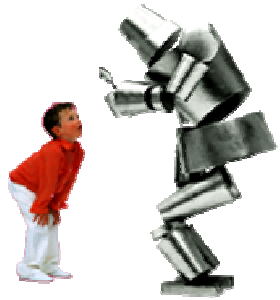
OMRON
Sensing tomorrow™

ソフトウエアアーキテクチャ分析の実践

-ソフトウエア部品化に向けて-

2009年 10月6日
オムロン株式会社
公共ソリューション事業部
小田利彦



発表の構成

OMRON
Sensing tomorrow™

1. はじめに
    - 事業の紹介
    - ソフトウエア部品化とソフトウエアアーキテクチャ
2. アーキテクチャ共通化の事例
    - 鉄道の運賃計算アプリケーション
    - ドメイン分析とアーキテクチャ設計
    - アーキテクチャの管理
    - 製品開発後の成果
3. アーキテクチャ分析
    - 既存ソフトウエアに対するアーキテクチャ分析
    - アーキテクチャ分析の目的
    - アーキテクチャ分析の構成と実施ステップ
    - 実施結果
    - 考察

最後に

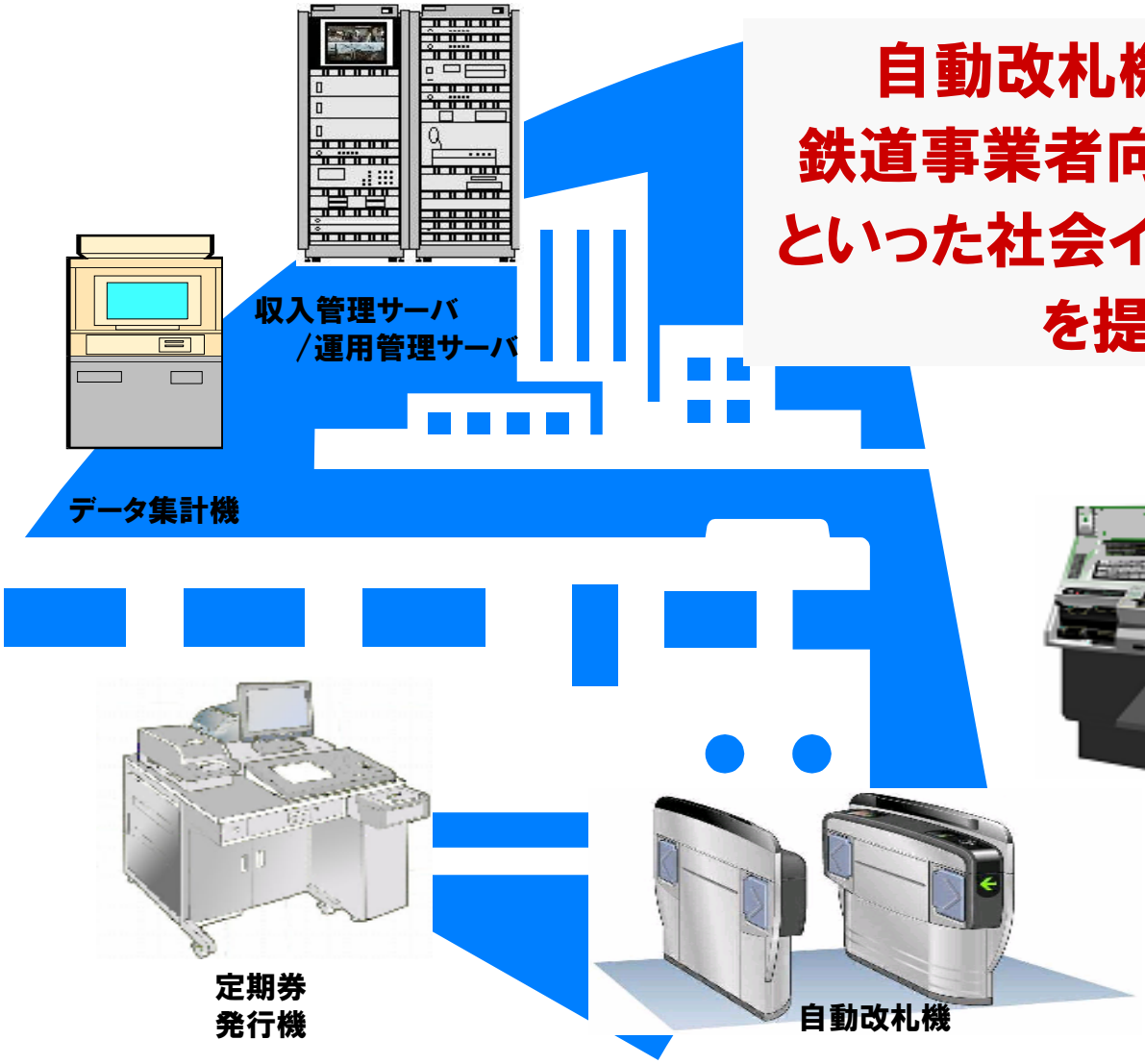
※　本発表中では、「ソフトウエアアーキテクチャ」という用語は「アーキテクチャ」と略す。

