メインフレーム同等の信頼性と処理性能を実現した基幹系Webシステム構築事例

（株）日本総研ソリューションズ
産業事業本部　システム開発統括部
流通サービスシステム開発グループ

上野　智史

日本総研ソリューションズ
JRI Solutions, Limited

アジェンダ

1. プロジェクト概要
   - -プロジェクトの目的
   - -求められたシステム要件

2. システム要件を満たすためのアーキテクチャ
   - - 信頼性
   - - 利便性
   - - 開発生産性

3. 総括

0.会社紹介

株式会社　日本総研ソリューションズ

■設立
2006年7月

■資本金
50億円

■従業員
1,300名

■URL
http://www.jri-sol.co.jp

日本総研ソリューションズを語る4つのキーワード。

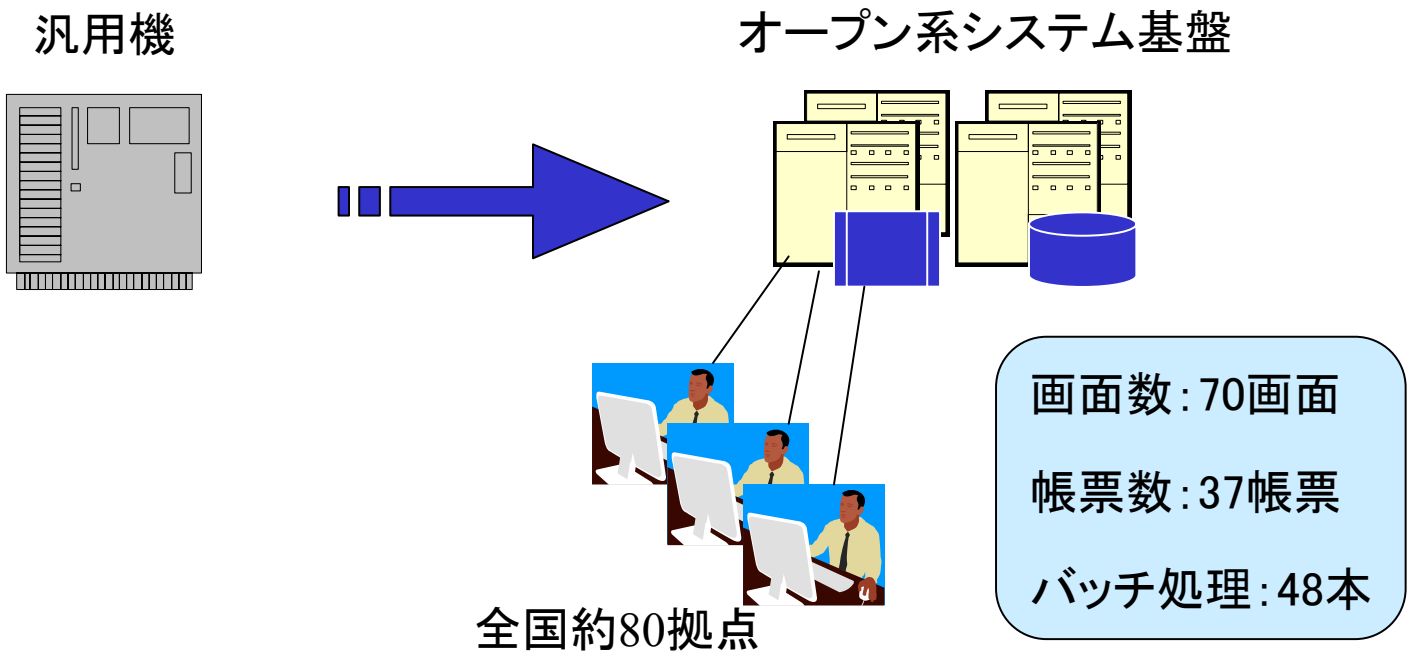
1. お客様との信頼を結ぶ掘るぎないパートナーシップ。
2. シンクタンク、コンサルティングと連携した真の総合力。
3. 最適化された環境を実現する先進のテクノロジー。
4. 中立性とコーディネート力。経験豊かな業務プロフェッショナル。

