性炭繊維） 不織布 （ポリエステルспанボンド、綿спанレース） 生分解材料</li><li>•繊維事業 産業資材 衣料・生活雑貨・寝装 バイオマスプラスチック（テラマック）</li></ul>



UNITIKA LTD.

# プロジェクト概要

(基幹システム・リプレイス)

# プロジェクト概要

## プロジェクト背景 ～現行ホスト利用継続の場合の問題～

### ▪ ホスト機の保守切れ（2021年3月）

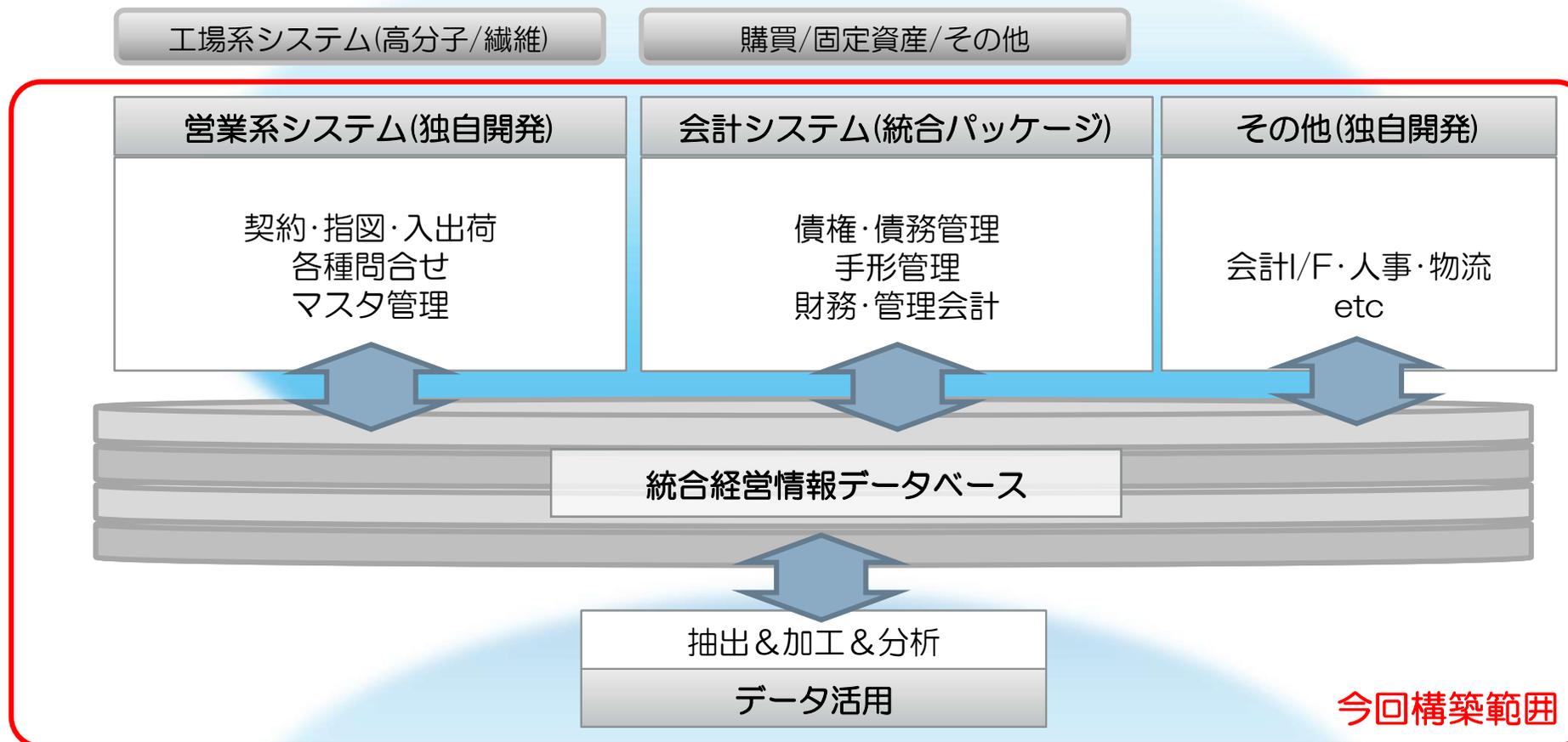
- バージョンアップに必要な作業期間は、6年
- バージョンアップにかかる費用は、数億円規模

### ▪ バージョンアップを完了したとしても

- 古くてマイナーな言語（PL/1）の保守要員が将来的に確保困難。
- ホストのランニング費用は高額のまま。
- 監査法人より、監査、統制面で品質向上の要望がきている。
- 経営判断に要するデータの収集、加工についてスピード、粒度について課題多数。
- このままでは、他システム連携や取引先との密な連携、グローバル対応などが困難。

# プロジェクト概要

## プロジェクト概要 ～対象システムの構成～



UNITIKA LTD.

# プロジェクトの歩み

2011年3月～2012年12月

SWINGプロジェクト

SOAによる基幹システムの再構築

※パイロット的な位置付け

2013年10月～2017年10月

COMPASSプロジェクト

**システムリフォーム**により、  
ホストシステムと同じ機能を実現

ユニチカ  
+  
大手SIer

方向転換

ユニチカ  
+  
SoftRoad

成功

- ▶ そもそも基幹システムの再構築は困難
- ▶ コスト、開発期間の制約

# 方向転換の理由

- そもそも基幹システムの再構築は困難
  - 規模が大きい（画面数1400以上、JOB数2200以上）
  - 仕様書が無い
  - 基幹システムが多くその他システムと連携している
- コスト、開発期間の制約
  - メインフレームの保守切れのタイムリミット
  - 今の業務を急に変更することが困難
  - 今の仕様と合致していることを確認できる有識者が不足

これらの問題をシステムリフォームが解決してくれた。

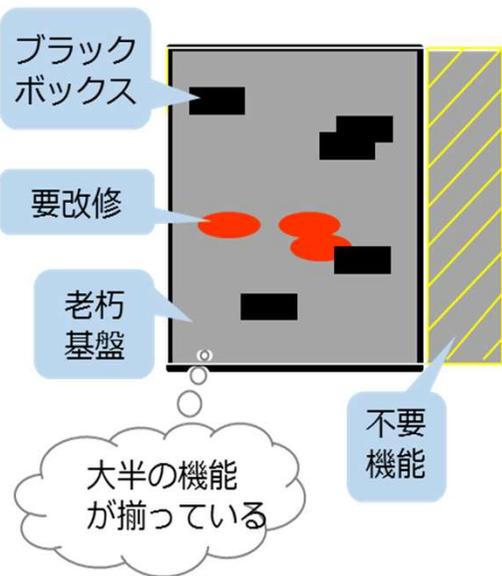
# システムリフォーム・ユニチカ モデルとは

# システムリフォーム・ユニチカモデルとは

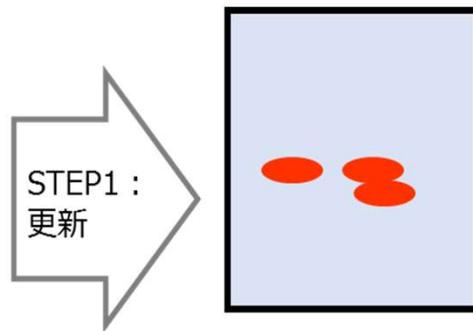
## システムリフォームとは

既存のシステムリソースをいかしながら、システムを最新のIT基盤にリフォーム（引越し）する手法です。

### 1. 既存システム

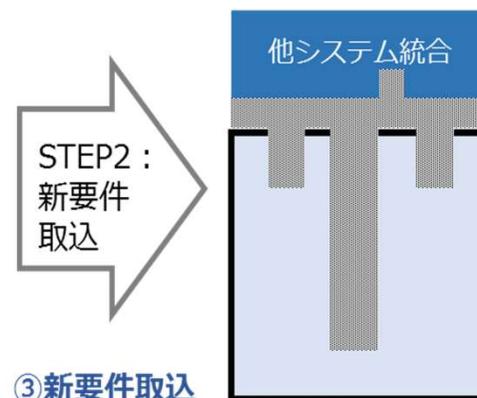


### 2. 更新後 中間システム



- ① **新技術に移行する**  
ツールで言語変換する。
- ② **スリム化・構造変更・見える化**  
ツールと手作業で、  
不要ソース削除、構造改善、  
設計書再生