事業の紹介

OMRON
Sensing tomorrow™

公共ソリューション事業部では、

自動改札機、自動券売機等の
鉄道事業者向け機器およびシステム
といった社会インフラを支えるシステム
を提供しています



自動券売機／自動精算機

部品化とアーキテクチャ

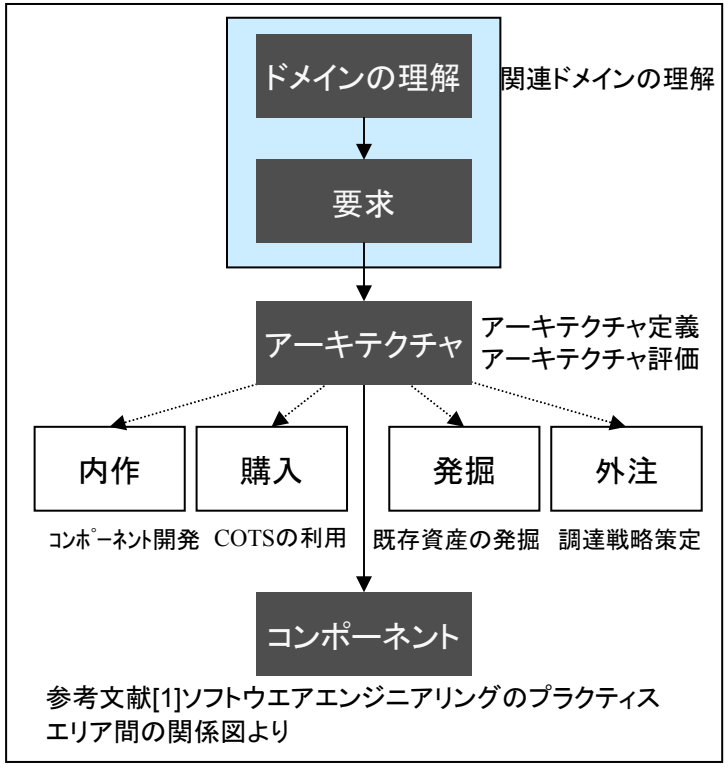
OMRON
Sensing tomorrow™

駅務機器事業ドメインは、成熟した製品シリーズを継続的に開発・保守していく事業ドメインである。

我々は、このようなドメインのソフトウエア開発において、早く安く良い製品を送り出すために、ソフトウエアをできるだけ再利用して開発するプロセスとそのための開発資産を作り上げていく取り組みを進めている。

アーキテクチャ

- アーキテクチャは、システムの構成要素や要素間のインターフェイスを定義し、規模の大きなシステムに導く指針となる記述である。
- 従って、事業ドメインにおいて、共通なアーキテクチャを確立することは、ソフトウエアの部品化と再利用を大きく促進する重要な鍵となる。



参考文献[1]ソフトウエアエンジニアリングのプラクティス
エリア間の関係図より

アーキテクチャ共通化の事例

OMRON
Sensing tomorrow™

「運賃計算ソフトウエア」の再構築 (2000年～)

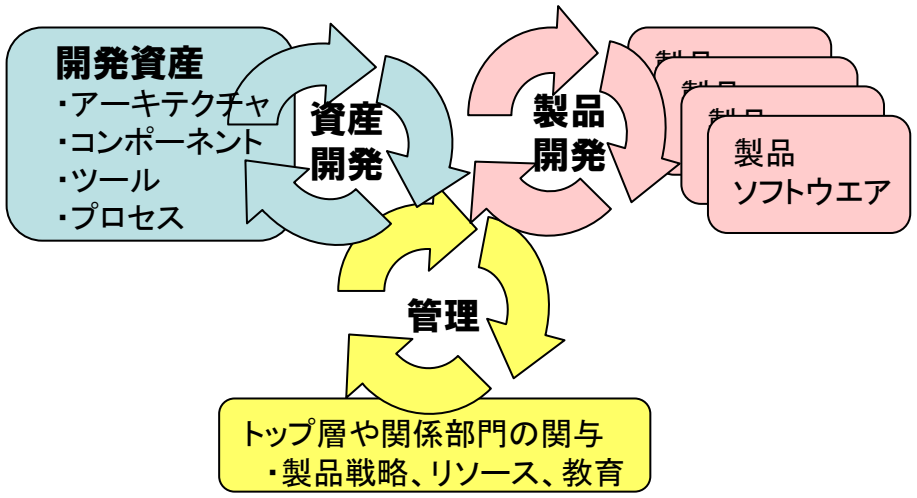
運賃不具合への厳格化	処理の複雑化と大規模化	コードバリエーション
運賃誤収受は社会問題化し、発生時は国交省への報告が義務	鉄道路線の相互乗り入れ拡大やカード共通利用から仕様範囲は大きくなる	鉄道会社毎や機種毎にコピー派生したためバリエーションが大量発生