■沿革
　一般産業界・公共法人のお客様向けに最適なITソリューションを提供する会社として、株式会社日本総合研究所から会社分割により設立

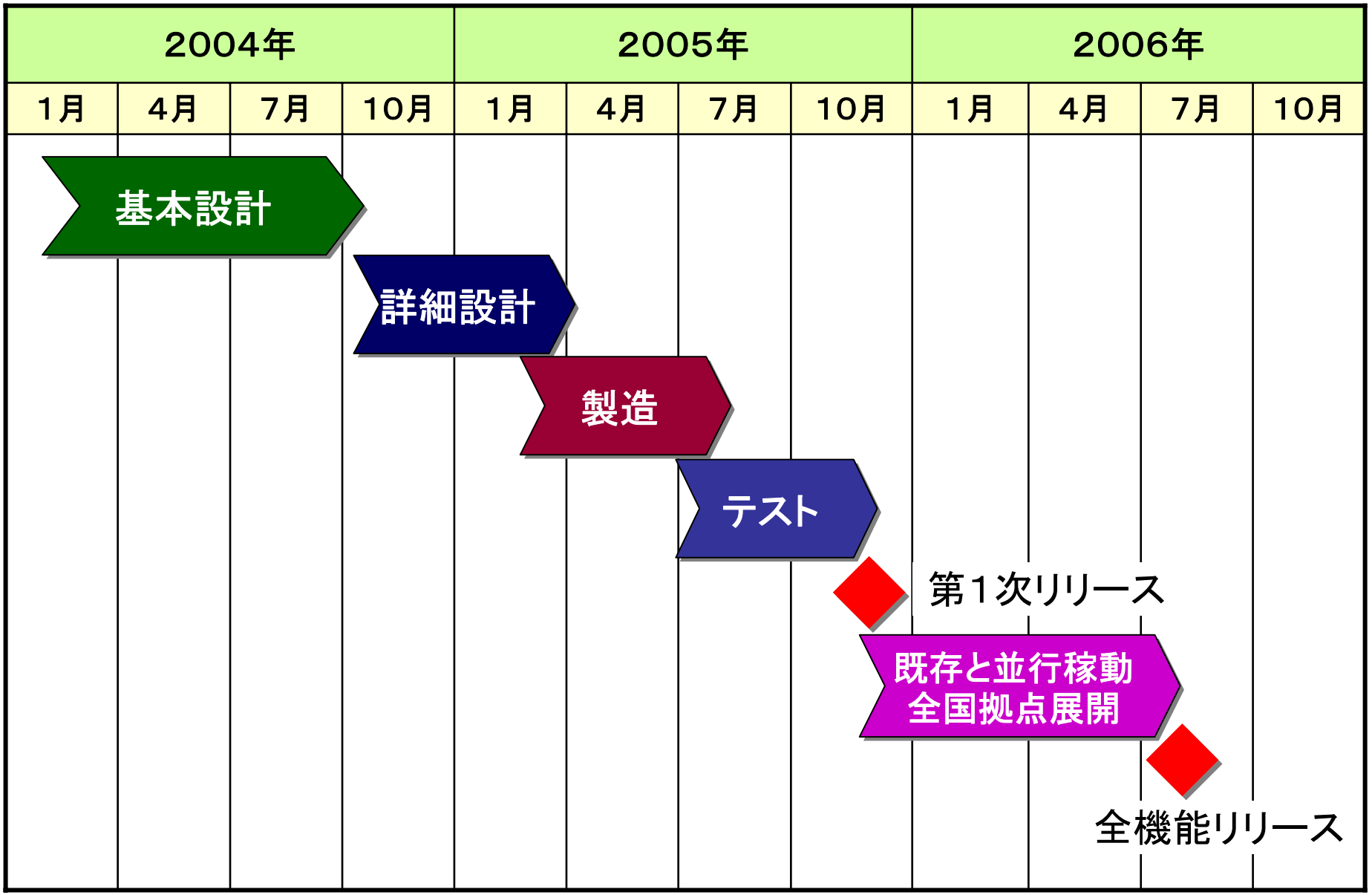
1-1. プロジェクト概要

プロジェクトの目的

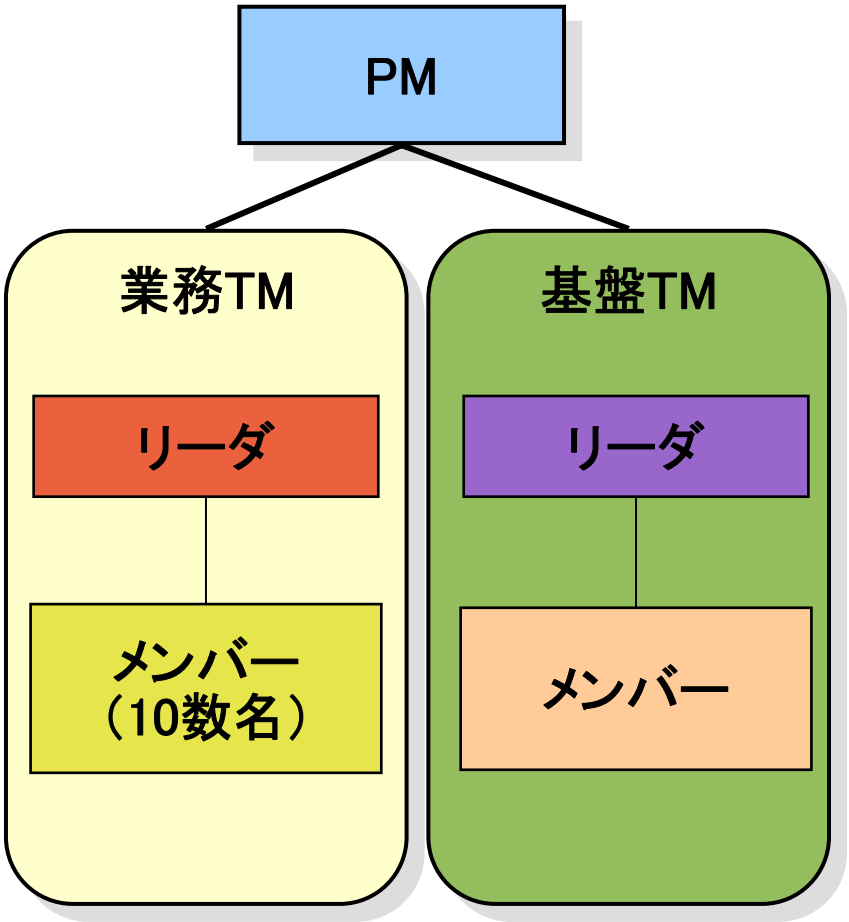
- 汎用機で稼動していた物流システムを、システムの信頼性を落とさずにWebシステムにダウンサイジングする。
- エンドユーザーの画面入力、帳票出力などの利便性の向上
- 開発工数を極力抑え、低コストで開発する