### 3. 修正後の 最終システム



- ③ **新要件取込**  
機能の改修・追加

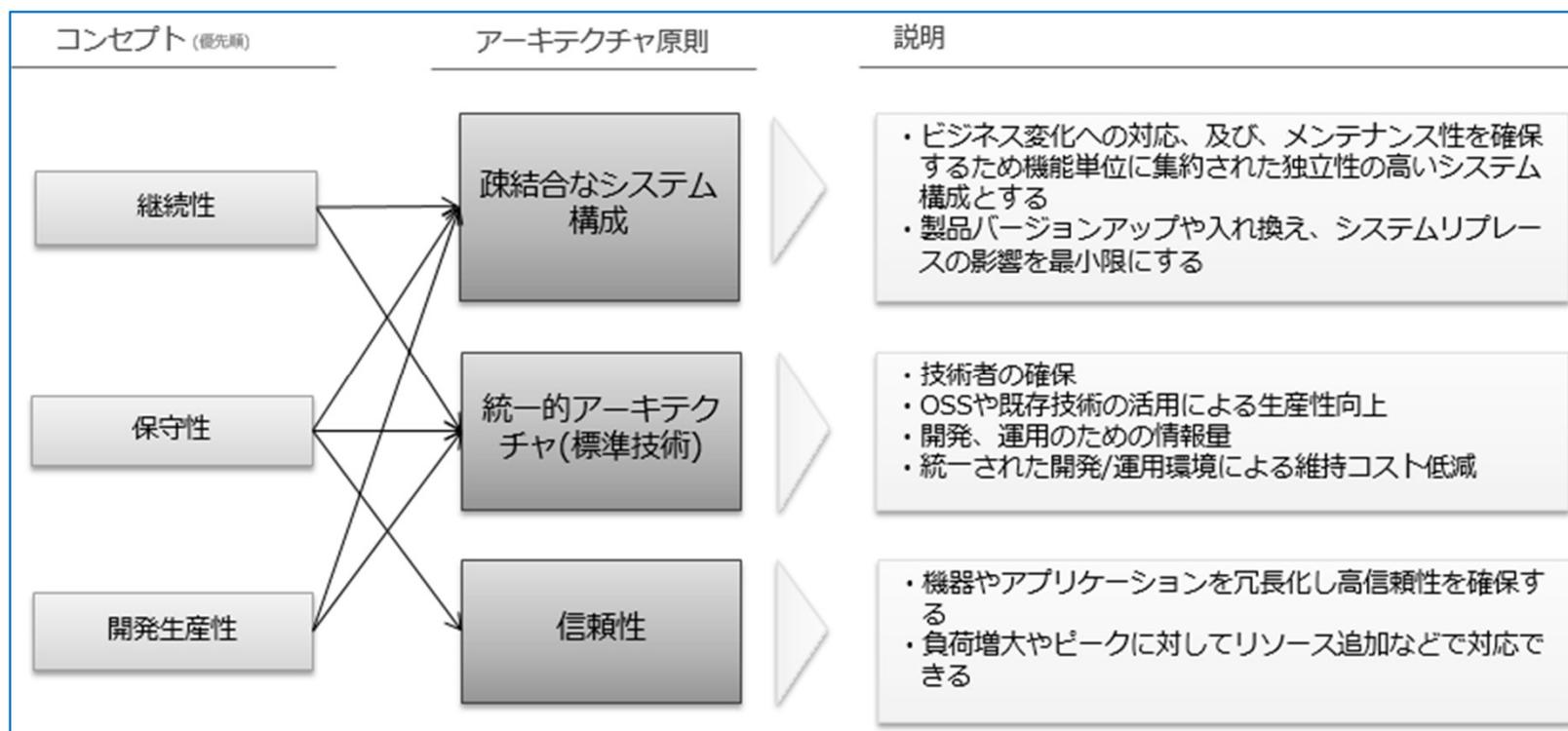
## ・メリット

- 開発コストが抑えられる。
- 通常のスクラッチ開発と比較して品質が高い。
- 発注側（ユニチカ側）の作業負荷（調査工数等）が軽減される。

# システムリフォーム・ユニチカモデルとは

## ユニチカモデルとは ～アーキテクチャ（原則）～

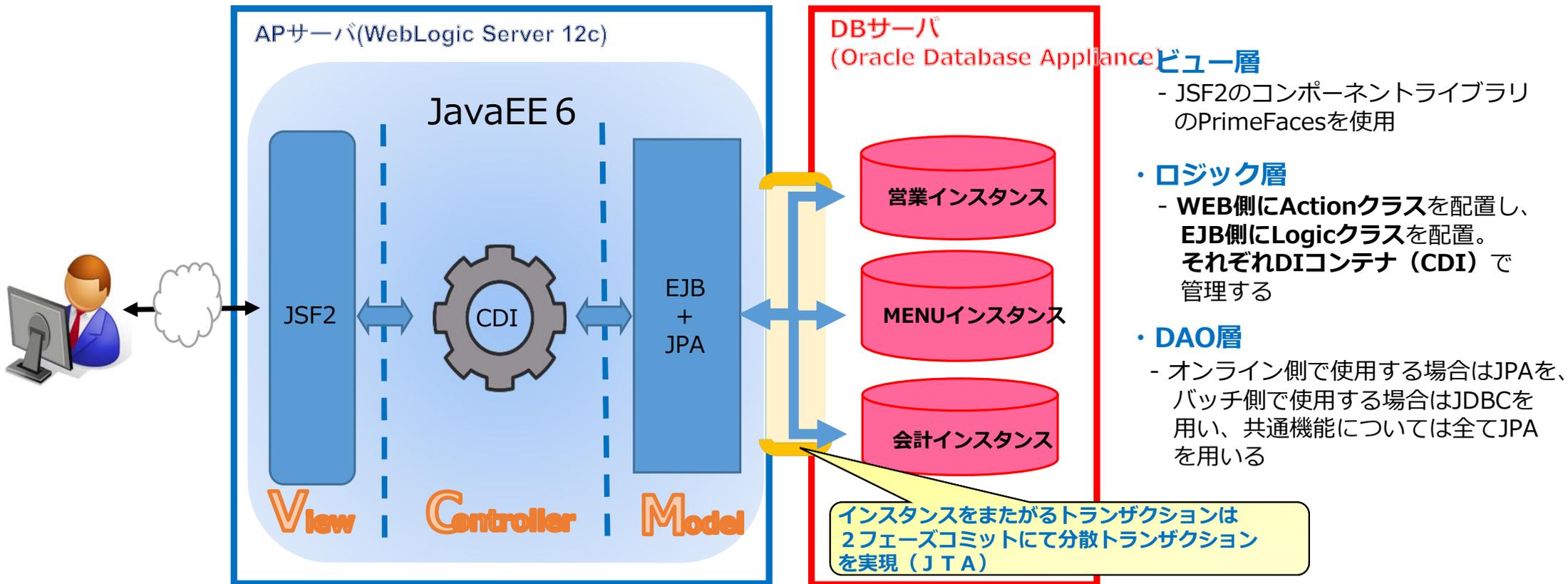
- アプリケーション基盤の構築にあたり、  
3つのコンセプトを元にアーキテクチャ原則を定めた。