鉄道ICカードの運用開始が関東地域や関西地域でほぼ同時期に重なるため、ICカードの運賃計算ソフトウエアのQCD達成に危機感があった。

アプローチ

そこで、高品質な運賃計算ソフトウエアを効率よく開発するプロダクトライン開発の確立に取り組んだ。

- **アーキテクチャや部品などの開発資産を構築**
- **開発資産を利用して製品を開発するプロセスを定義**



事例：アーキテクチャの開発

OMRON
Sensing tomorrow™

- ドメイン分析を行い、仕様の固定・変動部分を明確化
- 仕様の分割軸に基づき、共通アーキテクチャを定義
- OODに基づき、分割されたモジュールを部品化

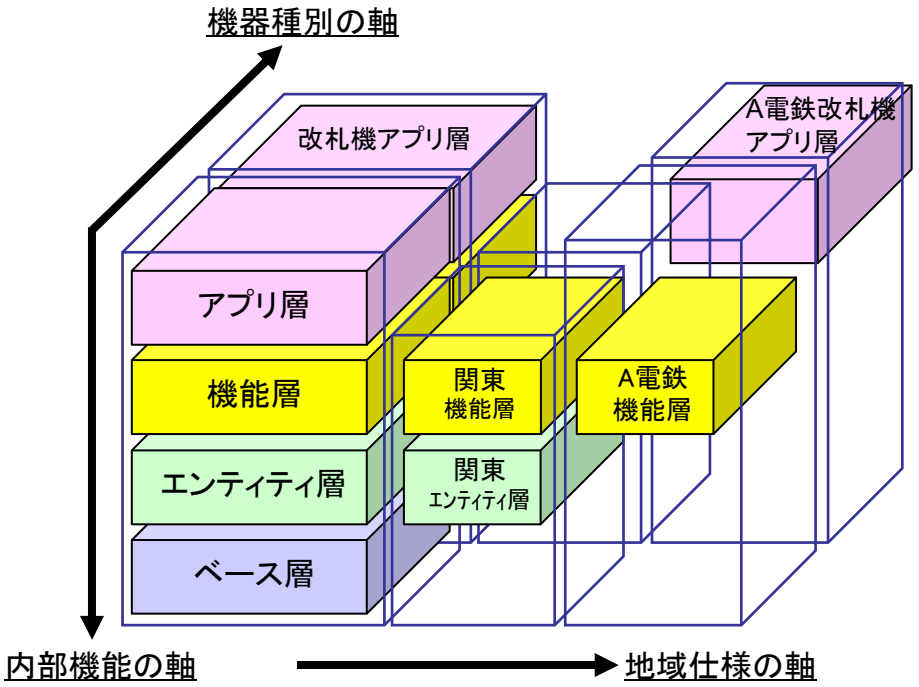
1. 固定変動分析 → **2. アーキテクチャ設計**

要求仕様の固定仕様と変動仕様を整理して分割する軸を見つける。

①内部機能の分割軸
→業務機能からミドルウエア機能、デバイス機能に至る機能単位の分割

②地域による仕様の分割軸
→関東地域と関西地域での運賃計算仕様の差異による分割

③機種による仕様の分割軸
→改札機、精算機など機種による運賃計算仕様の差異による分割



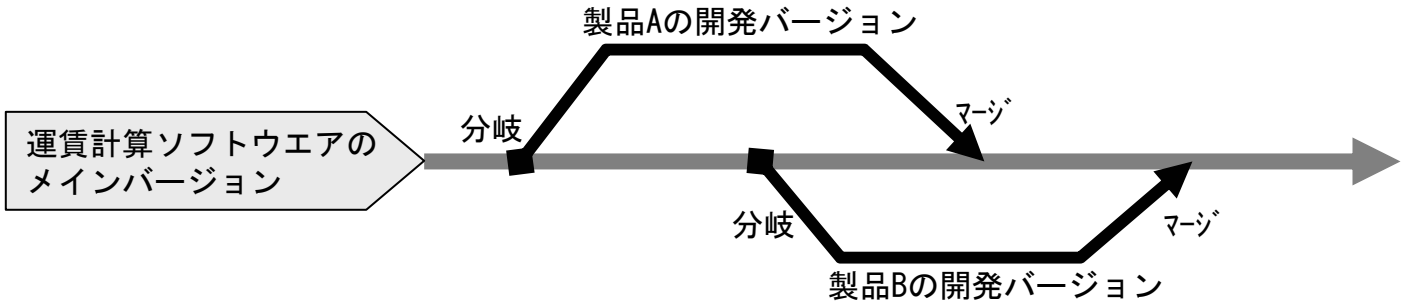
事例:アーキテクチャの管理の課題

OMRON
Sensing tomorrow™