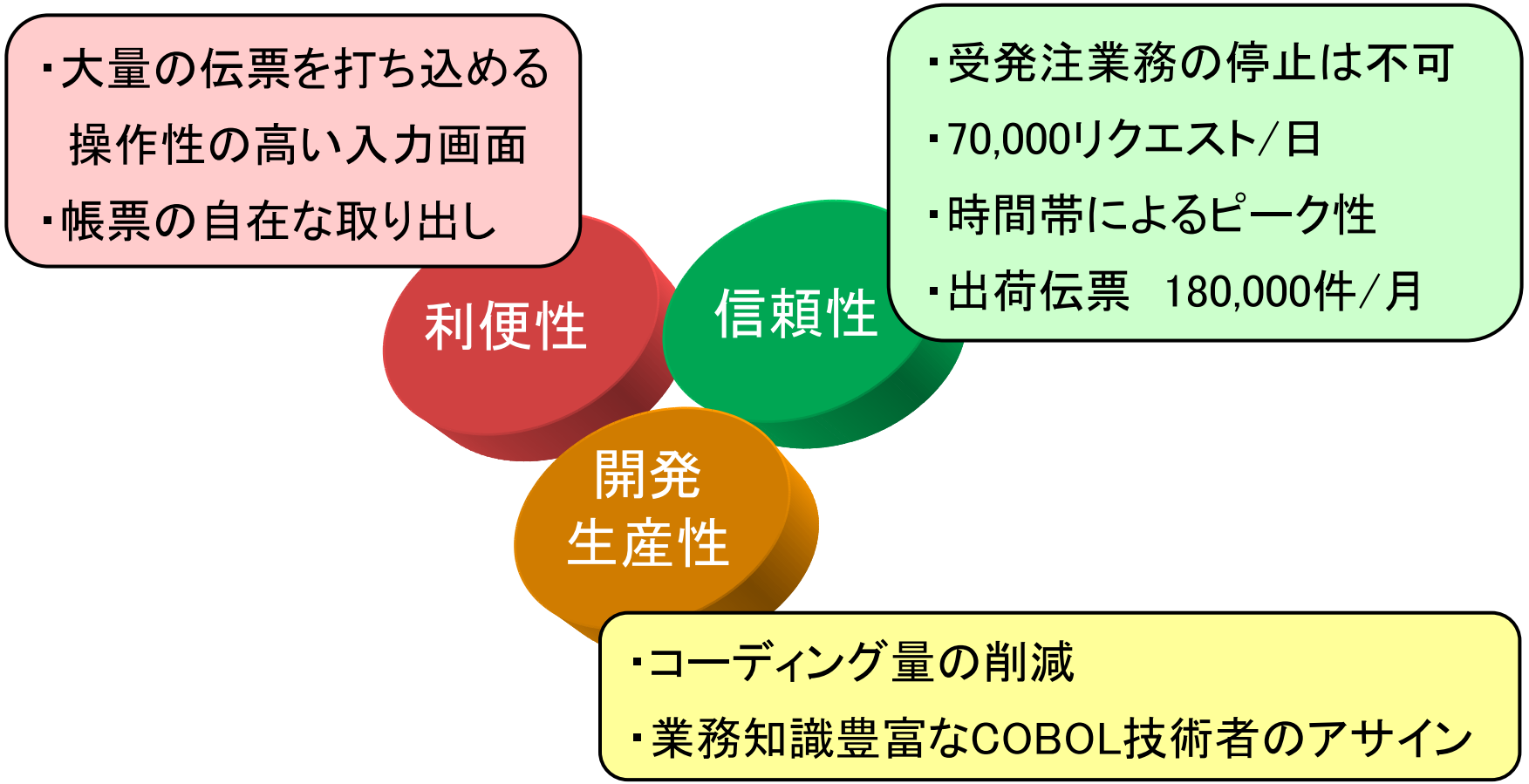
1-2. スケジュール



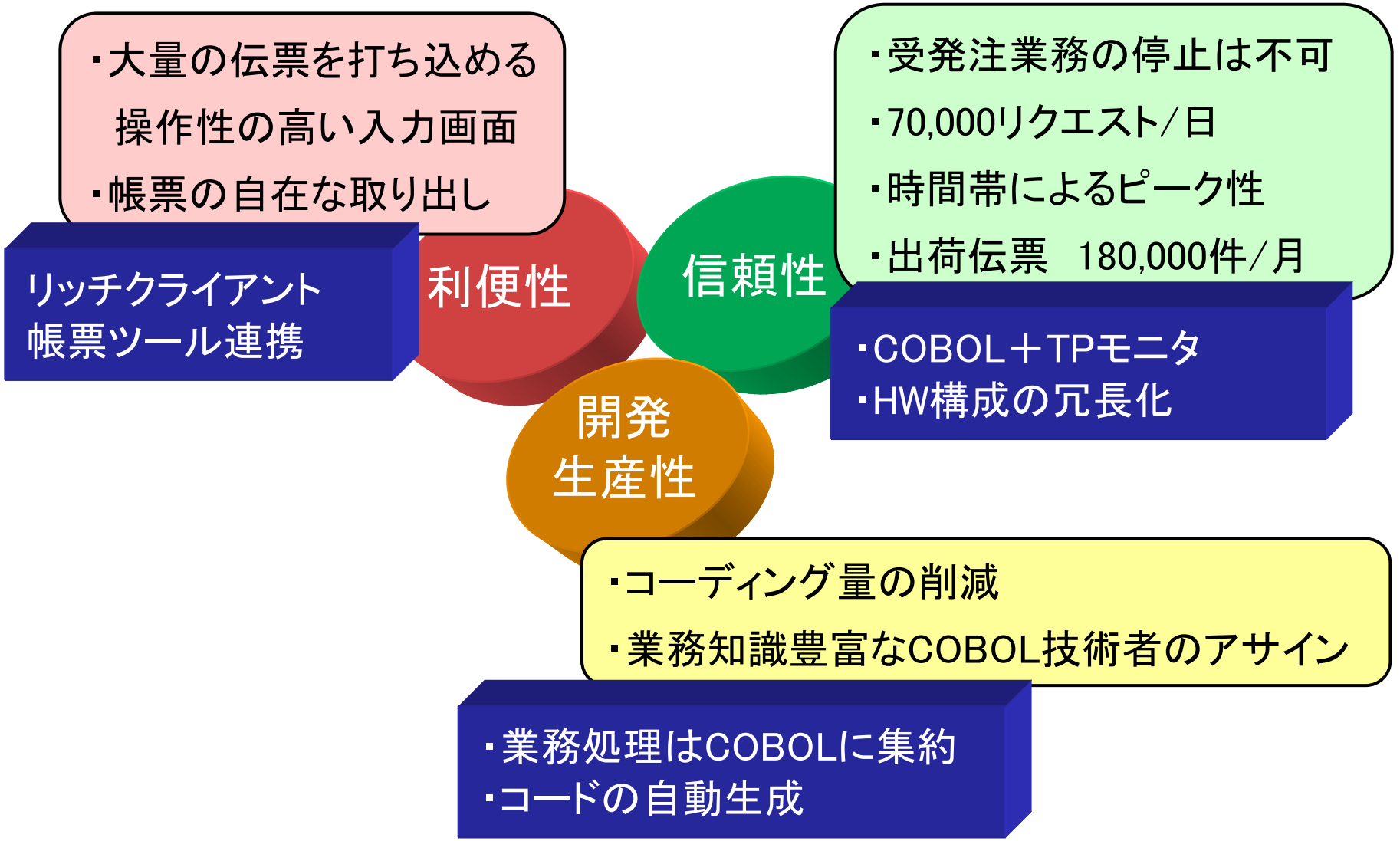
1-3.プロジェクト推進体制



1-4. 物流システムに求められた要件



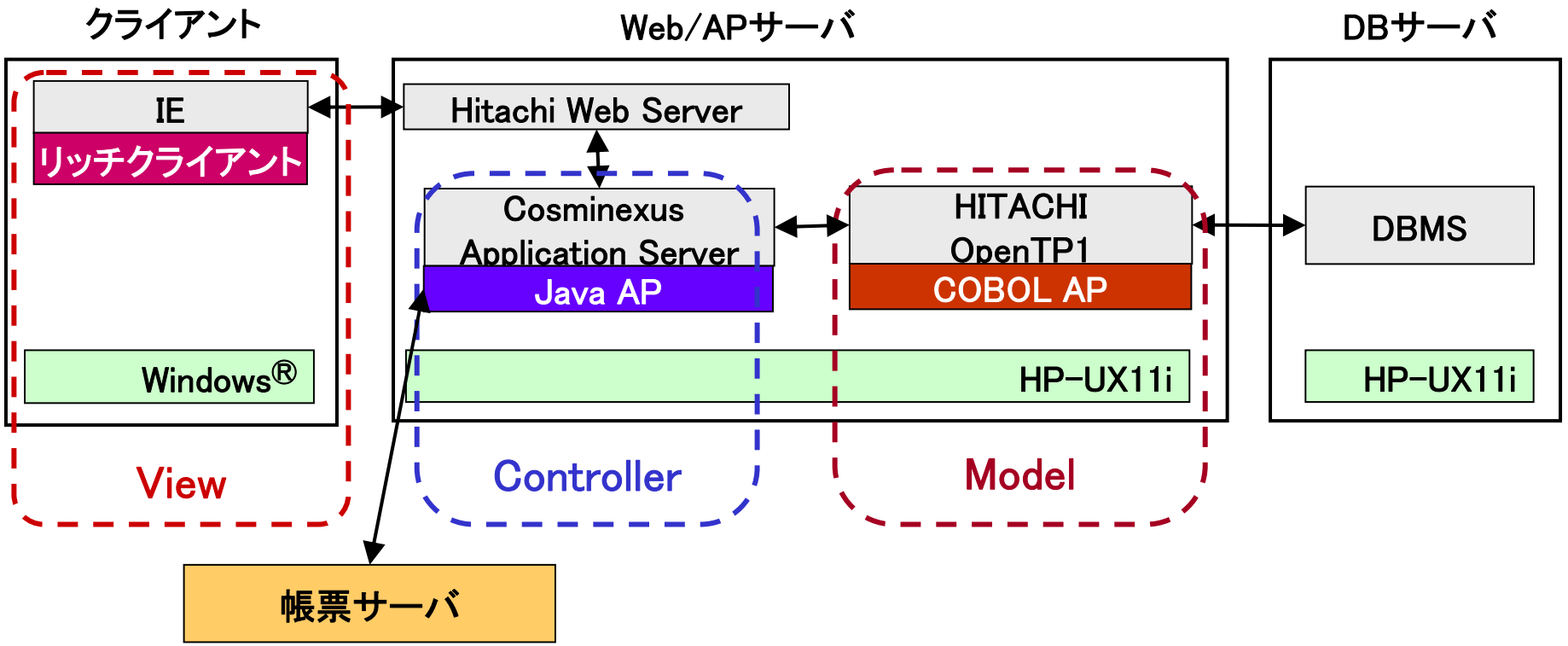
1-5. アーキテクチャの選択



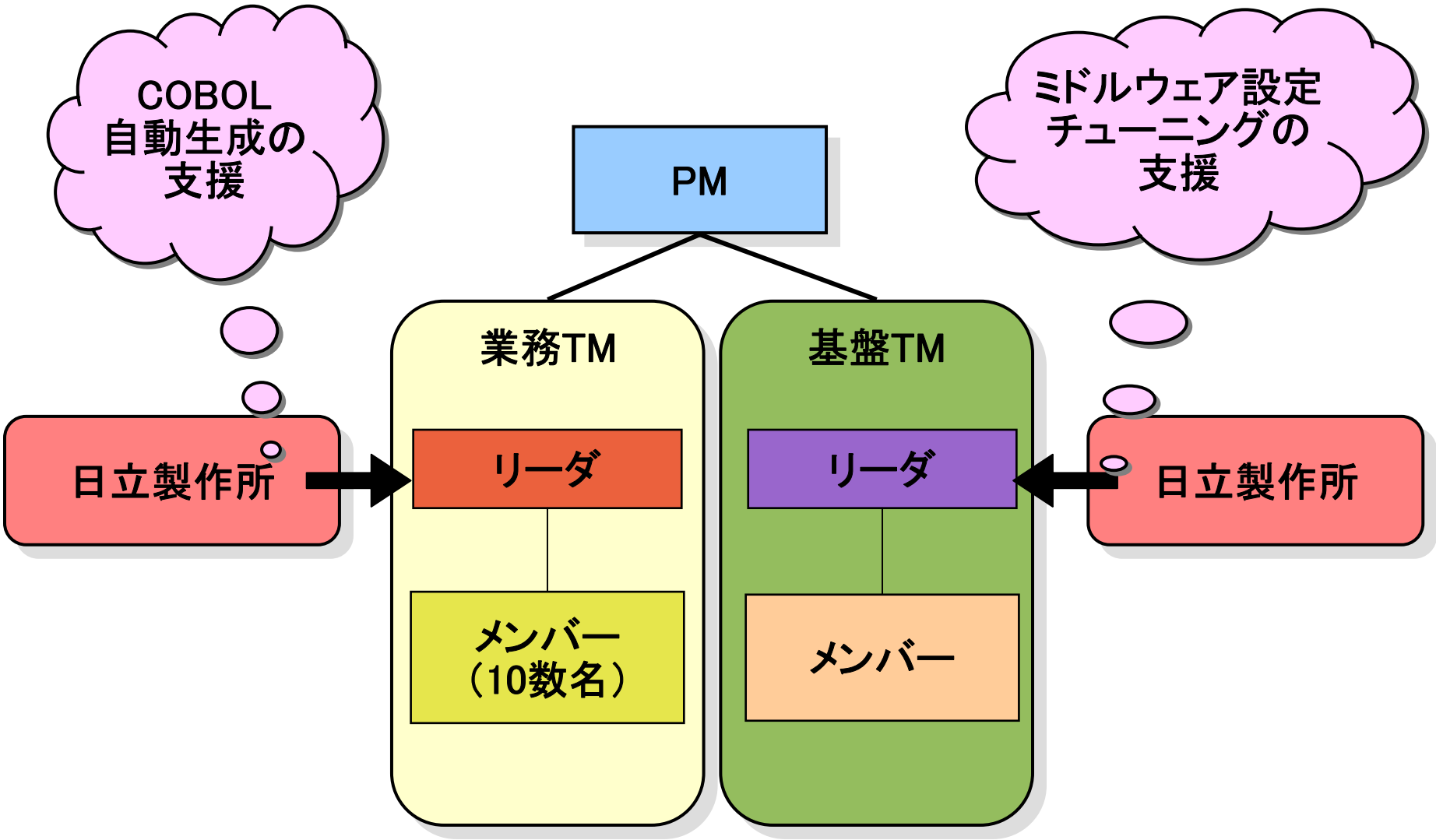
1-6. アーキテクチャ全体像

MVCモデル

- View　　：リッチクライアントで操作性を向上
- Controller　：JavaでCOBOLと画面、帳票ツールの連携