# システムリフォーム・ユニチカモデルとは

## ユニチカモデルとは ～アーキテクチャ編（1）～

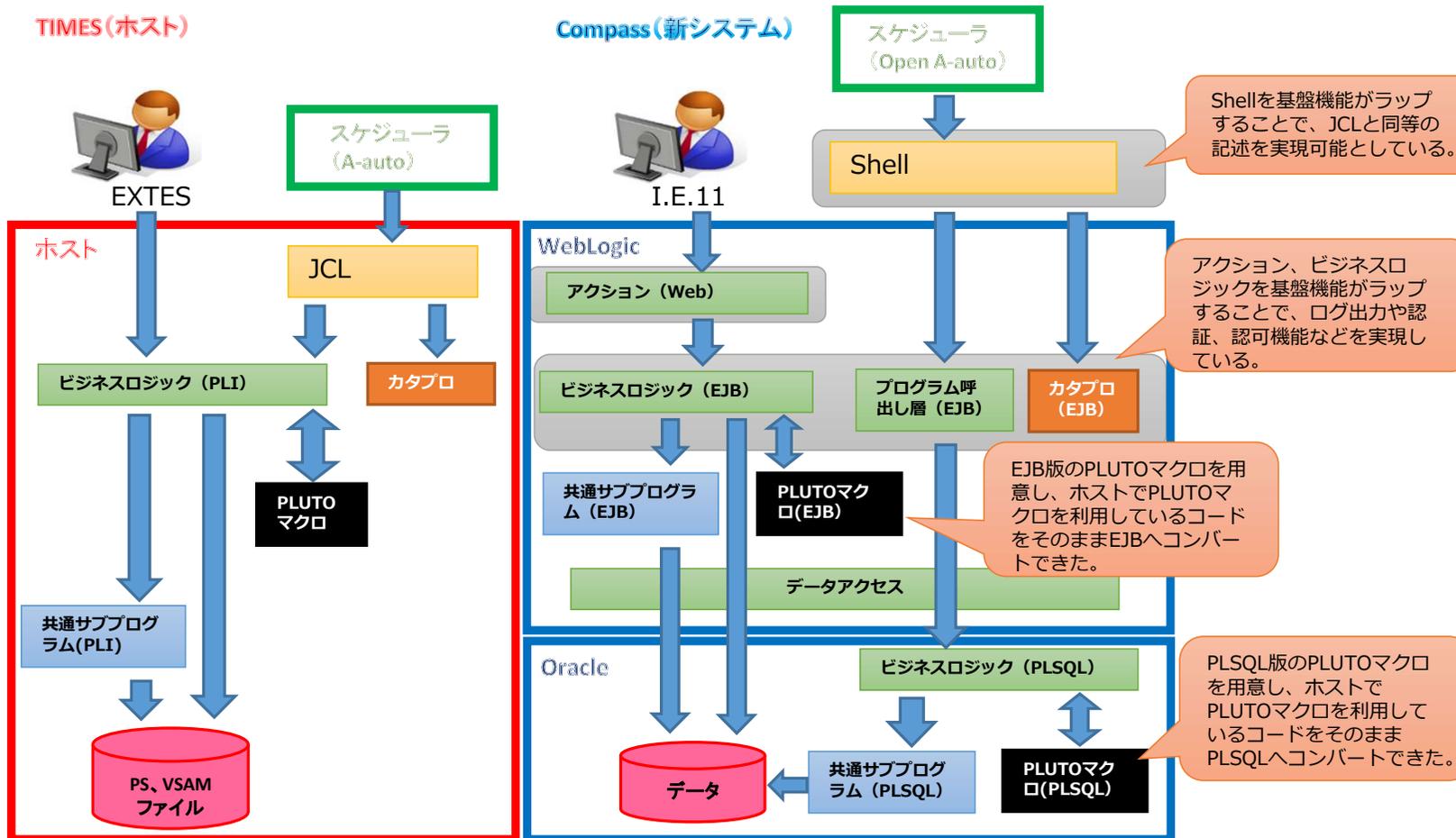
Compassプロジェクト共通で利用するWebフレームワークを作成。  
JavaEE6のフルサポート及び代表的なOSSに拘った製品群でアーキテクチャを構成している。



UNITIKA LTD.

# システムリフォーム・ユニチカモデルとは

## ユニチカモデルとは ～アーキテクチャ編 (2)～



- DIコンテナによりロジック層とビュー層、DAO層を分離し疎結合を担保。
- 更に、アプリ基盤でサンプル画面やテンプレートを用意し、できる限り疎結合の関係を保てるように工夫
- テンプレート例
  - プロパティクラス
  - Actionクラス
  - DTO&Logicクラス
  - 画面XHTML
- サンプル例
  - メニュー登録
  - グループメニュー登録
  - パターン別入力画面
- オンライン開発実例等も用意  
120頁超の詳細なサンプル説明資料も
  - オンライン編
  - バッチ編
  - ロジック編
 に分けて提供

# システムリフォーム・ユニチカモデルとは

## ユニチカモデルとは ～データベースデザイン編（1）～

レガシーファイル→RDBMS(Oracle)の変換を実現。これをシステムリフォームに適用可能とした、ルールの一部を紹介します。

### 配列要素 > 第1正規化

配列パターン(第一正規化)

- 要素数が4以上の配列は、子テーブル化。

<イメージ>

現行形式

APRMSTR(契約マスタ)

等級1～3	単価1～3	...	色柄明細1～10
-------	-------	-----	----------

新テーブル



APT\_MSTR(契約マスタ) 主テーブル

主キー	等級1～3	単価1～3	...
-----	-------	-------	-----

等級・単価など3以内のものは主テーブルにそのまま横持ちで格納

5	1	KYC#FD	10	712	
7	2	C#		712	5 CHA
8	2	C#KYQ		717	72 PAC
9	2	C#SZQ		721	72 PAC
0	2	C#FNSQ		725	72 PAC
1	2	C#ONSQ		729	72 PAC
2	1	KYC#FD	10	712	21 CHA
3	1	KYSKNA		922	25 CHA
4	1	KYSENA		947	25 CHA
5	1	NONAME		972	22 CHA

APT\_MSTR\_KYCNOFD(契約マスタ色柄明細) テーブル

+

主キー	明細No	色柄明細情報
-----	------	--------

色柄明細、4以上のものは明細テーブルに分割、縦持ちで格納

(注意事項)

- ① 基本的には、上記のルールだが、ユニチカ希望により(将来を見越してor処理簡素化)、3つ以内だが、明細テーブルに持たせたい場合、4つ以上だが、明細に分けたくない場合が出てくるので、設計後、一旦ユニチカのレビュー確認後、開発作業に入るようにしていただく。

UNITIKA LTD.

# システムリフォーム・ユニチカモデルとは

## ユニチカモデルとは ～データベースデザイン編（2）～

### ・ 複合コード > 最小単位ルール

- 最も小さい単位で、1項目とする。

<イメージ>

現行形式

組織明細		
会社	部門	...



新テーブル

会社	部門	...
----	----	-----

最小単位でDB項目とする。

2	KYLSN		112	1	CHAR			契約課内優先区分
1	KYSKM		102	11	CHAR	102		約定担当の組織明細
1	KYS		113					契約先

※) 同種の例

契約No、指図No、入出荷No。 これらは現行、下記の構成となっているが、それぞれ新テーブルとしては最小単位で項目を持つ

<イメージ>

入出荷No			
指図No		枝3	
契約No	枝2	枝3	
発連	枝1	枝2	枝3



新テーブル

発連	枝1	枝2	枝3
----	----	----	----

発生連番

但し、現行システムで、例えば"入出荷No"として項目にアクセスしている箇所があれば、JavaのEntityオブジェクトに、入出荷NoとしてアクセスするGetter/Setterを定義し"入出荷No"として扱えるように新システムをコーディングするものとする。

(例外事項)

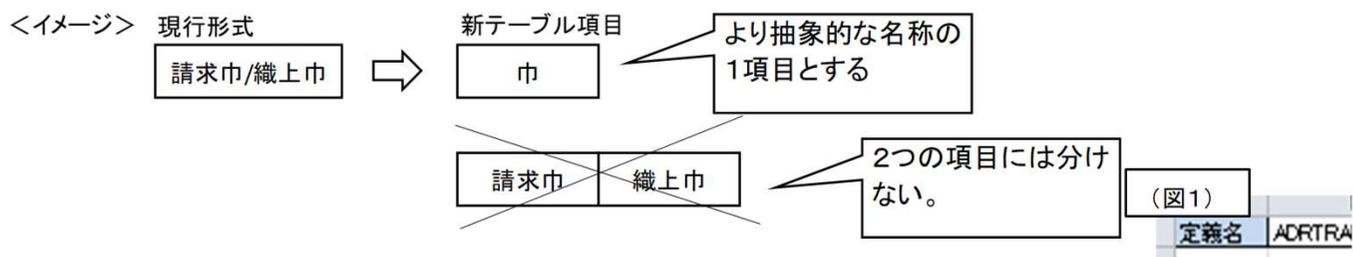
① 下記の項目については、最小単位で設定しない。

- A) 最終更新日時・・・最小単位ルールに従うと、「最終更新年月日」「時」「分」に分かれてしまうが、そうせずDB設計規約に従った最終更新日時項目に置き換わる。(※) 決済条件は例外でなく
- B) 物的属性・・・最小単位ルールとすると「物的属性1桁目」「2桁目」に分かれるがそうせず、新テーブルでは、「物的属性」(2桁)の1項目として定義する。 項目を分けてDB化する