共通アーキテクチャを設計して部品化を確立した後に、アーキテクチャの管理面において下記の対応を行った。

1.コピー派生を作らせないこと

- 並行した製品開発では、一時的にバージョン分岐を行うが、その後必ず、メインバージョンにマージすることで、メインバージョンのみを継続させた。



2.共通部品の品質が崩れないようにすること

- 複数の製品開発が並行する中で、共通部品に対する修正が必要となれば、必ず事前申請し、影響調査を経て認可する手順とした。

3.顧客にもアーキテクチャの理解を求めること

- 製品のバージョンアップで確認する差分コードの中に、ある顧客にとって無関係なバグ修正や機能追加が存在する可能性がある。その場合、顧客に対して、ソフトウエア構造上から品質に影響なきことを説明する義務がある。

事例：開発結果と成果

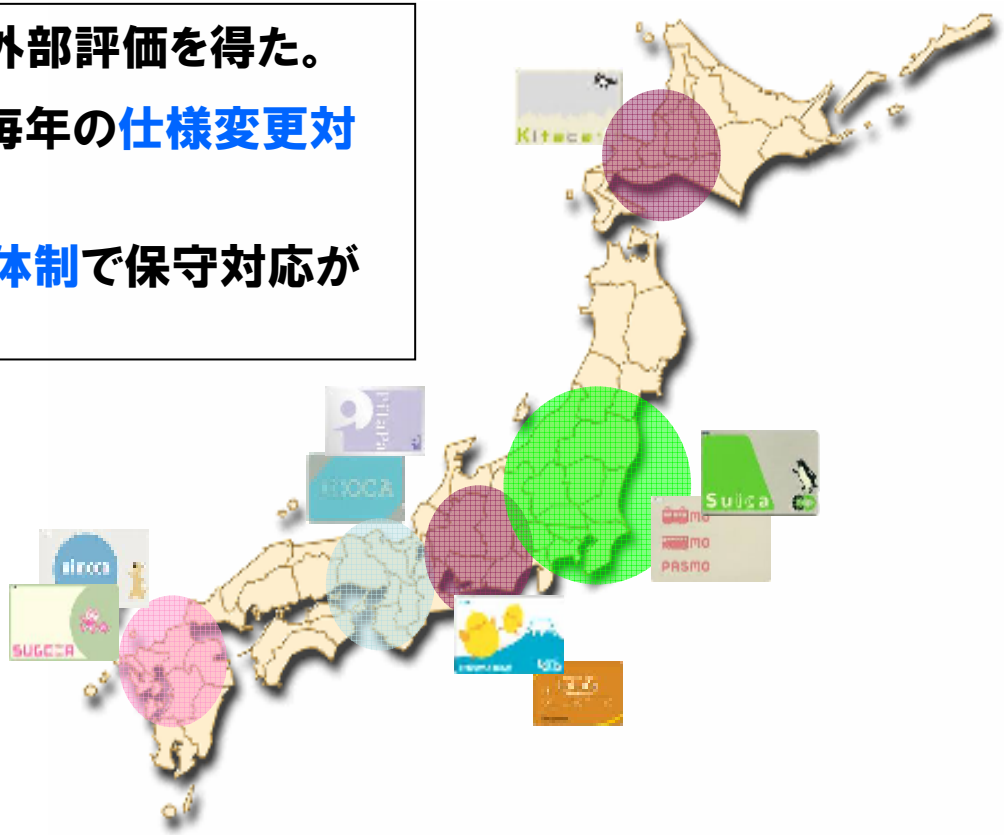
OMRON
Sensing tomorrow™

単一のアーキテクチャ上で部品化された構成要素に基づき、差分開発を行い、全国の鉄道ICカードの運賃計算を行うソフトウエアを順次開発してリリースした。

効果

- 業界トップクラスの品質であると外部評価を得た。
- 部品化されたソフト構成により、毎年の仕様変更対応が容易になった。
- 従来に比べ、少ない人数の開発体制で保守対応が可能になった。

