CTCにおけるSOAの適用

伊藤忠テクノサイエンス株式会社
情報システム部
小林　雅弘

CRM/WEB 技術部
Web プラットフォーム技術グループ
川田 聡

目次

- 背景
  - 本社移転について
  - ワークスタイルの変革
  - IT の現状と課題
  - Why SOA
- 社内プロジェクト事例
  - スモールスタート
  - 体制
  - プロトタイプイメージと実装手順
  - 実装結果
  - SOA におけるサービスとは
- 反省点とポイント
  - サービス定義
  - Object Oriented Architecture
  - BEA 製品の効果

本社移転について

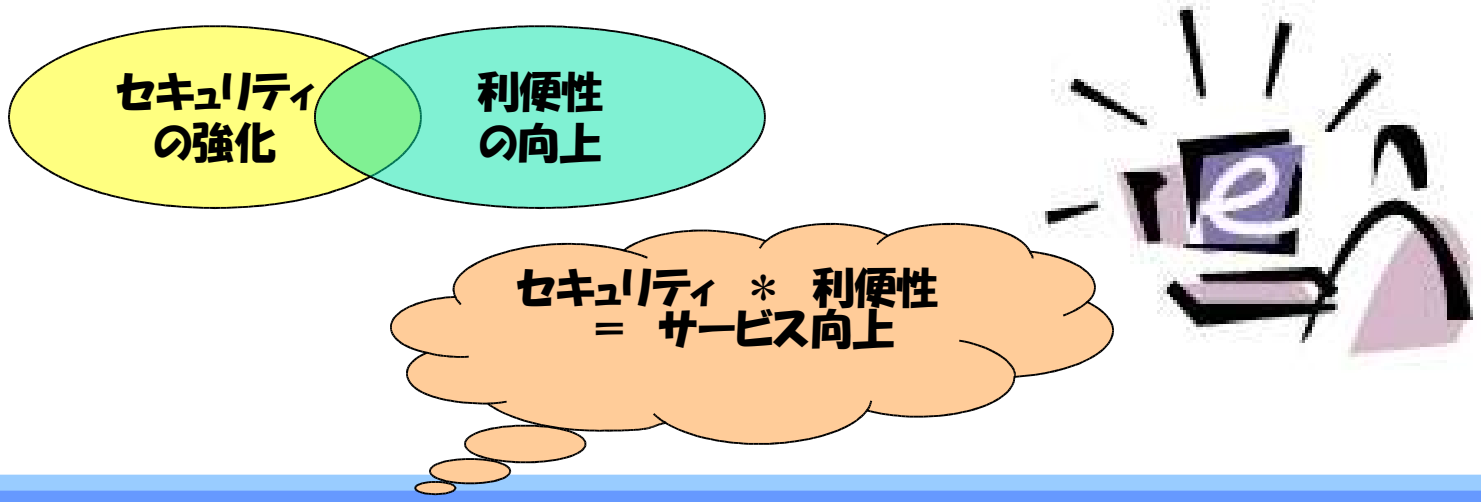
東京地区の主要オフィスを霞が関ビルに統合

- セキュリティを考慮した快適なオフィス空間の構築
  - 万全のセキュリティ
    - ICカードシステムによる入退室管理
  - 一体感の醸成
    - コミュニケーションゾーンの充実
    - コラボレーションゾーンの充実

- 顧客密着型営業の推進
  - 営業とSEを事業グループに集約
    - システム提案から設計・開発、保守サポートまで一貫したサービスを提供
  - 事業グループ内連携および各部署間連携を強化
    - コラボレーションワークの推進