- Model　　：業務ロジック、データ操作は全てCOBOL2002で実装



1-7.日立製作所の技術支援



2-1. TPモニタを採用したメリット

TPモニタ(OpenTP1)による流量制御

・サービス(機能)単位での流量制御・優先制御が可能

受注入力は使用頻度が高く、
レスポンス低下は許されない

受注入力は
初期起動数を多く

アクセス集中時には
起動数を自動拡張

受注入力
受注入力
受注入力
受注入力
発注入力
発注入力
納期照会
在庫照会

HITACHI　OpenTP1

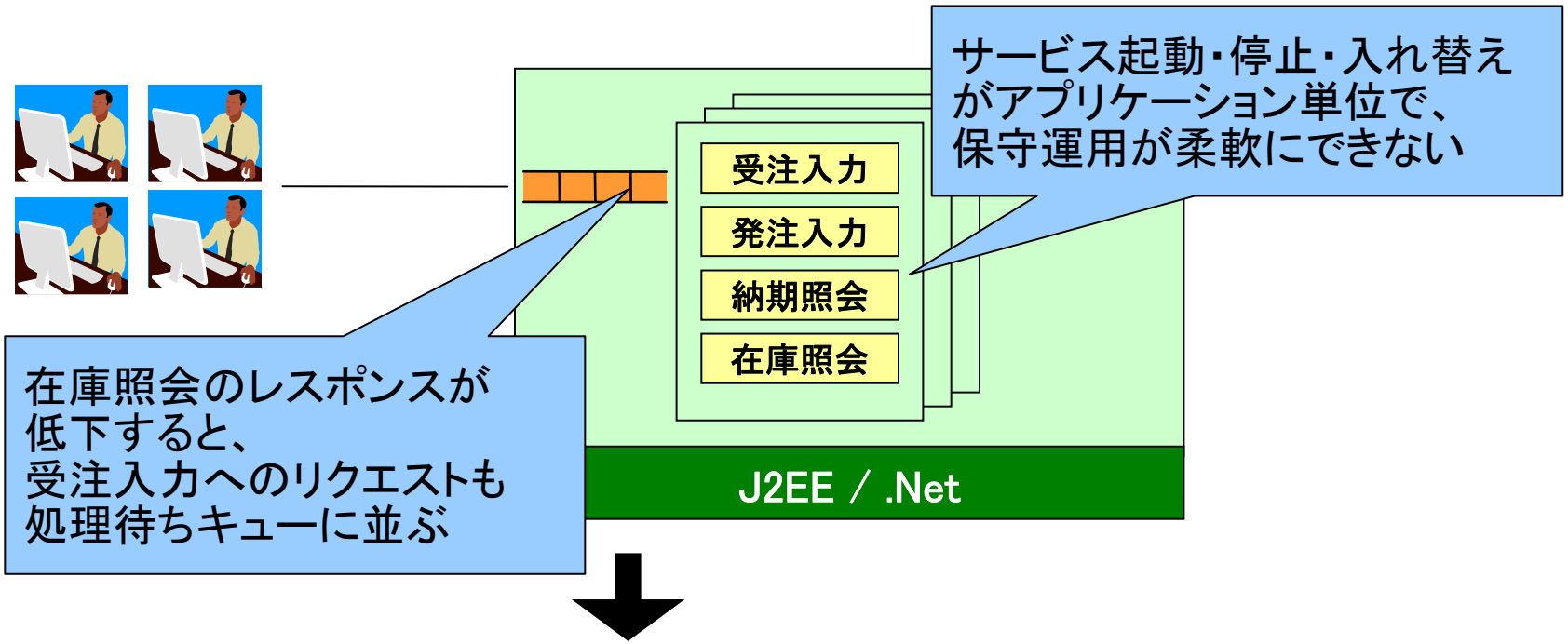
処理待ちキューがサービス単位のため、
他処理のパフォーマンス低下が他に影響
しない