長期的なソフトウエア開発のQCDに対して、アーキテクチャが重要な役割を果たすことを改めて認識した。

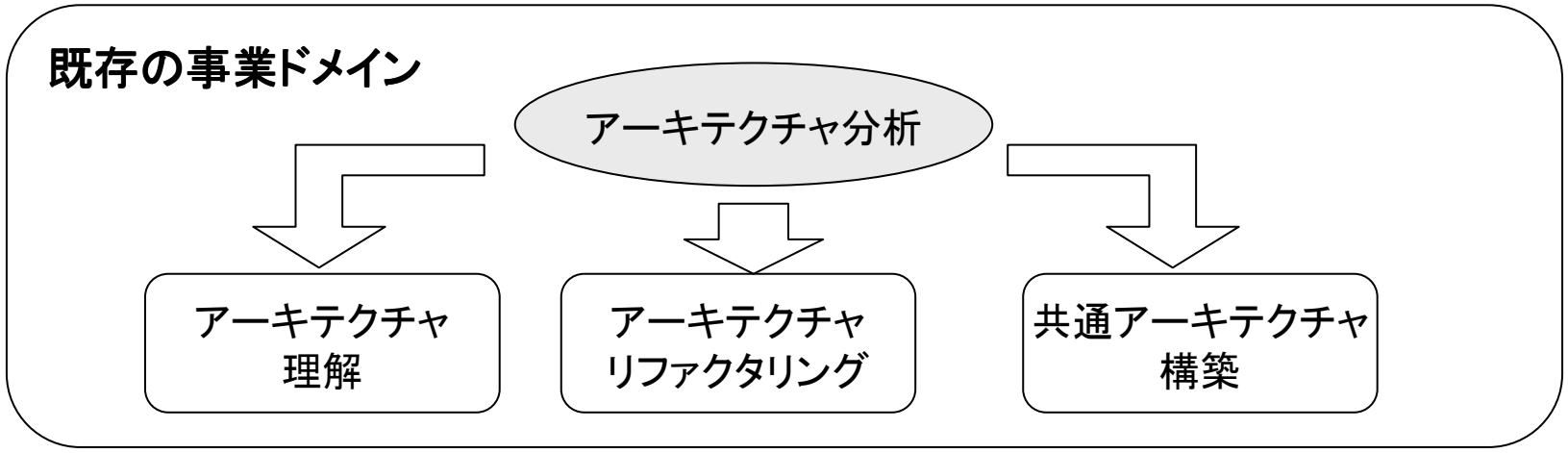


アーキテクチャ分析の目的

OMRON
Sensing tomorrow™

事業ドメイン全体の部品化・共通化を目指して

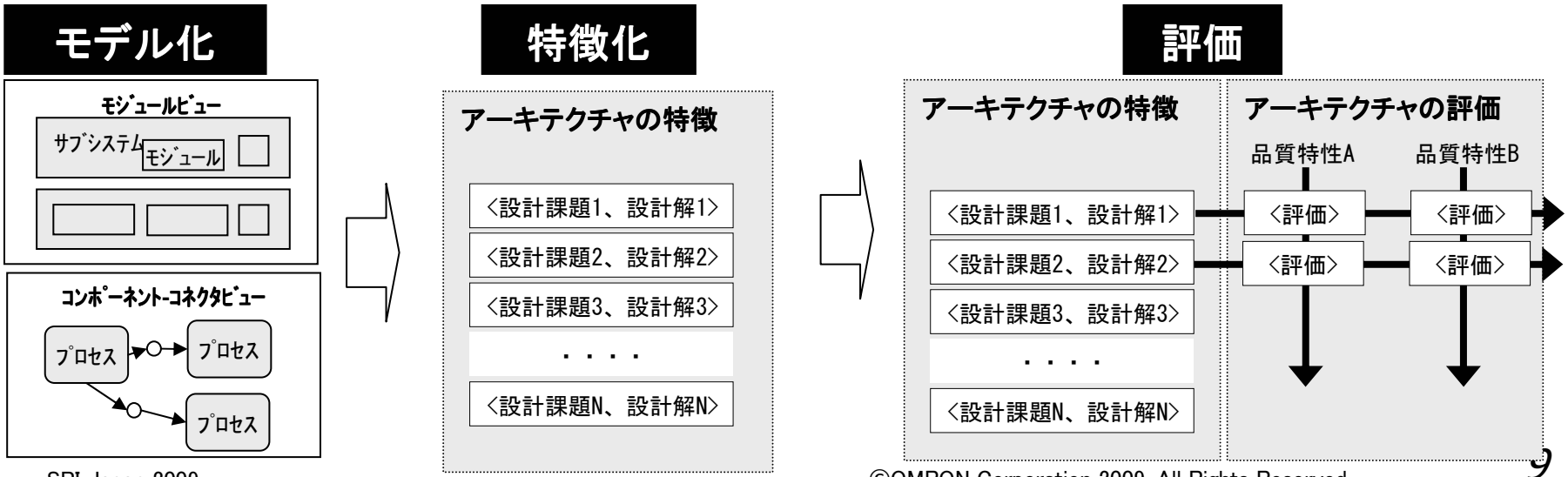
- アーキテクチャ分析の目的
  - ソフトウエアの開発効率や品質の問題において、アーキテクチャ上の課題を明らかにし、その改善策を示すこと。
  - さらに、事業ドメインの主要なアーキテクチャ群を分析対象として、ドメイン共通なアーキテクチャの要件を明らかにする。
- 分析の対象
  - 特定の事業ドメインに含まれる製品のアーキテクチャ群