# システムリフォーム・ユニチカモデルとは

## ユニチカモデルとは ～データベースデザイン編 (3)～

- 1項目複数意味項目（同一タイプ・レイアウト） > 項目名の抽象化



- 1項目複数意味項目（別項目格納） > 項目分割

- 別項目として定義する。



# システムリフォーム・ユニチカモデルとは

## ユニチカモデルとは ～データベースデザイン編 (4)～

### マルチレイアウト > テーブル分割

<イメージ>

現行形式

APRMSTR(契約マスタ)

主キー	共通部分	識別区分	レイアウト01/レイアウト02

マルチレイアウト部分

4	2	GGNTNT		616
5	2	GTEHM		617
6	2	GSHQRJ		619
7	2	SKTNR		626
8	2	GSKKT		633
9	2	DBTKZK		634
0	2	DSHN		636
1	2	DTPN		640



新テーブル

APT\_MSTR(契約マスタ) 主テーブル

主キー	共通部分	識別区分

+

APT\_MSTR\_APM01 子テーブル(レイアウトAPM01)

主キー	レイアウト01

APT\_MSTR\_APM02 子テーブル(レイアウトAPM02)

主キー	レイアウト02

### 前述の基本ルール決定に加えて、新テーブルレイアウト設計については、全レイアウトをレビュー

- 例外項目、例外テーブルについては、レビュー指摘により妥当に訂正
- レビュー確定後の新テーブルレイアウトを加味した、移行開発を実施

# システムリフォーム（ユニチカモデル）であればこそ 容易に実現できた事例①：DB正規化

- 樹脂の原料個数の拡張（4レコード → nレコード）
- テーブルを正規化（子テーブル化）し、入力項目数の拡張が容易となった。

HPWQ041 委託生産契約（新規） Version: NOVERSION

CU001011: データを登録しました(契約No.=GPP7620) 伝送先(M868Z00)に送信します

部課/先部課/契約No. V41BA

契約日 / 委託先 M868Z00 XXXX> XXX'XX XK XXXd

加工場 M868Z00 XXXX> XXX'XX XK XXXd

納入日 / 納入先 0426

製品 / 単品 E101 SRRU/ 88-X001

摘要 / ロット AAAAA BBBB 梱包形態指定 99

数量 / 単位 10000 K

加工費単価単位 K

A格単価 50000 輸入形態 仕向地

B格単価 30000 OR B格支払率 %

C格単価 10000 OR C格支払率 %

決済条件 F055B055 ユニチカ連絡 NO

備考

C/#	数量								

原料	製品	単品	摘要	ロット	期間	K	当り量
(1)	E106	1050/ 42- HYB			5	0.1	K
(2)	K471	EBA-16G			6	0.2	K
(3)	Z095	U.M.Z			7	0.3	K
(4)	E470	ASP-SD			8	0.4	K