ワークスタイルの変革

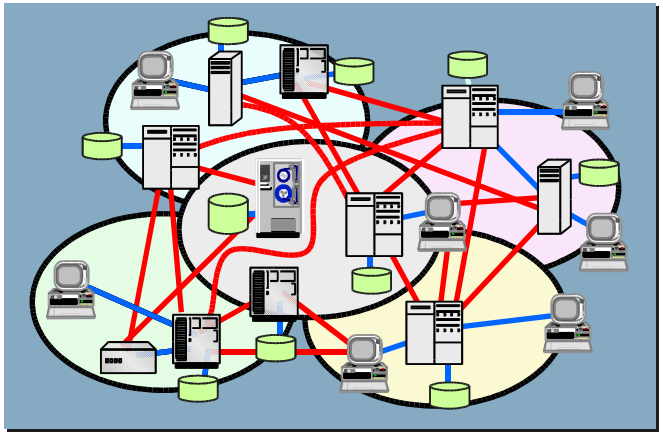
- 新ワークスタイルを **eWork @ CTC** として以下を実現
  - セキュリティの強化
  - 利便性の向上
    - 生産性の向上、ビジネスの効率化、サービスの向上
  - コラボレーションの強化
  - システムのショーケース化



IT の現状と課題

現状

- 企業活動と共に構築されたシステムの肥大化と複雑化
  - 業務プロセスの複雑化
  - 連携システムの増加
  - 個別最適による機能・情報の多重化



課題

個人情報保護の施行
電子取引の活発化
ディスクロージャー対応・・・

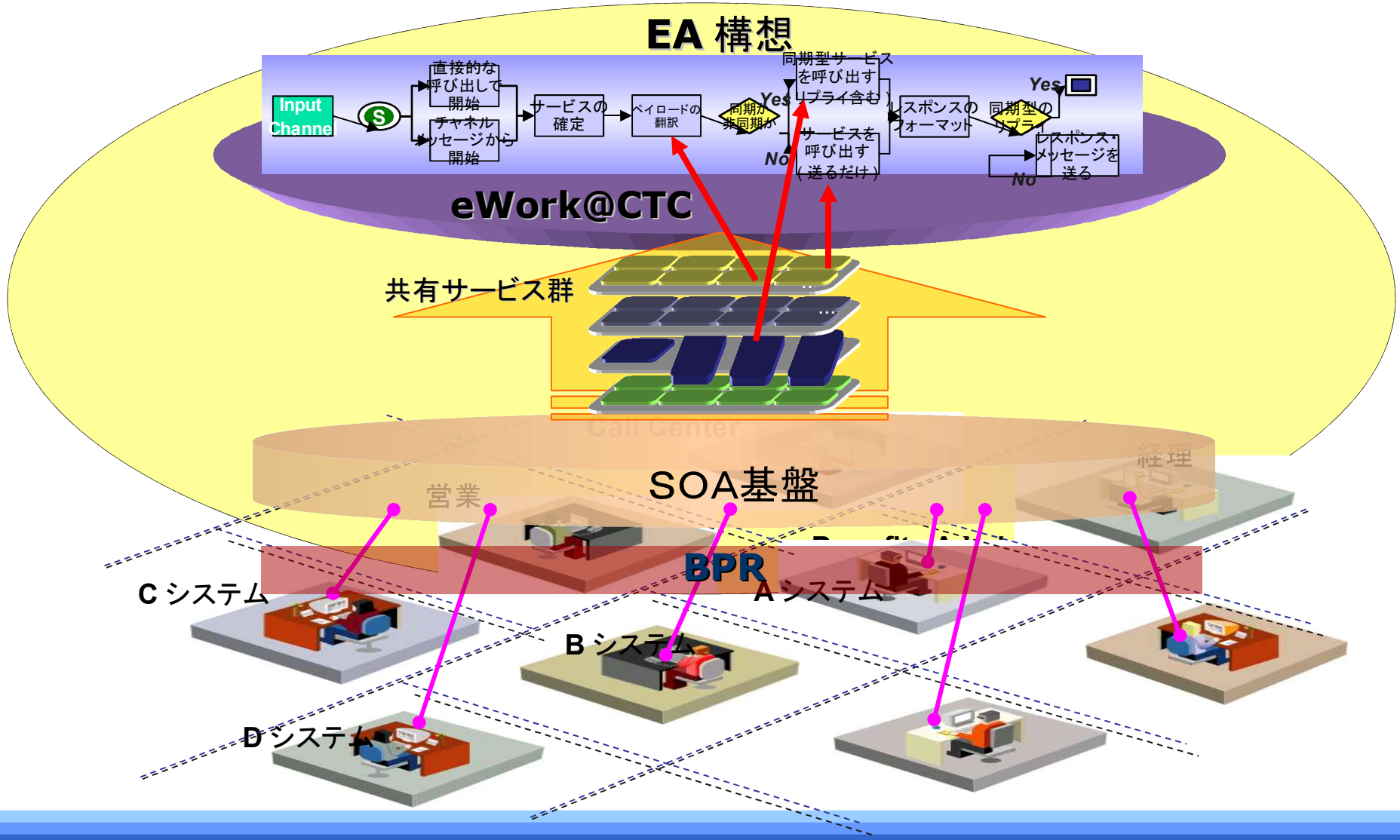
- ビジネス環境の変化に対応するシステムを効果的な投資で迅速に実装可能な仕組みが必要