重要処理の
パフォーマンスを
保証する

2-1. TPモニタを採用したメリット

もしTPモニタを採用しなかったら？

・同時実行数の制約が、サービス単位ではなくアプリケーション単位．．．



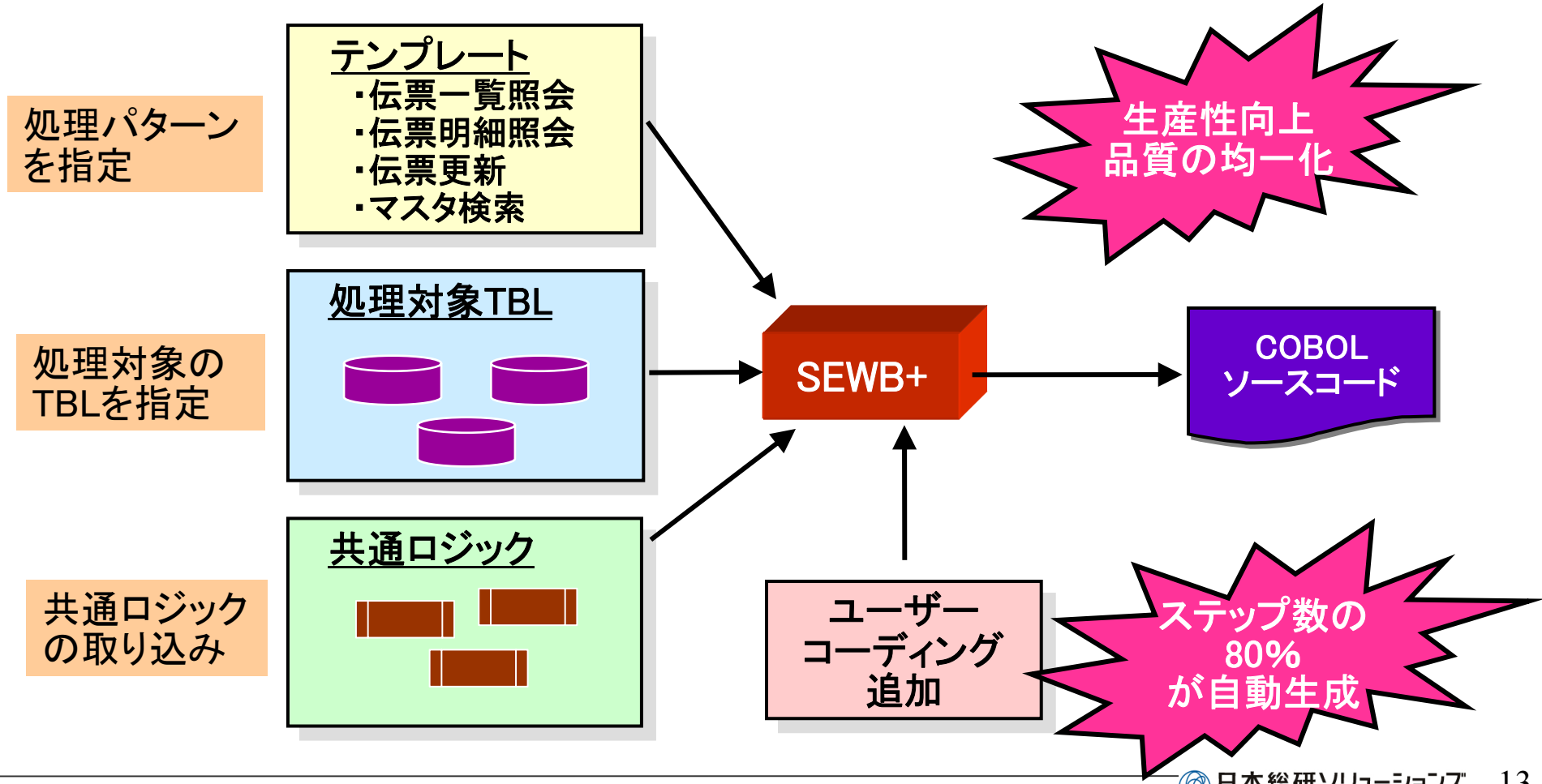
重要性の低い処理のパフォーマンス低下が
重要な処理のパフォーマンスに影響を与える危険性