アーキテクチャ分析の構成

- 大規模で複雑なアーキテクチャ全体を理解することは、開発者にとっても困難である。
- 複数のアーキテクチャを横並びに分析する場合には、調査範囲を限定して、系統立てた手順を持つ必要がある。

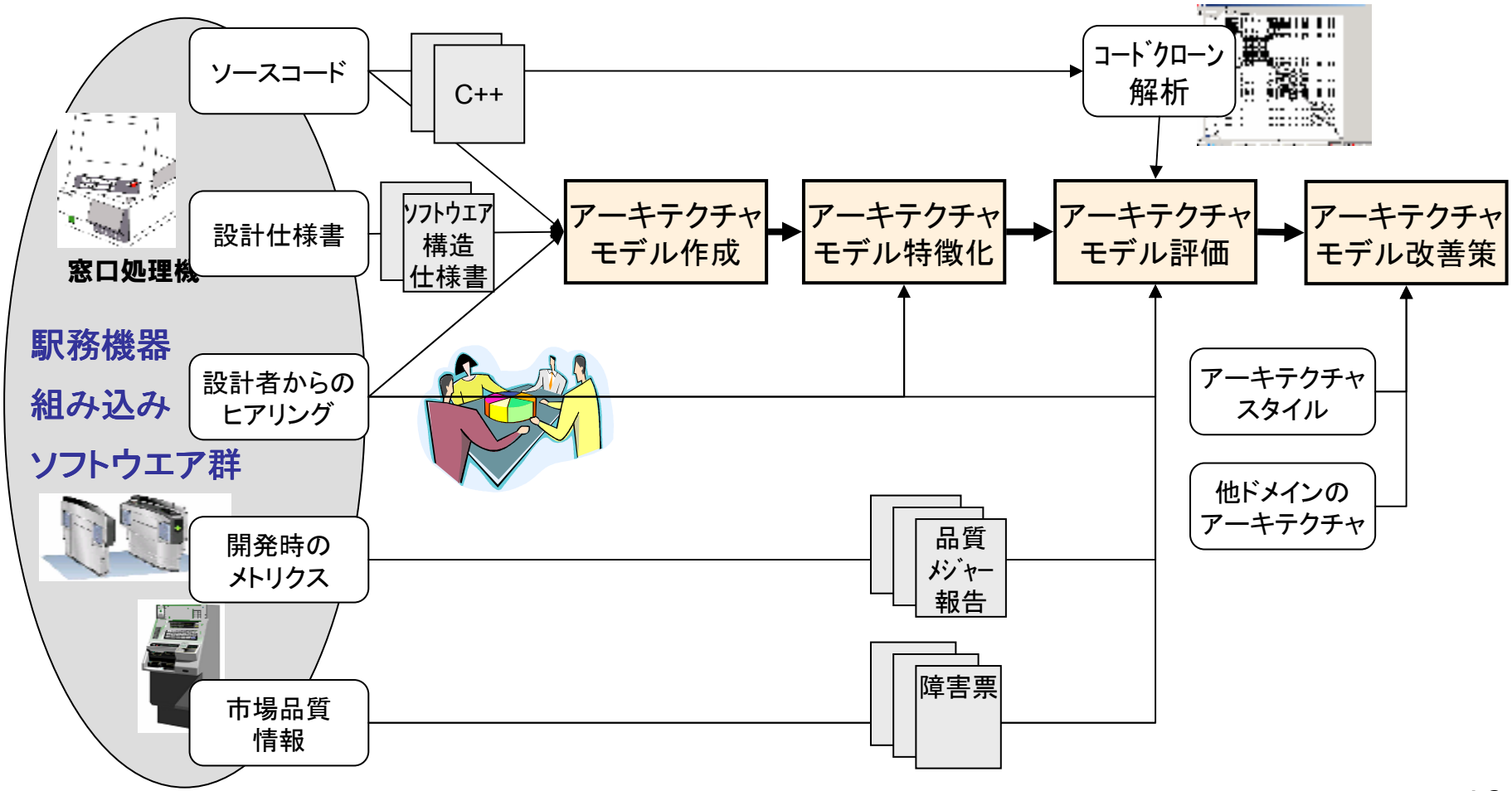
Step1 アーキテクチャのモデル化	： 複数の既存アーキテクチャに対して、実装単位（ﾓｼﾞｭｰﾙ）と実行単位（ｺﾝﾎﾟｰﾈﾝﾄ）の観点に基づき、粒度や層レベルを均一にしてモデル化する。
Step2 アーキテクチャの特徴化	： 対象ドメインに特有なアーキテクチャレベルの設計課題を定義し、既存アーキテクチャの設計解の特徴を明らかにする。
Step3 品質特性の観点から評価	： 市場に製品リリースした結果に基づき、信頼性、保守性、拡張性など品質観点から設計解を評価する。
Step4 改善策の検討	： 良くない設計解に対して、リファクタリングの改善策を示す