Why SOA

- 課題解決に期待
  - ビジネスプロセスの可視化
  - システム連携と統合の工数削減
  - 既存システムを改修することなく利用
  - 次期システムの移行パスに活用
  - システム基盤の確立

最終的には生産性の向上とコスト削減を期待

CTC における SOA の位置付け



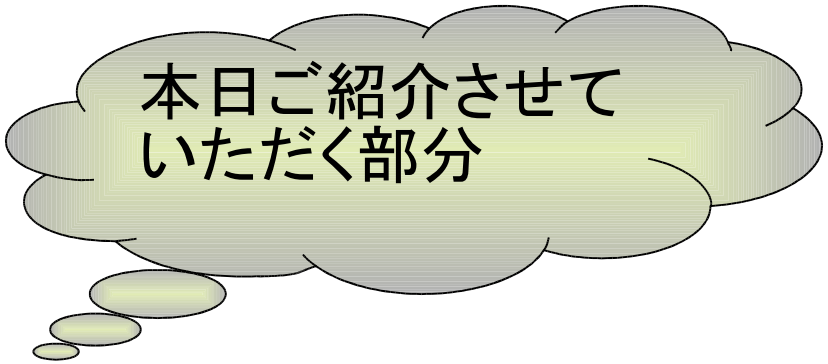
Challenging Tomorrow's Changes

社内プロジェクト紹介

－ プロトタイプ設計と実装 －

Project：スモールスタート

- プロトタイプ作成
  - プロジェクト関係者間におけるSOAに対する共通認識の確立
  - POC（Proof Of Concept）
  - システム拡張を想定したサイジング基礎データの取得
- FSの実施
  - 詳細要件の洗い出し
- 本開発
- 初期サービス開始
- システムレビュー（繰り返し）
  - 運用方法等の再検討
  - 拡張計画の立案