指図先 M868Z00

戻る 確認 登録 連続入力

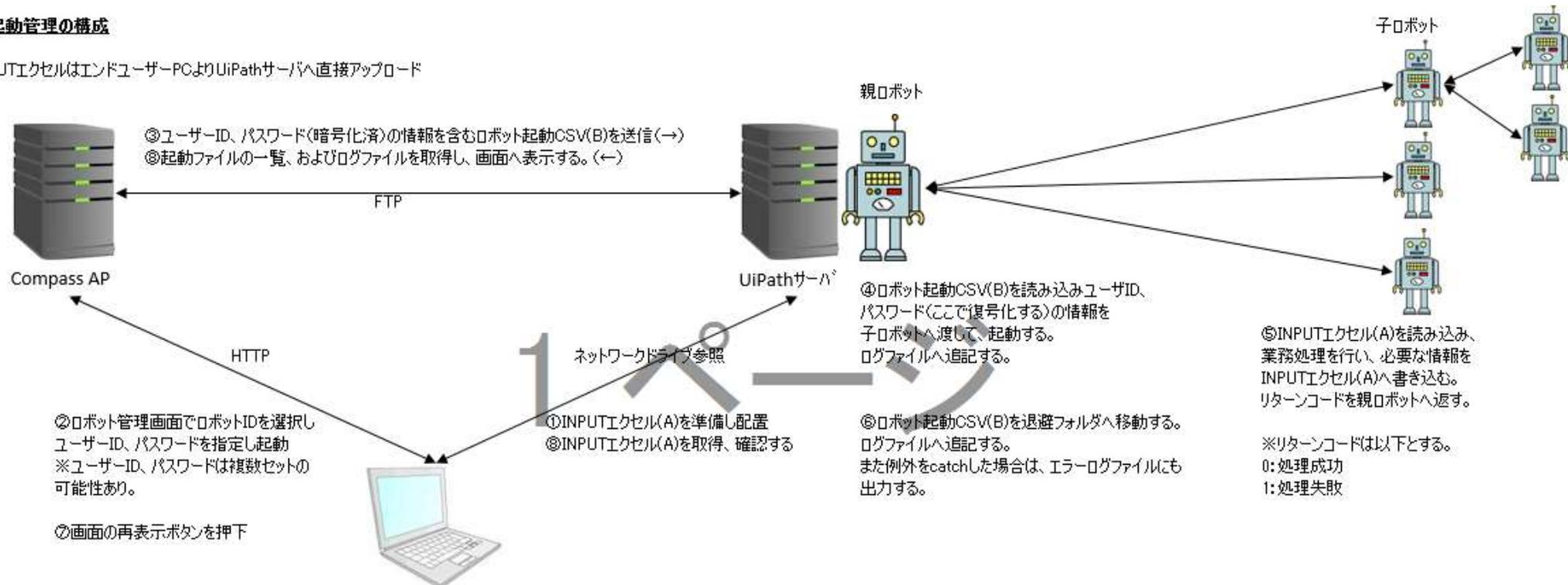
生産に必要な原料  
明細 4 行を n 行に拡張する

# システムリフォーム（ユニチカモデル）であればこそ 容易に実現できた事例②：外部アクセス機能

- RPA（Robotic Process Automation）ロボットの管理機能  
FTP通信、暗号/復号化などの基盤を整備し、実現可能となった

## ロボット起動管理の構成

案1) INPUTエクセルはエンドユーザーPCよりUiPathサーバへ直接アップロード

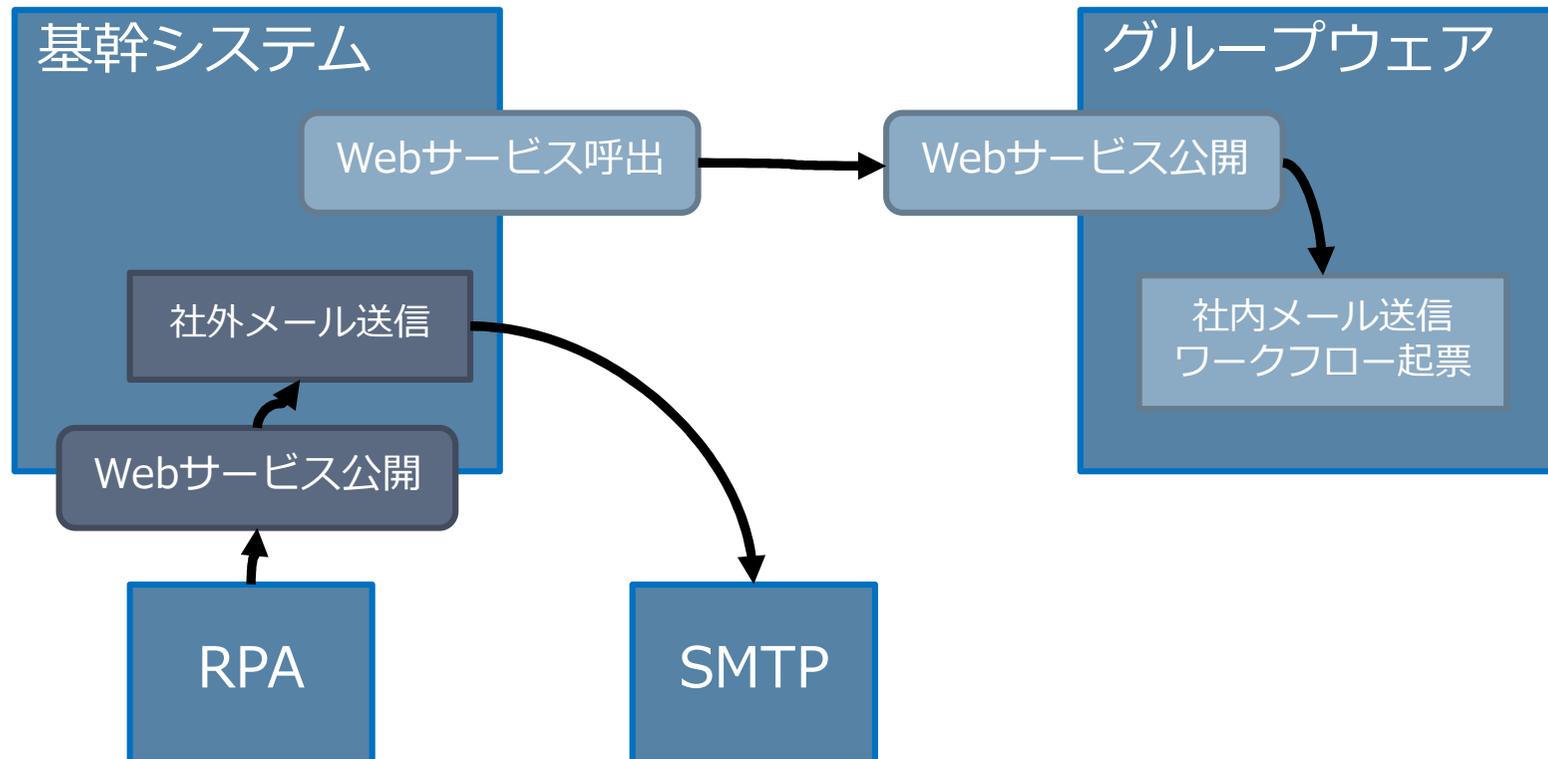


UNITIKA LTD.

# システムリフォーム（ユニチカモデル）であればこそ 容易に実現できた事例③：Webサービス連携

- 社内メール、ワークフローシステムの起票、外部メールサービスの公開

Webサービス関連の基盤を整備し、実現可能となった



## システムリフォーム（ユニチカモデル）であればこそ容易に実現できた事例④：大文字強制入力

- ユニチカのコード体系は半角アルファベット小文字「a,b,c,・・・」は使用せず、全て大文字「A,B,C,・・・」となっている。
- ホストの画面は大文字強制入力（**CAPSロックされていない状態でも大文字が入る**）となっていた。
- 本番開始後、利用部門よりCAPSキーをいちいち切り替えたりSHFTキーを押すのが面倒という問い合わせ。
- アプリケーション基盤を拡張し、大文字強制入力を実現した。

全システムへ影響するため  
かなりリスクのある変更

The screenshot shows a software window titled "Compass 【受入1】 GSWQ010 サンプル条件一覧明細 (条件)". The form contains the following fields:

- 部課課内: AAAAAA [検索]
- 処理区分:  新規  変更  取消  照会
- 製品: BBBB
- 単品: CCCCCC [X]

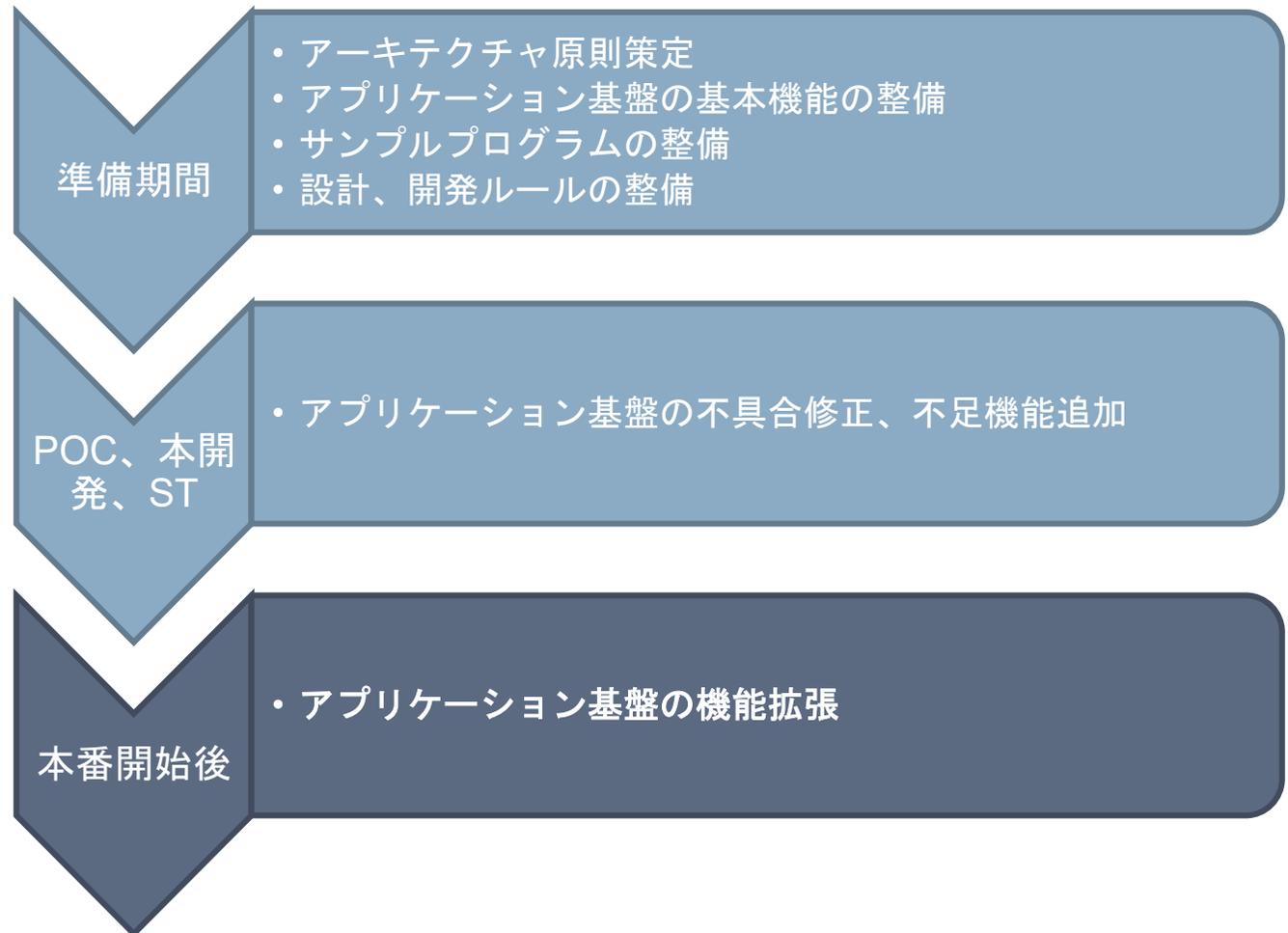
# 1. 今後の予定 (DX化 2025年の崖に向けて)

# 今後の予定（DX化 2025年の崖に向けて）

## 基盤の洗練・拡張

これまで、大規模な基幹システムを効率的に移行するためのベースラインアーキテクチャを構築してきましたが、今後、長期にわたって保守・拡張していける**エンタープライズな基幹システムに発展させるべく、アーキテクチャを洗練・拡張させていく。**