※信頼性・パフォーマンスの面からJava EJBよりOpenTP1+COBOLを選択

2-2. COBOLコードの自動生成

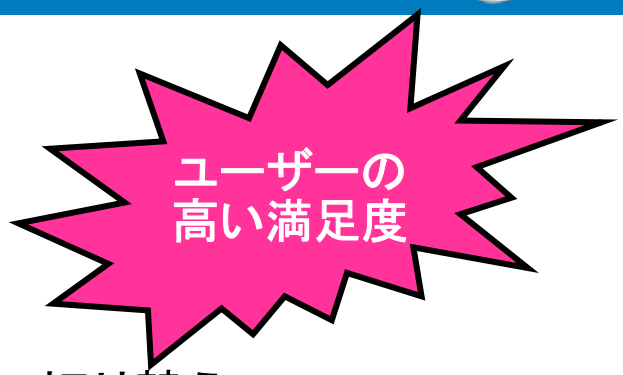
SEWB+による自動生成

- CASEツールとしてSEWB+ を使用
- オンライン画面を4パターンに分類し、各自動生成用テンプレートを作成



2-3. リッチクライアントのメリット

リッチクライアントを採用したメリット



・キーボードからの操作性
　－ ファンクションキーへの機能割り当て
　－数値、半角文字、漢字、日付（カレンダー）入力の切り替え

・通信するデータ量を軽減し、応答時間を短縮できる
　－モジュールのダウンロードは初回のみで、以降はデータのみの通信

・印刷機能
　－サーバーとの通信なし画面の情報を帳票出力

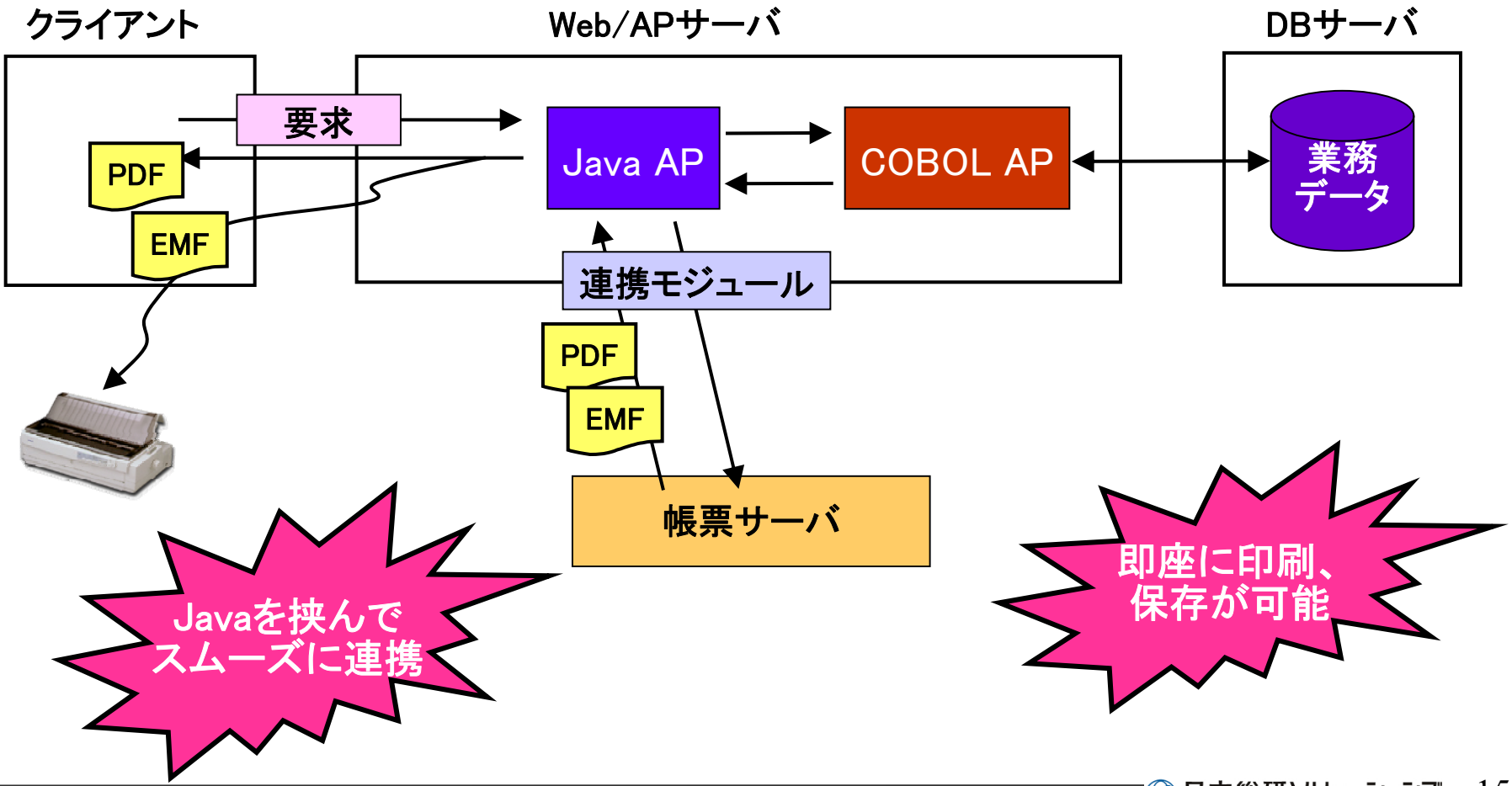
・タブ式の画面切り替えで、多数の情報を一度に表示、入力可能

・開発生産性
　－専用の開発環境があり、GUIツールによる設計、製造、デバッグが可能

2-4. 帳票ツールとの連携（同期処理）

即時性が求められる帳票

⇒同期処理でクライアント側にダウンロード、または直接印刷

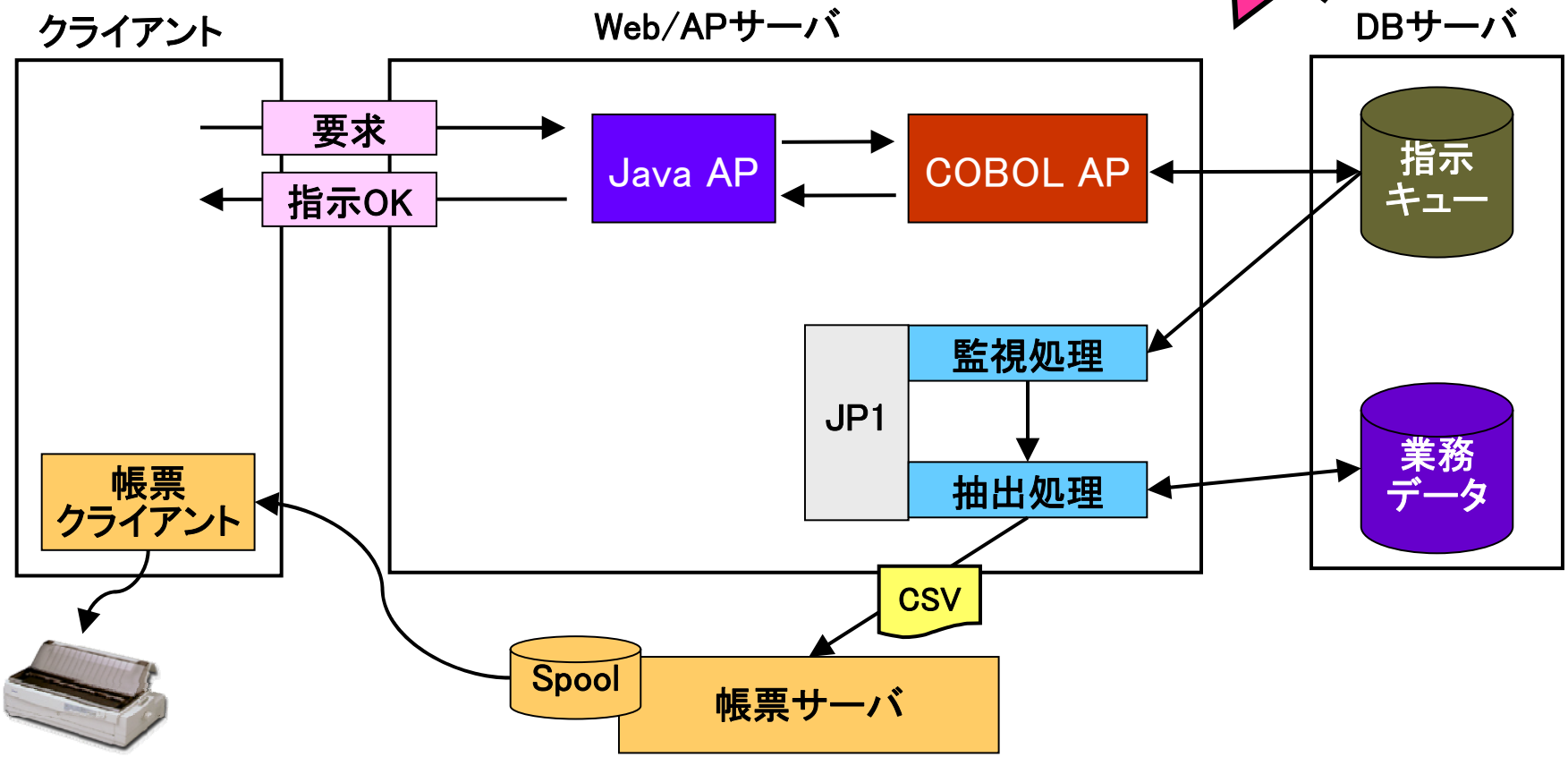


2-5. 帳票ツールとの連携(非同期処理)