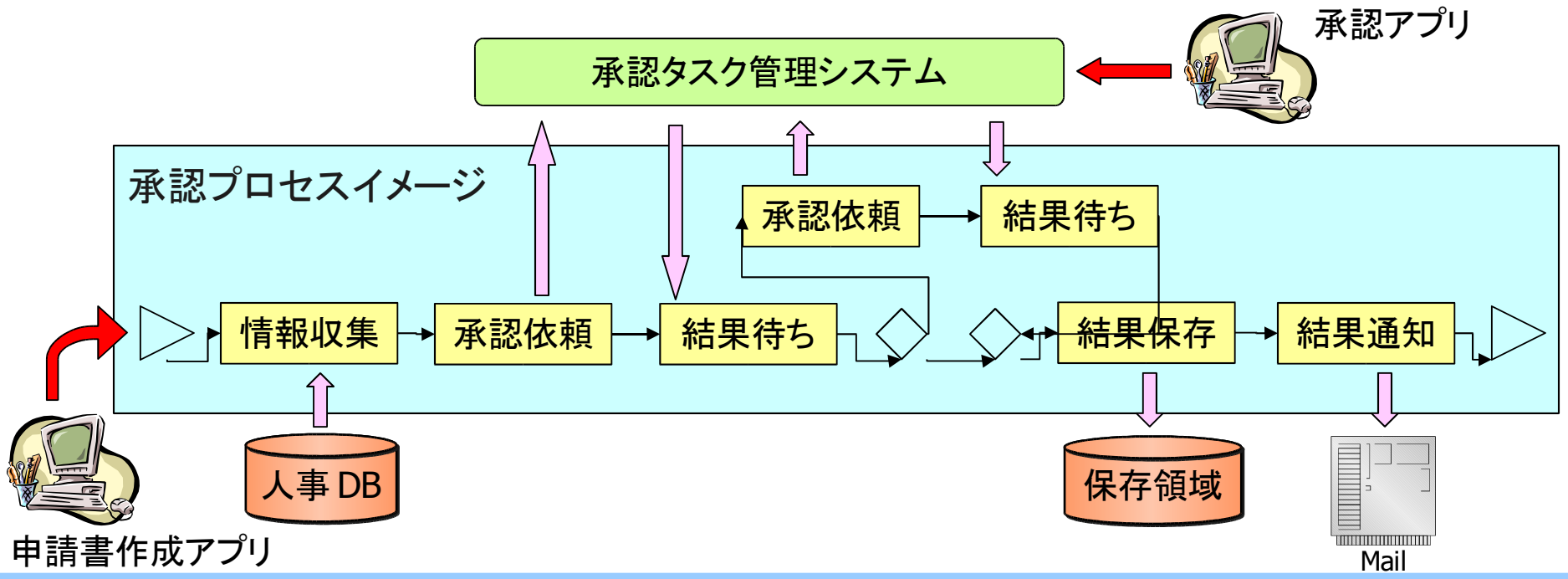


Project：体制

	プロジェクトにおける役割	本来のミッション
情報システム部	■ プロジェクトオーナー ■ ディシジョンメーカ	CTC のシステムを支える部署
CRM/WEB 技術部	■ 製品技術担当 ■ 設計 / 実装担当	製品に対する技術担当部署 ■ 社内 SE からの問い合わせ対応 ■ 製品技術情報の社内への提供
日本 BEA 株式会社	■ スーパーバイザー	製品提供ベンダー

Project：プロトタイプイメージ

- 申請／承認処理の実現
  - 承認業務は各種社内業務の一部になることに加え、そのプロセスより利用するシステムやコンポーネントをサービスとすることができるのでは？



Project：プロトタイプ実装手順

- 業務分析
  - 既存業務を分析 / 整理
    - 共通利用可能な機能の洗い出し
    - 対象業務の軽量化
- サービス候補の抽出
  - 業務分析から行うことによってビジネス的な側面から判断したサービス候補が得られることに期待
- システム要件の抽出
  - ユースケースシナリオ及びユースケース図によって確認
- 基本 / 詳細設計
- 実装

Think：サービス

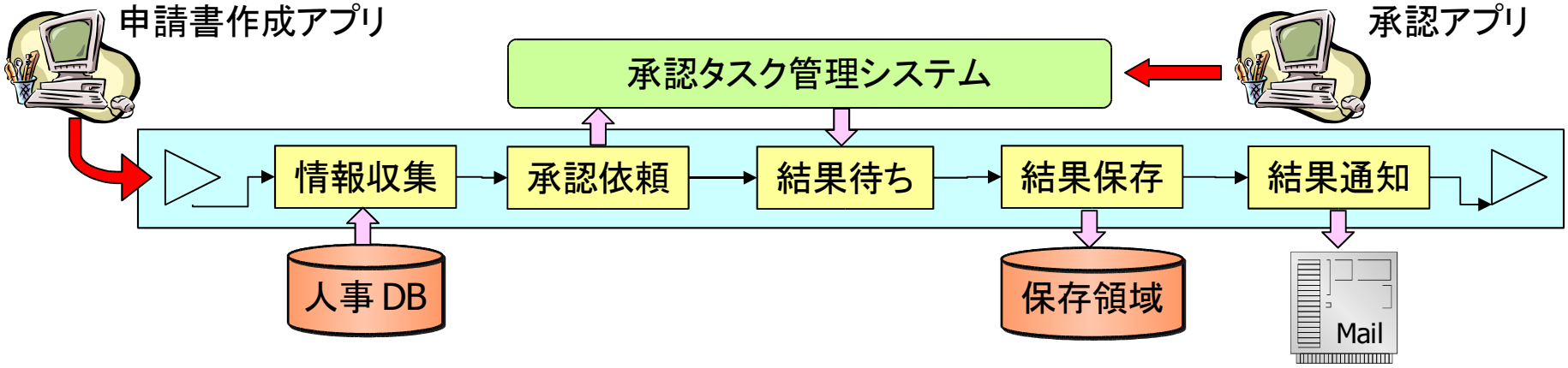
- SOAにおけるサービス
  - 明確に定義されたものはまだない
- サービスとは
  - 奉仕、給仕、接待
  - 商店で値引きしたり、客の便宜を図ったりすること
  - 物質的生産過程以外で機能する労働
- 上記用語から
  - 客＝利用者と考えると
    - ニーズがありそれに対するものがサービス？
    - 人から見ればシステム全体がサービス
    - アプリケーションから見るとコンポーネントがサービス？