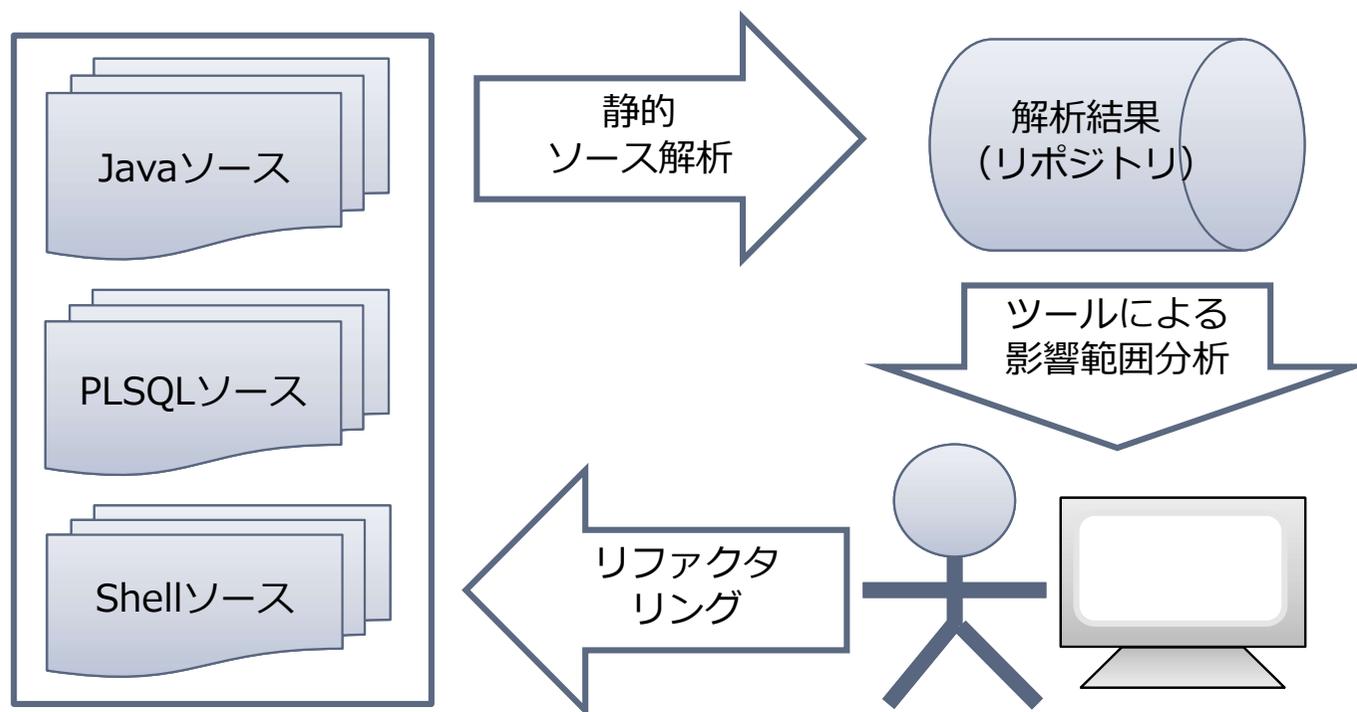
アプリケーション基盤構築の初期メンバーを抱えることで、本番開始後の拡張のリスクを軽減