アーキテクチャ分析の作業フロー

OMRON
Sensing tomorrow™

分析の作業は、複数のアーキテクチャを同時に分析対象として各Stepごとに進めていく。



Step1　アーキテクチャモデルの作成

OMRON
Sensing tomorrow™

複数のアーキテクチャの特徴や性質を調べるため、共通の表現方法や観点を定めてモデル化する。

開発現場において製品ソフトウエアの設計仕様書は、記述レベルや表記方法が不統一に作成されていた。

- **アーキテクチャを記述する観点**
  - **実装単位による観点と実行単位の観点の2つのビュー。**
- **表記方法**
  - **UMLの表記法を用いてサブシステムやインターフェイスを表現する。**
- **作成方法**
  - **開発当時の設計仕様書やソースコードからソフトウエア構造を調査し、さらに開発者へのヒアリングにより意思決定の根拠を確認する。**

	モジュールビュー	コンポーネント-コネクタビュー
概要	機能分担を実装したコード部位の構成を表現する。	実行時の振る舞いを表現する。
要素	機能を集約し実装したモジュールが要素となり、例えばサブシステム、レイア、ライブラリ、パッケージ、クラスである。	コンポーネントは、主要な処理とデータ記憶が要素となり、例えばプロセス、スレッド、タスク、共有データリポジトリである。 コネクタは相互作用の仕組みが要素となり、例えばメッセージパッシング、同期化などである。
関係	モジュールを「～に含まれる」、「～に依存する」、「 ～の継承である」で関係づける。	コンポーネントがコネクタの相互作用により他のコンポーネントと関係づける。

参考文献[3]

Step2　アーキテクチャモデルの特徴化

OMRON
Sensing tomorrow™

- **設計課題の抽出**
  - **事業ドメインのシステム要件に対して、アーキテクチャレベルで重要な設計課題を定義する。例えば、リアルタイム性、拡張性、高セキュリティなど、ドメインに特有な課題あるいは優先度が高い課題が示される。**
- **設計解の特徴同定**
  - **各アーキテクチャの設計を調べ、その設計解の特徴を記述する。**

設計課題			設計解		
設計課題		説明	アーキテクチャA	アーキテクチャB	アーキテクチャC
1	計算資源の配置	ソフトウエアの実行をCPUに割りつける配置構成	・・・	・・・	・・・
2	階層構造	モジュールの依存関係に基づくシステムの階層構造			
3	モジュール分割	ソフトウエアをモジュールとして分割する方法			
4	イベント処理	UI、機器、通信で発生する事象のイベントを通知する仕組み			
5	共有データ管理	業務処理間で共有する大域的なデータを管理する仕組み			
6	並行処理の実装	周辺機器の動作や通信、業務を並行で処理する方法			
7	業務処理の状態管理	業務処理や機器動作の状態を表現し管理する仕組み			
8	リアルタイム処理	一定時間内で処理完了を保証するなどの仕組み			
9	排他制御	資源が同時に複数からのアクセスに対して不整合を防ぐ仕組み			
10	障害処理	異常状態を検知し通知する、さらに復旧するための仕組み			
11	ﾄﾗﾝｻﾞｸｼｮﾝの完全性	電断時などのデータ保証性、2重化、ロールバックへの対応方法			