Question：サービス

- せっかくSOAに対応させるのであればサービスをたくさん作らなくては！
  - 確かに利用できるサービスが多くなければSOAのメリットを享受できない
    - 変化するビジネスニーズに対応しうる環境が整わない
  - 疑問点がたくさん
    - コンポーネントもサービス？
      - SOAがObject Oriented Architectureと同じになる？
      - コンポーネントと明示的に記載されている文献もある
    - 再利用される機能がサービス？
      - 間違いではないがOOAでも言われていた事？
    - ライブラリー化されたものはコピーして利用
      - ライブラリーを更新した場合に利用するアプリケーション全てを更新？
      - 今までとあまりかわらないのでは？
    - UIを持つものもサービス
      - 人が利用するものであり利用形態が制限されている？

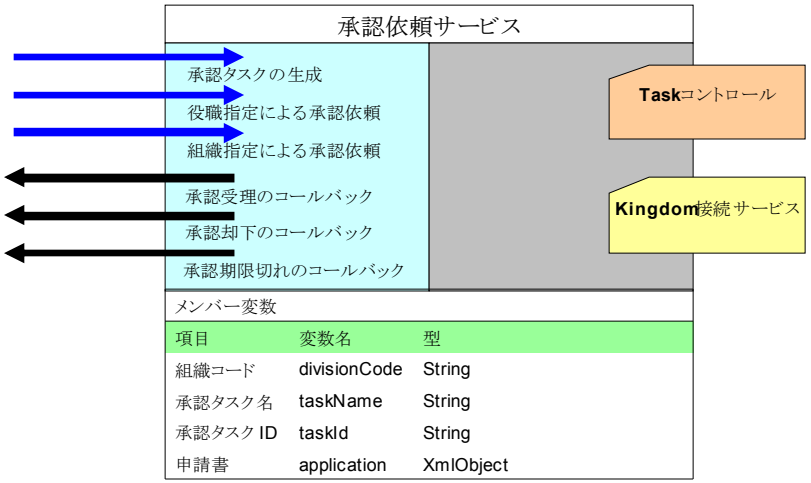
Project：サービス

想定しうるものは全てサービスとして考えることに決定

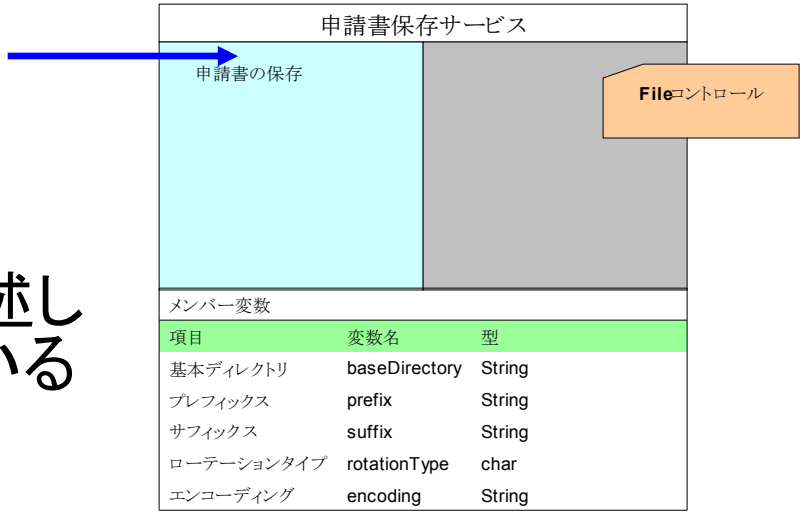


サービス候補	提供機能	実装方法
申請書作成サービス	社員の申請書作成用いる Web アプリケーション	Web アプリケーション
承認依頼サービス	承認タスク管理システムとのインターフェース	コンポーネント（コントロール）
承認作業サービス	承認 / 却下などを行う Web アプリケーション	Web アプリケーション
個人情報取得サービス	人事データベースに相当する社員情報を保持したデータベースより、社員に属する情報を抽出し XML データとして提供	コンポーネント（コントロール）
申請書保存サービス	申請書が紙ではなくなるため、最終的な記録として申請書データ（ XML ）をファイルシステムへ保存	コンポーネント（コントロール）
Email 通知サービス	システムからの通知を SMTP によって実現	コンポーネント（コントロール）