# 今後の予定（DX化 2025年の崖に向けて）

## リファクタリング

- 静的ソース解析の情報を利用し、利用されないロジックの削除や、部分的な置換えにより段階的に改善していく。

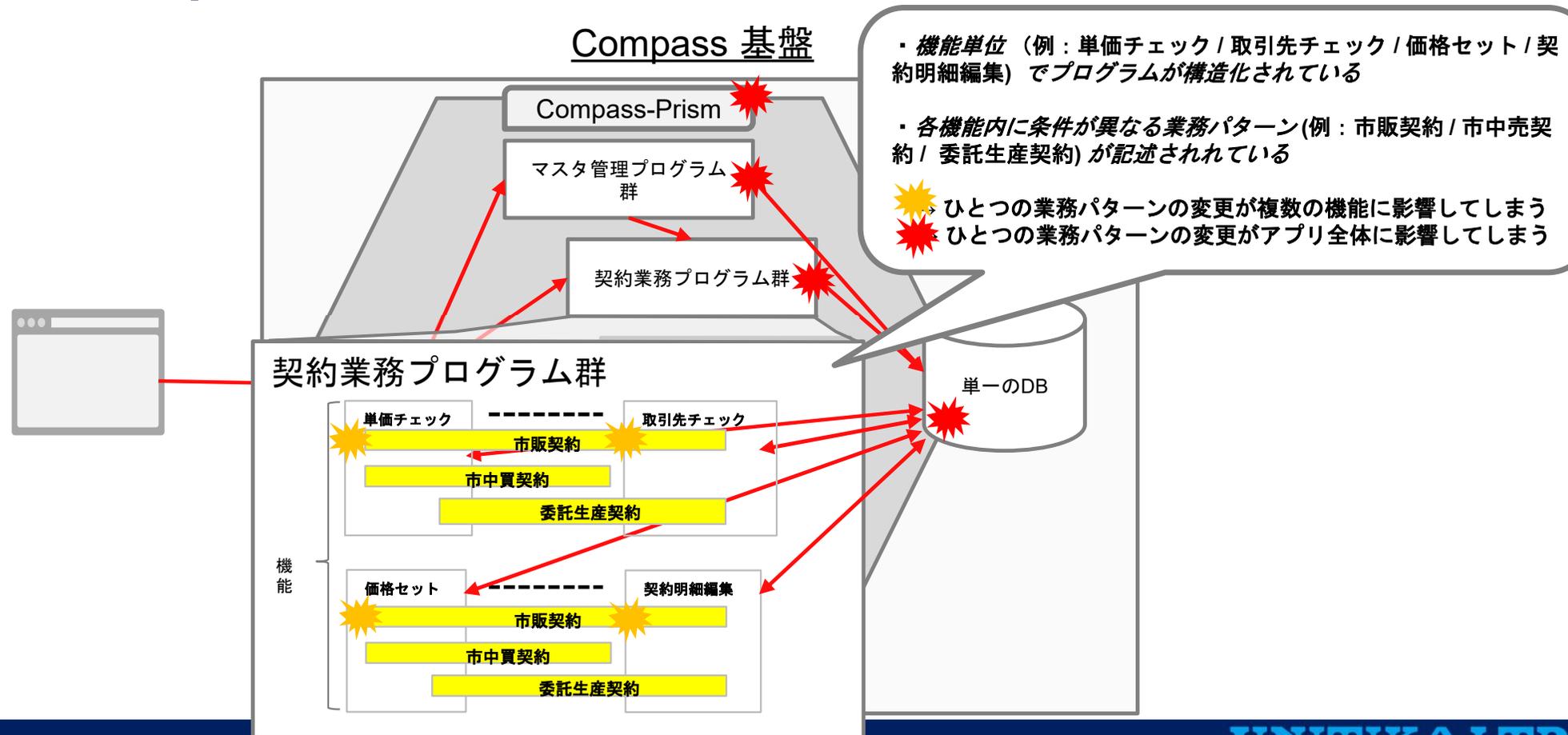


- 改善内容：  
コードのシンプル化  
変数スコープの限定  
⇒ **マイクロサービス化**

- リファクタリング目的：  
パフォーマンス向上  
保守性向上

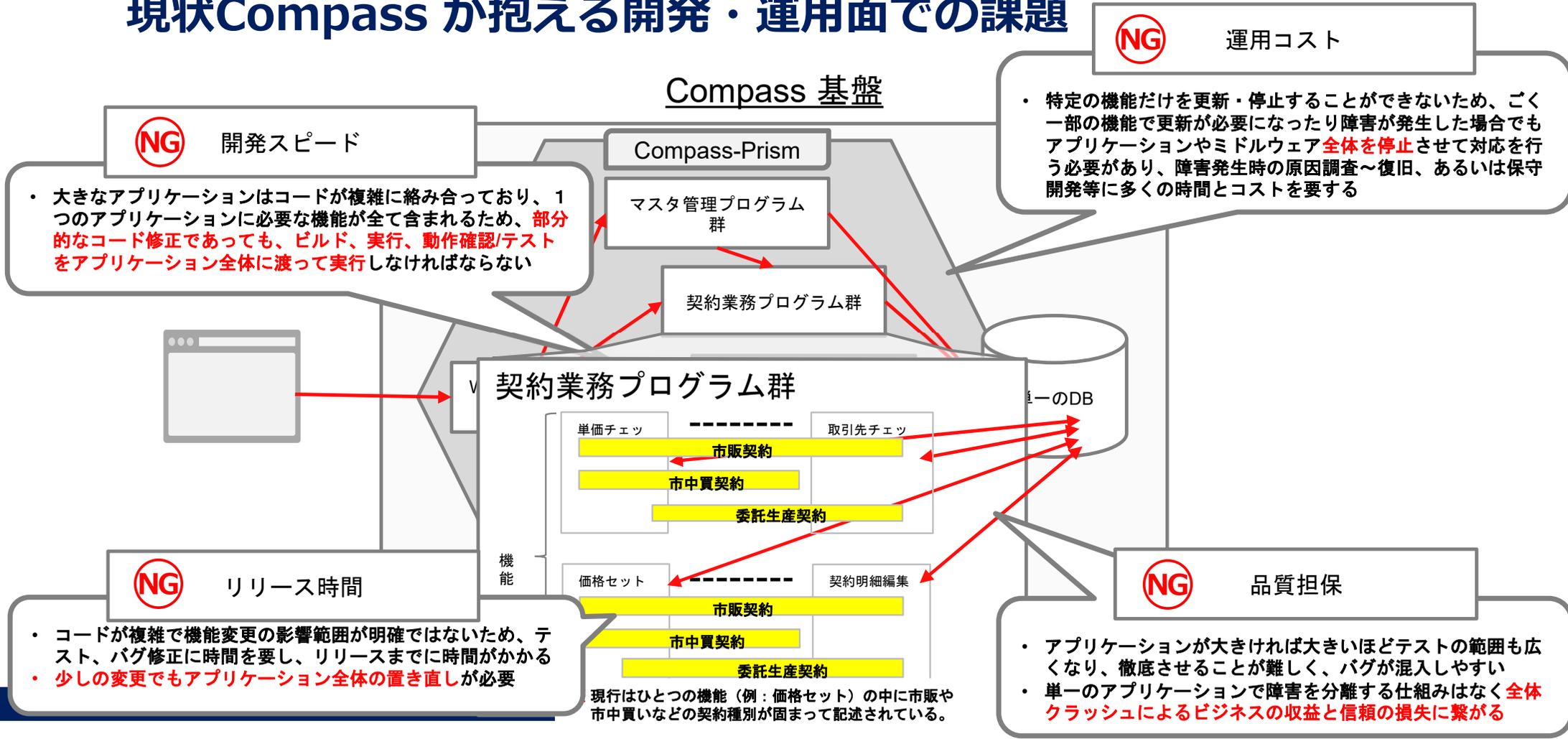
# 今後の予定（DX化 2025年の崖に向けて）

## 現状Compass 基盤上の実装イメージ



# 今後の予定（DX化 2025年の崖に向けて）

## 現状Compass が抱える開発・運用面での課題



**NG** 開発スピード

• 大きなアプリケーションはコードが複雑に絡み合っており、1つのアプリケーションに必要な機能が全て含まれるため、部分的なコード修正であっても、ビルド、実行、動作確認/テストをアプリケーション全体に渡って実行しなければならない

**NG** 運用コスト

• 特定の機能だけを更新・停止することができないため、ごく一部の機能で更新が必要になったり障害が発生した場合でもアプリケーションやミドルウェア全体を停止させて対応を行う必要があり、障害発生時の原因調査～復旧、あるいは保守開発等に多くの時間とコストを要する

**NG** リリース時間

• コードが複雑で機能変更の影響範囲が明確ではないため、テスト、バグ修正に時間を要し、リリースまでに時間がかかる  
 • 少しの変更でもアプリケーション全体の置き直しが必要

**NG** 品質担保

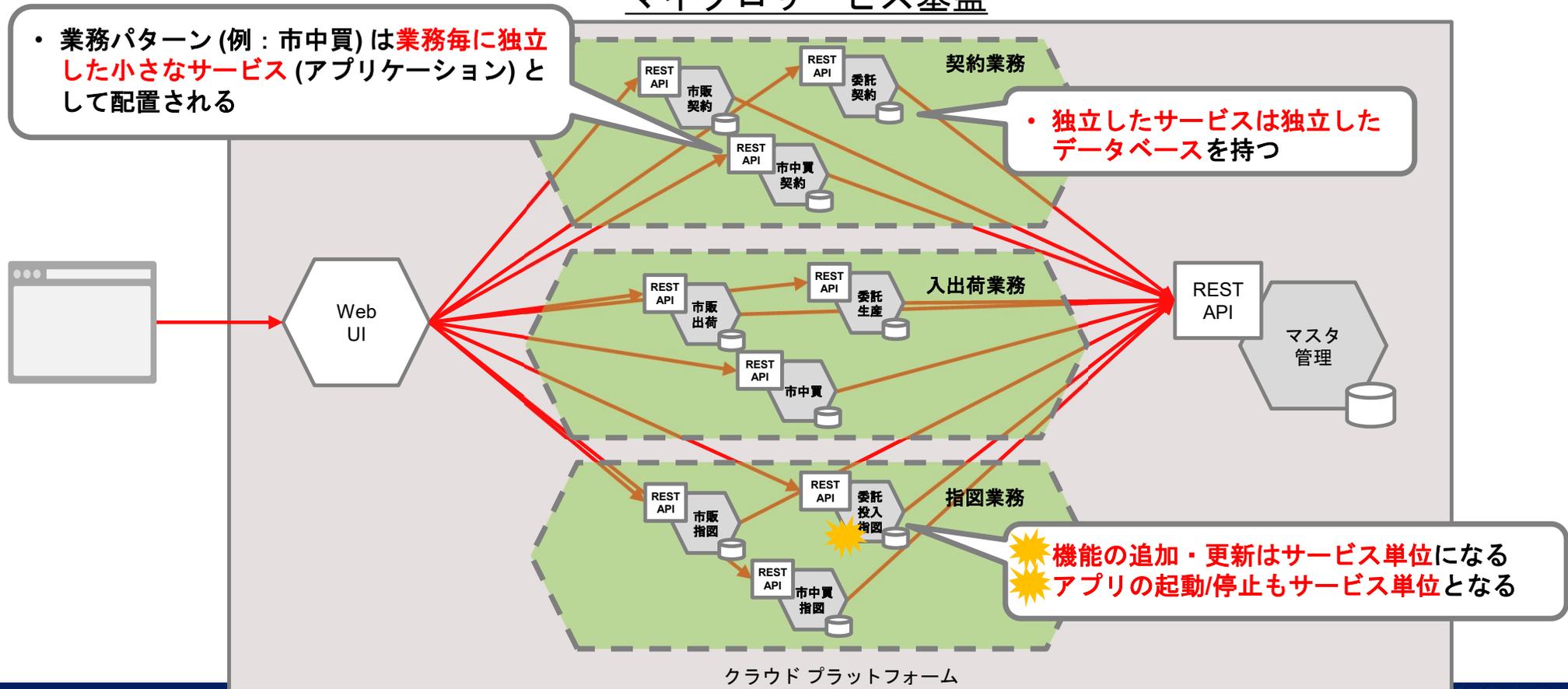
• アプリケーションが大きければ大きいほどテストの範囲も広くなり、徹底させることが難しく、バグが混入しやすい  
 • 単一のアプリケーションで障害を分離する仕組みはなく全体クラッシュによるビジネスの収益と信頼の損失に繋がる

現行はひとつの機能（例：価格セット）の中に市販や市中買などの契約種別が固まって記述されている。

# 今後の予定（DX化 2025年の崖に向けて）

## マイクロサービス基盤上の実装イメージ

### マイクロサービス基盤



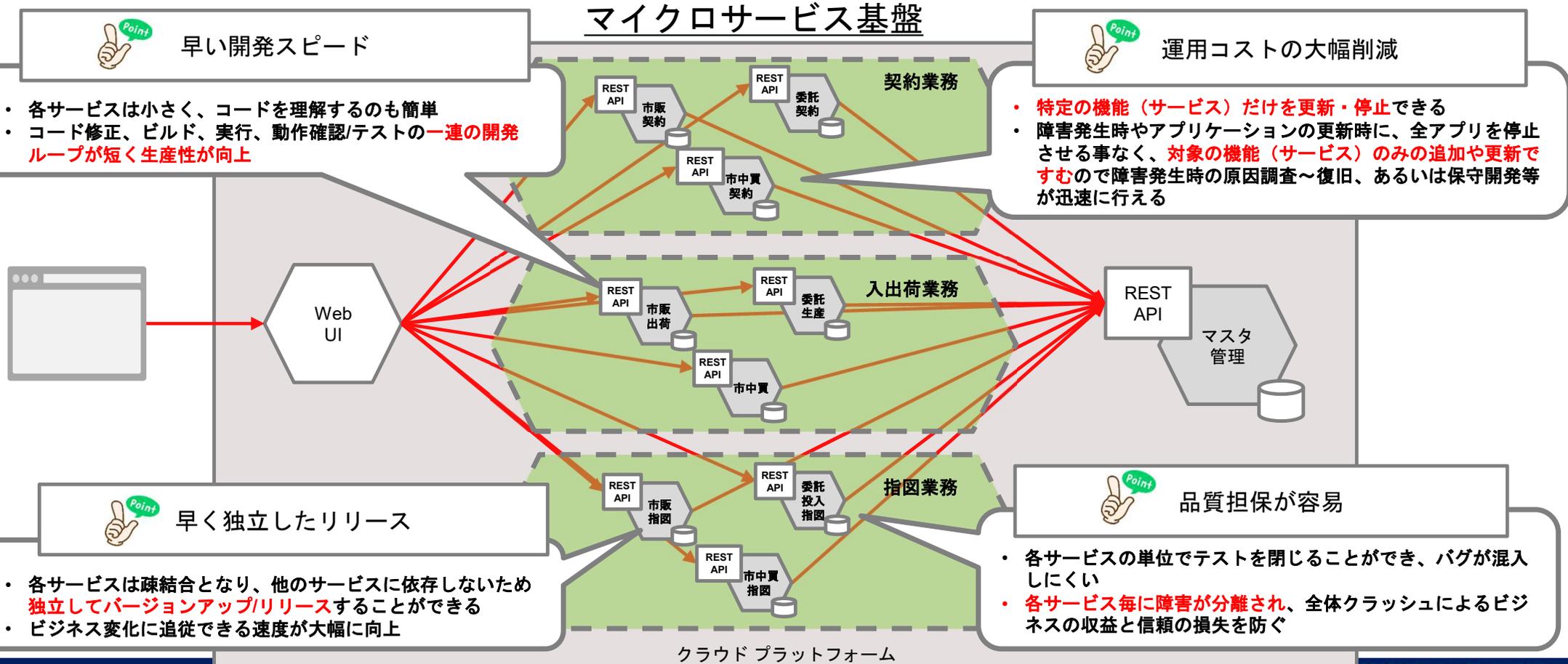
• 業務パターン(例：市中買)は業務毎に独立した小さなサービス(アプリケーション)として配置される