即時性が求められない/データ抽出に時間がかかる処理

⇒非同期処理に切り出す

スループットの向上



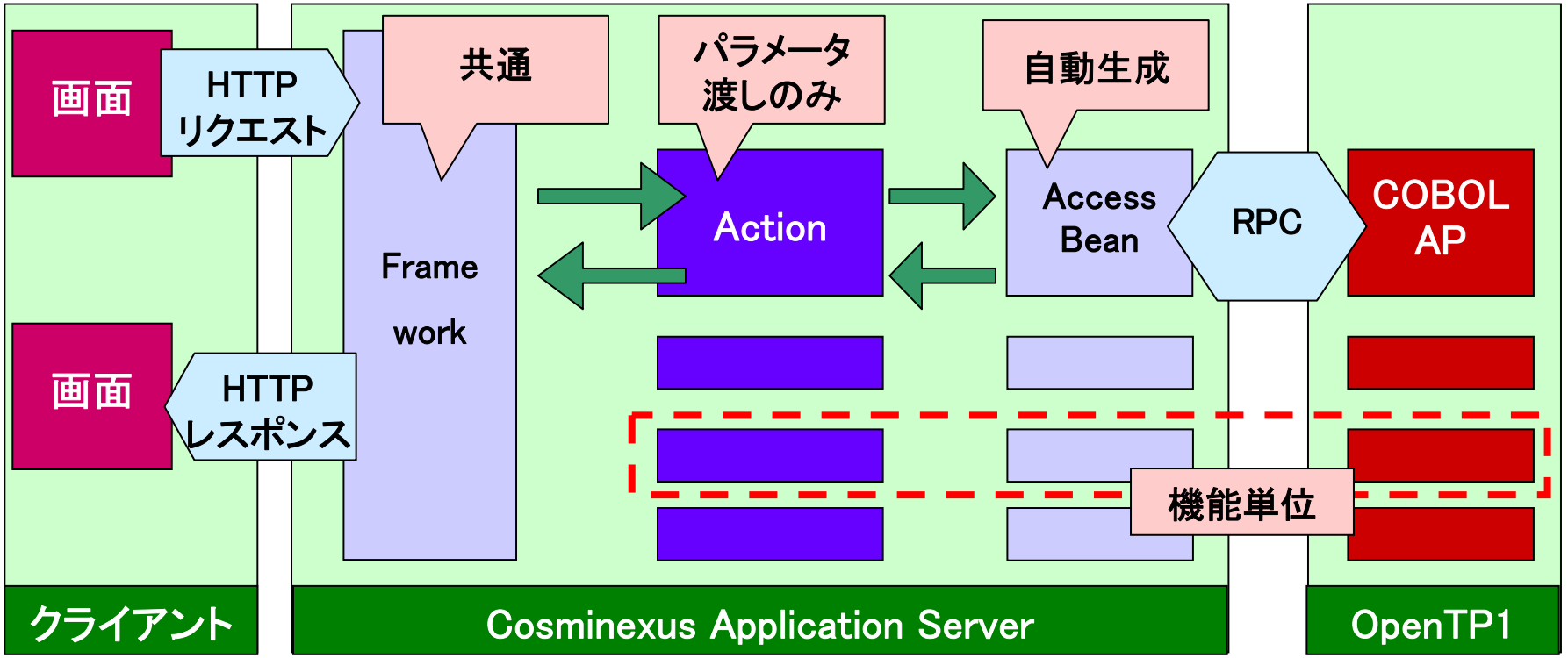
オンライン側のリソース負荷軽減・ユーザーの処理待ち時間軽減

2-6. Java連携部分の実装

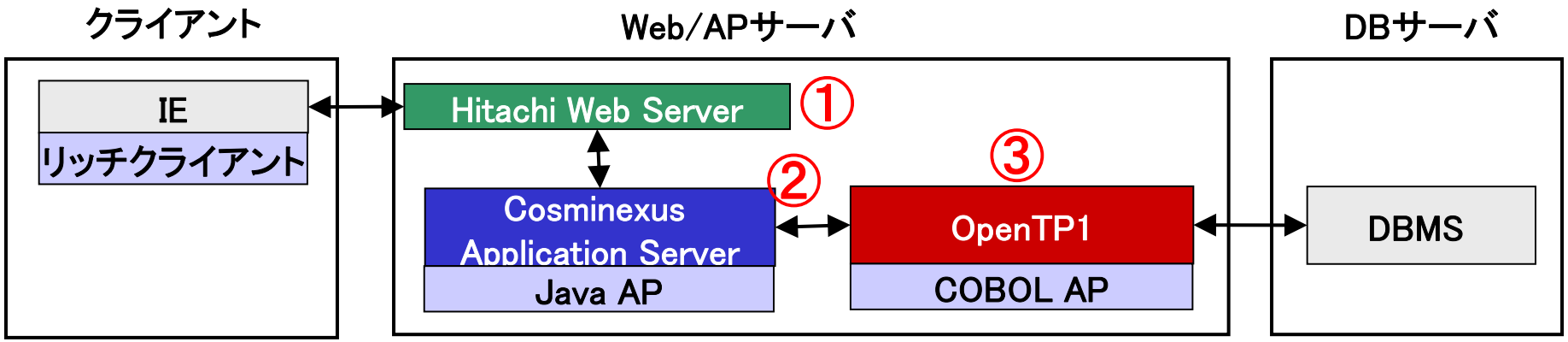
画面毎のJava
開発は殆ど不要

Javaコーディング量の削減

- 独自にFrameworkを構築し共通処理を実装
- COBOLを呼び出すAccessBean は、Cosminexus Studio で自動生成



2-7. システム全体の流量パラメータ

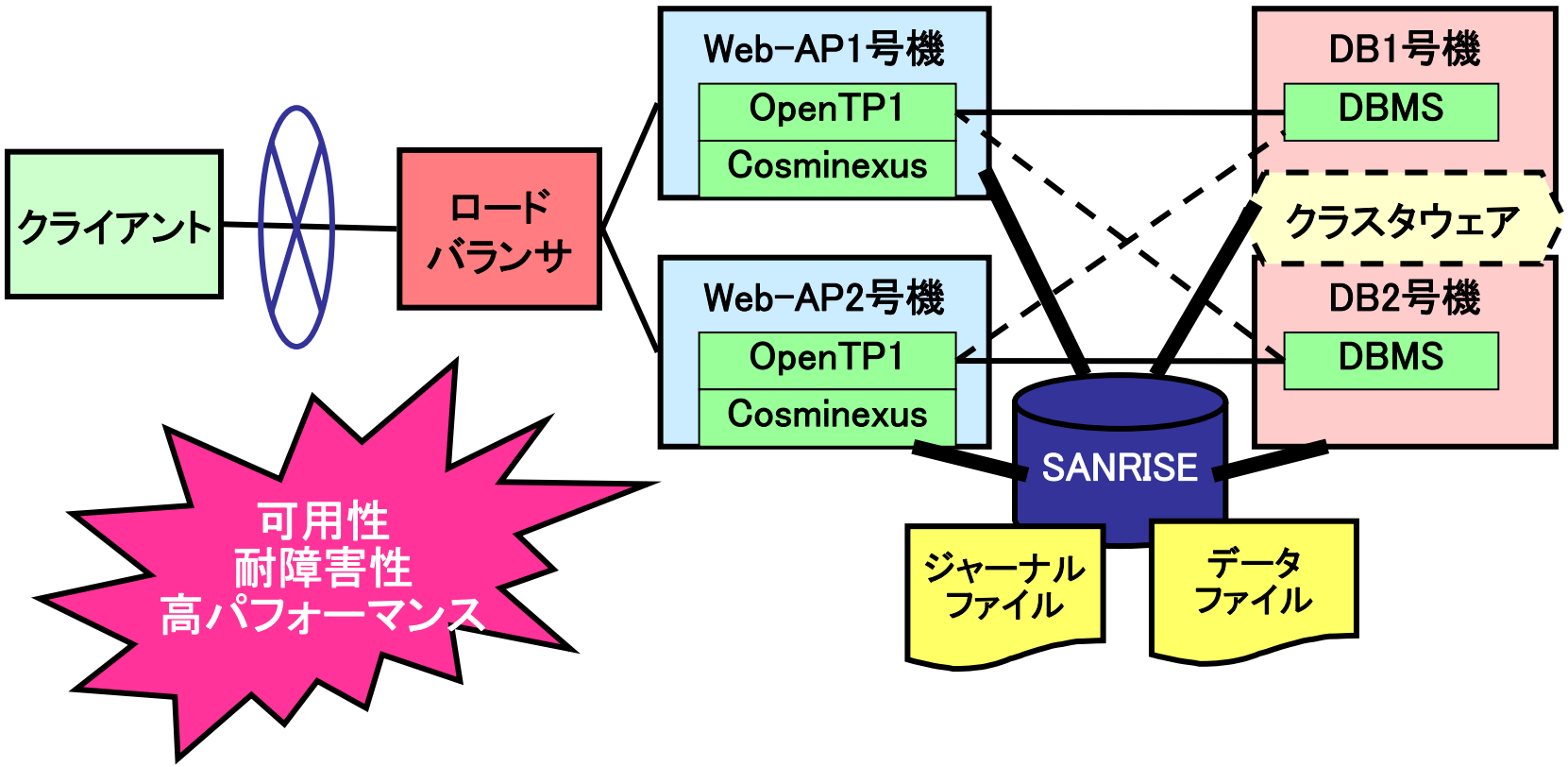


	対象	項目	説明
①	Web Server	HTTPリクエストキュー	クライアントの接続要求待ち行列数の上限
		HTTP処理プロセス数	サーバープロセス数（初期、最大）
②	AP Server	Java Servlet キュー	処理待ち行列数の上限
		Java Servlet 最大スレッド数	APサーバの最大同時実行スレッド数
		TP1コネクションプール	TP1接続のプーリング数（初期、最大）