Project :コントロールの実装

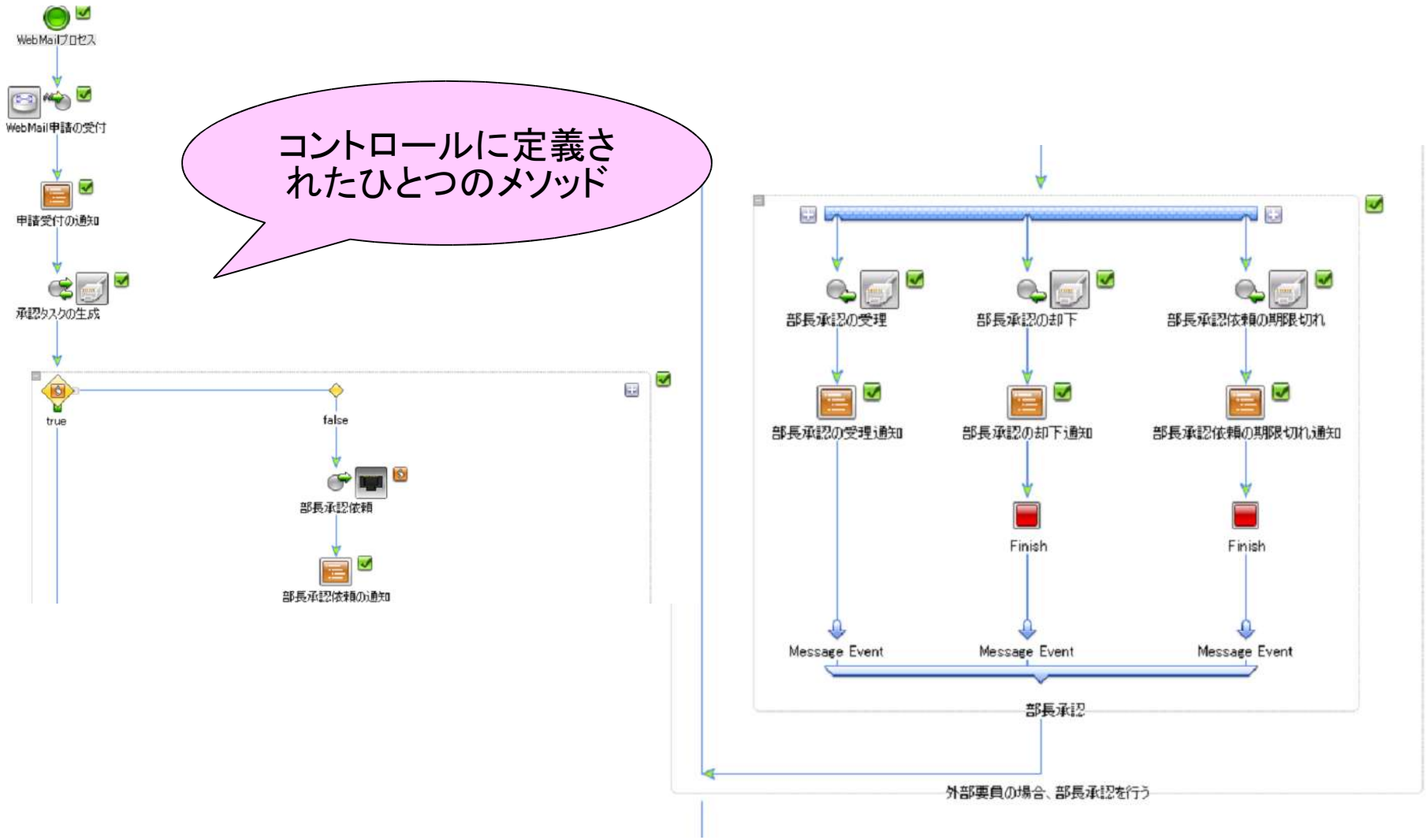


- コントロール
  - BEA が提供するコンポーネント化テクノロジー
  - 利用時にはコントロールをインスタンス変数として扱う



- 実装
  - 各メソッド内にロジックを記述し、必用に応じて提供されているコントロールを利用

Project：実装結果

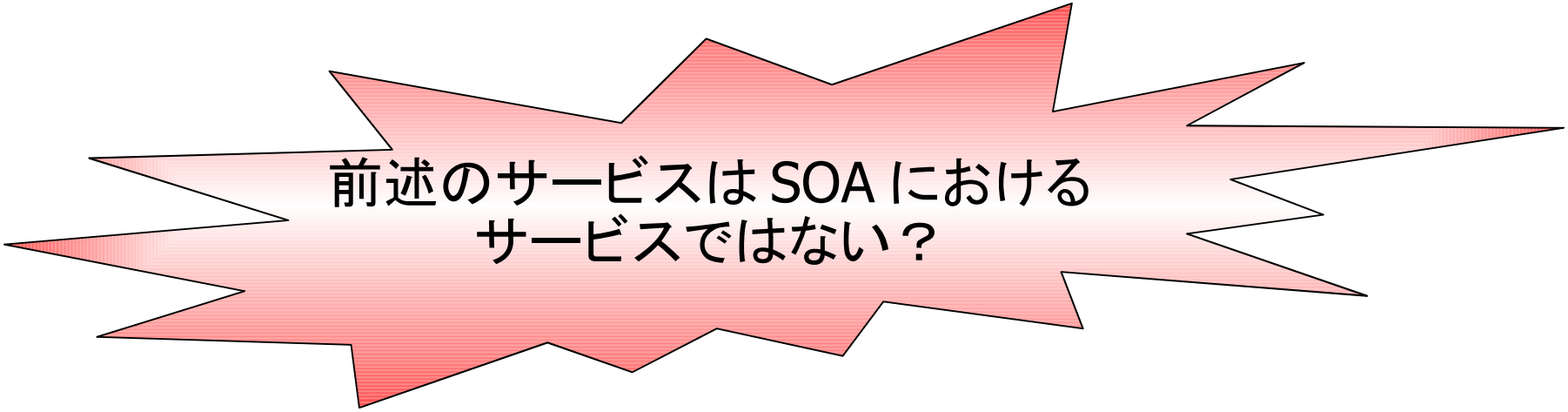


Re-Think：SOA におけるサービス

- 従来のコンポーネントと比較すると粒度が粗い
  - 企業全体のシステムを管理するにあたり、細かな実装を把握するのではなく、より大きな視点での管理を実現
- ネットワークを越えて異種アーキテクチャー上のアプリケーションから呼び出される
  - 再利用されないものはサービスでなくてよい
  - UI でしか利用できないものはサービスではない
- サービス利用者と提供者の関係は疎結合
  - 実装がインターフェースによって隠蔽される
  - 実装の変更が利用者に影響しない
- コンピュータアーキテクチャーに依存しないプロトコルを用いる
  - スタンダード → ベンダーロックインを回避