• 独立したサービスは独立したデータベースを持つ

★ 機能の追加・更新はサービス単位になる  
★ アプリの起動/停止もサービス単位となる

# 今後の予定（DX化 2025年の崖に向けて）

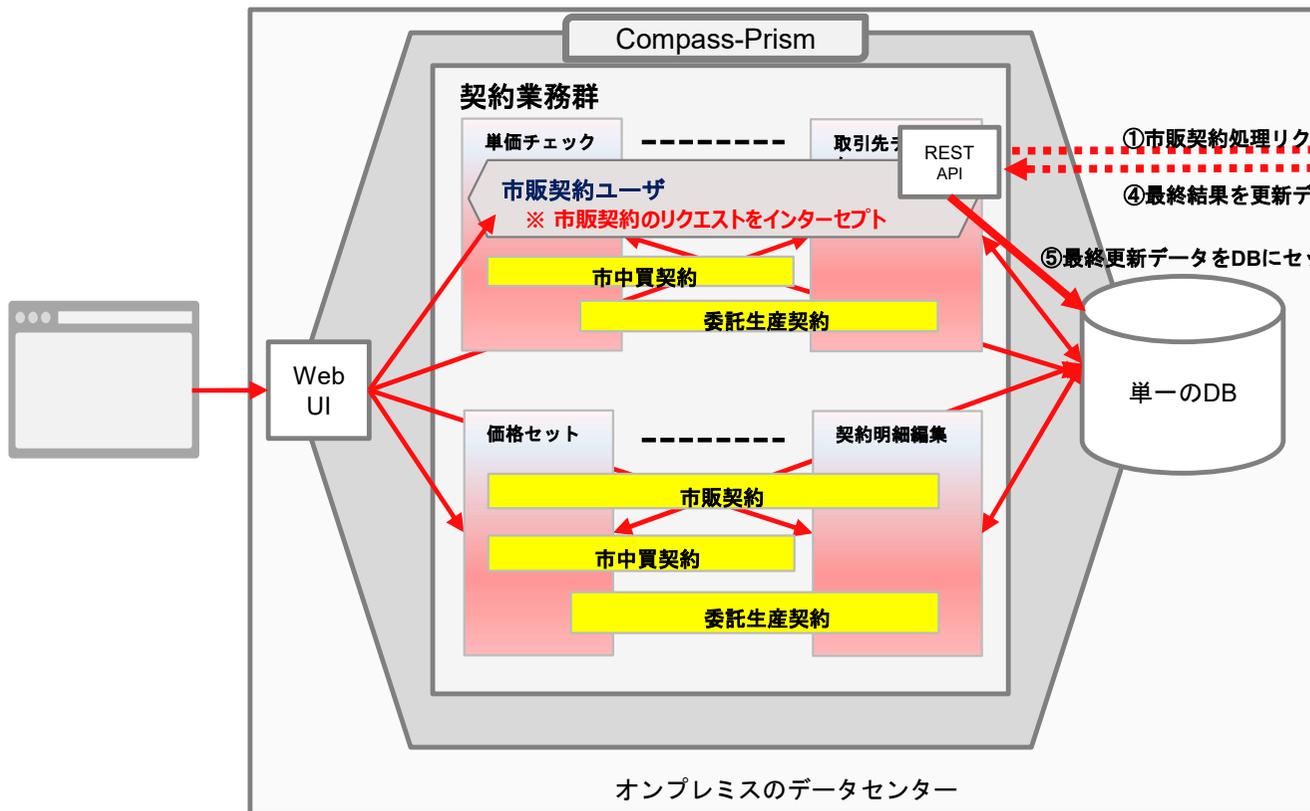
## マイクロサービス基盤上の実装イメージ



# 今後の予定（DX化 2025年の崖に向けて）

## Compass 基盤とマイクロサービス基盤の併用

### Compass 基盤

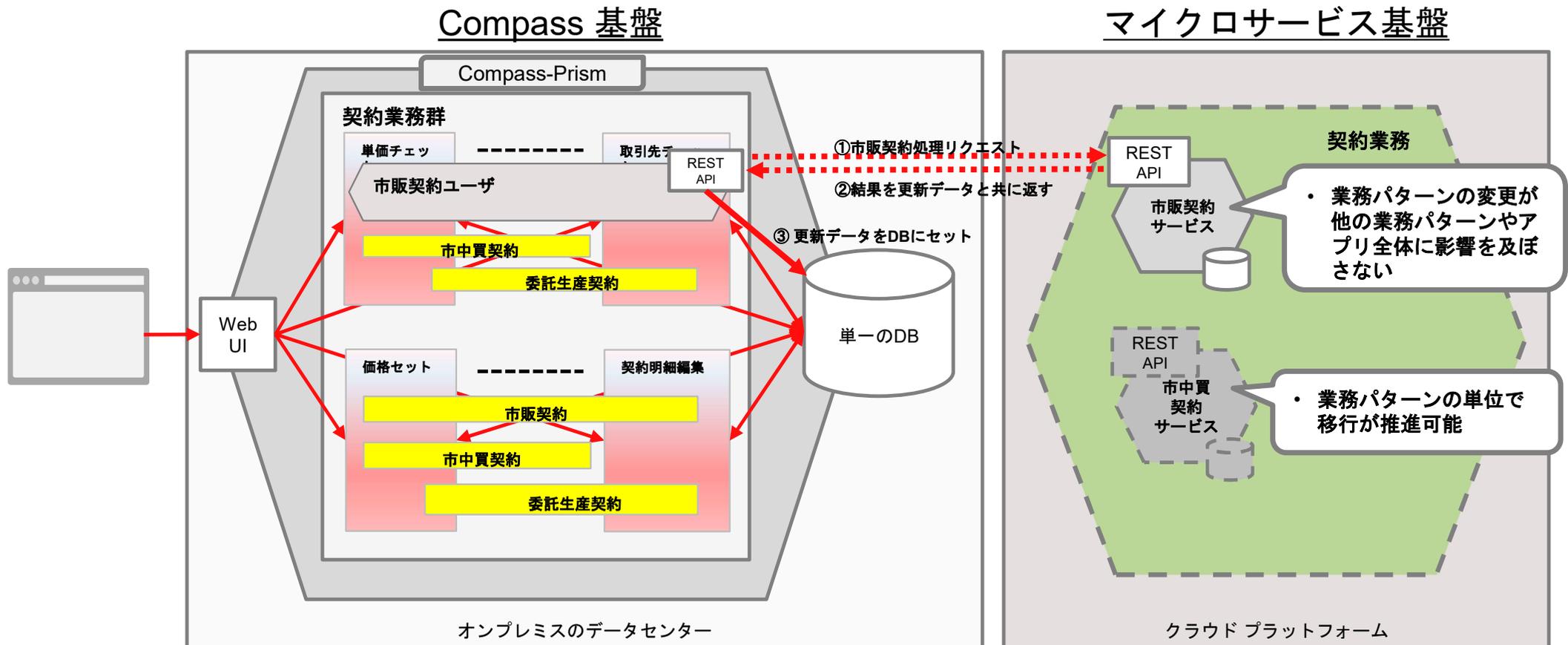


### マイクロサービス基盤



# 今後の予定（DX化 2025年の崖に向けて）

## Compass 基盤とマイクロサービス基盤の併用



UNITIKA LTD.



 技術 ×  発想力

素材で未来をカタ手に。

**UNITIKA**  
We Realize It!

ご清聴ありがとうございました。