③	OpenTP1	サーバープロセス数	起動するサーバープロセスの数 （サービス毎、初期、最大）

Webサーバ⇒APサーバ⇒業務処理の順番に流量を絞り込むように注意

2-8. 基盤構成の冗長可

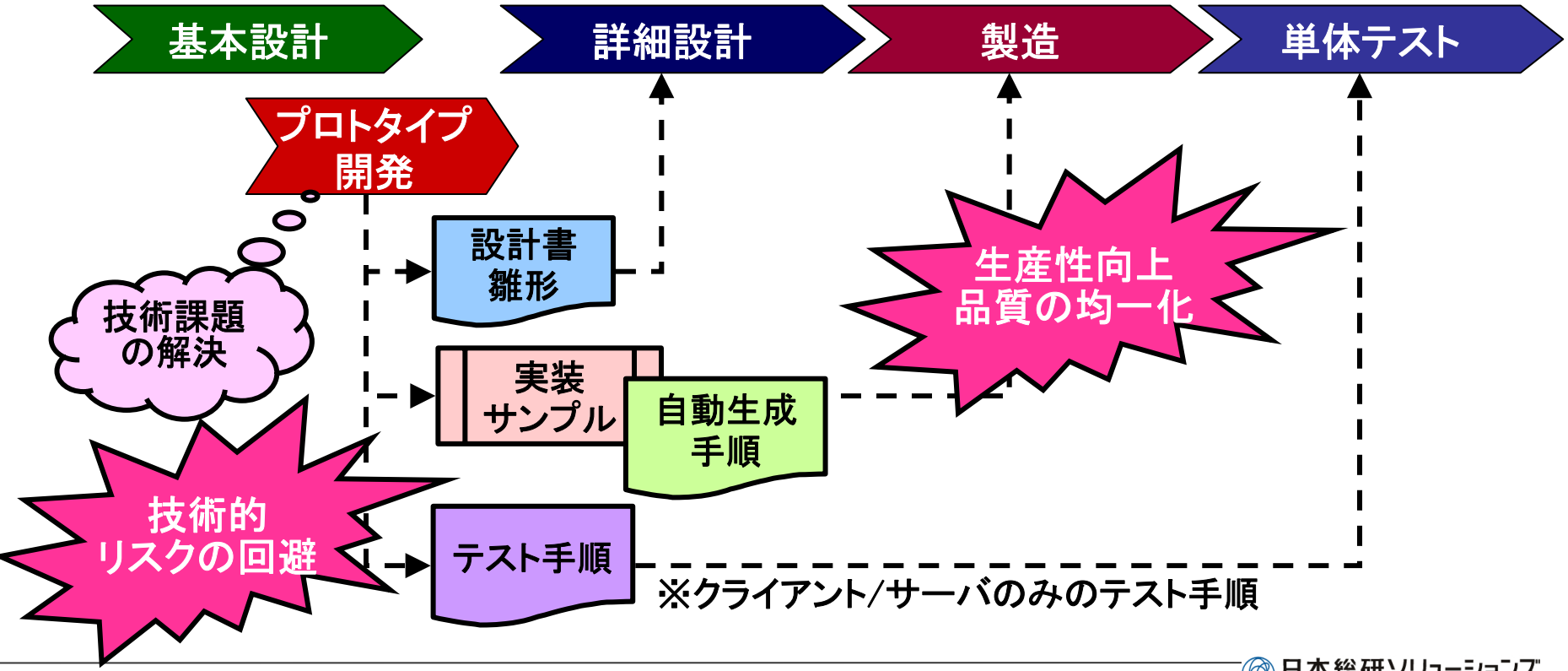
- ロードバランサで2系統に負荷分散
- DBサーバはクラスタリングを実施して可用性を向上
- 重要ファイルはSANRISEに格納し汎用機と同様のI/O性能と耐障害性を実現



2-9. プロトタイプ作成

複数の技術要素を組み合わせたアーキテクチャでシステムを構築する

詳細設計前にプロトタイプ開発を実施しておくことが重要



3. 総括

・OpenTP1とCOBOLを採用し、業務処理を堅牢に構築できた。
・COBOL、Javaのコードを自動生成し、高い開発生産性を実現した
・ロードバランサやDBサーバのクラスタリングによる基盤の冗長化、SANRISEの利用で、耐障害性の強い基盤を構築できた。
・Javaを経由して、COBOLとリッチクライアントや帳票作成ツールの連携をスムーズに実装できたことで、エンドユーザーにとって利便性の高いシステムを構築することができた