Re-Think：本当にサービス？

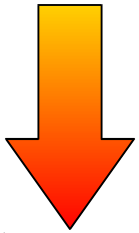
サービス候補	サービスであるのか否か
申請書作成サービス	単なるアプリケーションであり UI を用いるため SOA サービスではない
承認依頼サービス	コントロールとして実装しているため現状ではサービスではない
承認作業サービス	単なるアプリケーションであり UI を用いるため SOA サービスではない
個人情報取得サービス	コントロールとして実装しているため現状では SOA サービスではない
申請書保存サービス	コントロールとして実装しているため現状では SOA サービスではない
Email 通知サービス	コントロールとして実装しているため現状では SOA サービスではない
承認サービス	プロセス全体を SOA サービスとすることは考えられる



反省点とポイント

Important：サービスの定義が大切

- SOA サービスの定義がずれていても開発者のスキルがあれば稼動するシステムを構築することは可能



- SOA サービスに対する明確な定義がなされていないため、最低限の指針以外の部分については、自社にて定義する必要あり
  - 固定的なものを最初から定めるのではなく、逐次必要に応じて変更を行うべく、運用を考えておく必要あり

技術的指針

- サービスの粒度
  - 構成要素の数（あくまでも指標）
- サービスのレイヤー
  - サービスの構成要素
- サービスの利用形態
  - リモートから用いられる
    - パフォーマンスを捨てる価値があるかどうか？
      - 柔軟性（再利用も考慮）
- インターフェースのタイプ
  - UI だけではサービスになりえない

サービス粒度

- サービスコンポジション
  - サービスを組み合わせて作成したサービス
- サービス
- コンポーネント
  - 分散コンポーネント
  - 分散オブジェクト
- 実装言語オブジェクト
  - POJO 等

各項目間の閾値などの定義も有効

サービスレイヤー

- サービスの数が多くなってきた場合に必要になると思われる
  - サービスオーケストレーション
    - ビジネスプロセス
  - サービス
    - ビジネストランザクション
  - 共有サービス
    - 機能サービス

より上位のものから下位のものが呼び出される

非技術系指針

- ビジネス面での基準
  - 全社的な視点からみた定義
    - 各企業毎の定義が非常に大切
- 例外条件
  - 例外時の判断基準等
    - 実運用では定型的な基準だけでは決定できないことも多くあることが考えられる

Important：Object Oriented Architecture

- OOA は SOA サービスへの準備
  - そもそも特定の機能を提供する単位にてコンポーネント化されていることになるため、Wrap することによって SOA サービスとする考え方ができる
    - 既存システムやレガシーアプリケーションに対する考え方も同様

新規開発のシステム / アプリケーションに
SOA サービスとなることを期待する場合
には、OOA を必ず採用すること

BEA 製品の効果

- SOA 準拠のビジネスプロセスを描くことができる
  - BPM 機能によってサービスを組み合わせてビジネスプロセスを描くことが可能
- アプリケーションを構築する場合
  - BPM 機能から呼び出すものをサービスではなく、コンポーネントとすることによって、コンポーネント化されたアプリケーションの構築が行い易い
    - コンポーネントそのものについては Object Oriented Architecture を意識する必要がある
- コンポーネントからサービスへの変更が容易
  - コントロールで作成してあることが条件

まとめ

- まとめ
  - SOA の適用はコンセプトの理解が重要
    - POC の実施を強く推奨
  - サービスの指針と自社環境を踏まえ自社の定義が不可欠
  - SOA そのものの運用が不可欠
    - 定義の変化に対応
  - Object Oriented Architecture を強く意識
- 課題
  - 定義の明文 / 図示化
  - 運用ルールの意識
  - デザインガイドの作成