設計解の記述では、複数のアーキテクチャ間で設計解の比較が行えるように、ある程度抽象化した記述が望ましい。

例えば、アーキテクチャスタイルやデザインパターンの用語を用いる。

Step3　品質特性の観点による評価

OMRON
Sensing tomorrow™

- **品質項目**
  - **ソフトウエア品質特性から、特にアーキテクチャ設計で責務の高い品質特性として　信頼性、効率性、保守性 、移植性、安全性を評価の観点とする。**
- **評価の方法**
  - **製品開発における開発効率や不具合の混入状況などから、アーキテクチャの設計解が、良い設計解であったのか、そうでないのか評価する。**

設計課題		設計解（アーキテクチャA）	品質特性観点の評価				
			信頼性	効率性	保守性	移植性	安全性
1	計算資源の配置						
2	階層構造						
3	モジュール分割						
4	イベント処理						
5	共有データ管理						
6	並行処理の実装						
7	業務処理の状態管理						
8	リアルタイム処理						
9	排他制御						
10	障害処理						
11	トランザクションの完全性						

各設計課題の設計解に対して、市場不具合や開発での問題などから判断し、評価を記述する。

保守性：データの通用範囲が局所化されていることや、モジュール分割で機能やデータが隠蔽されていること等

効率性：メモリ消費量が適切になるように配慮されている、性能目標を満たす高速化手段が正しく取られていること等

移植性：ハードウェアやOSの変更に対して階層構造により影響する範囲が局所化されていること等

信頼性：　メモリの破壊や不正アクセスを防止する機構、デッドロックを防止する排他制御機構が対応されていること等

安全性：個人データの暗号化や、セキュリティの高い部分のパッケージ化ができていること等

Step4　アーキテクチャの改善案

OMRON
Sensing tomorrow™

- **改善案**
  - 品質特性観点の評価が低い設計解に対して、リファクタリングのための改善案を示す。
- **検討方法**
  - ドメイン内の複数のアーキテクチャを比較して、その中で優れた設計解を検討して改善案を示す。

設計課題		設計解（アーキテクチャA）	品質特性観点の評価					改善策
			信頼性	効率性	保守性	移植性	安全性	
1	計算資源の配置	・・・						
2	階層構造	・・・						
3	モジュール分割	・・・						
4	イベント処理	・・・						
5	共有データ管理	・・・						
6	並行処理の実装	・・・						
7	業務処理の状態管理	・・・						
8	リアルタイム処理	・・・						
9	排他制御	・・・						
10	障害処理	・・・						
11	ﾄﾗﾝｻﾞｸｼｮﾝの完全性	・・・						

分析したアーキテクチャ同士の比較のみならず、他ドメインのアーキテクチャ、あるいはアーキテクチャスタイル（パターン）なども参考にして、優れた設計解を十分に検討する。

実施結果

OMRON
Sensing tomorrow™

今回の分析の目的

駅務機器事業ドメインにおけるソフトウエア部品化を推進するため、
現状アーキテクチャの問題と課題を把握すること、
そこから共通アーキテクチャに向けた要件抽出を目的とした。

対象ドメイン		駅務機器ソフトウエア
分析対象アーキテクチャ		券売機、自動改札機、窓口処理機の 組み込みソフトウエア3種 ※参考として、銀行ATMドメインの2種類の アーキテクチャも分析対象とした。
ソフトウエア規模（1機器全体）		500Kステップ～2000Kステップ
分析期間		約 4か月 モデル化：2ヶ月、特徴化と評価：1ヶ月、改善策：1ヶ月
メンバ	分析者	2名
	レビューア	5名
	ヒアリング対象者	8名

実施結果：分析の例

OMRON
Sensing tomorrow™

改札機、窓口処理機、券売機の既存ソフトウエアに対して、

アーキテクチャモデル化 ⇒ 特徴化 ⇒ 品質評価 ⇒ 改善策

のステップでアーキテクチャ分析を実施した。

ある製品のアーキテクチャ分析の例

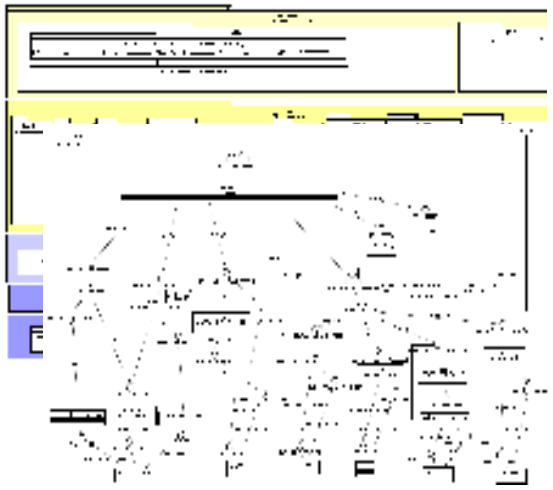
設計課題		設計解（ｱｰｷﾃｸﾁｬA）	品質特性観点の評価					改善策
			信頼性	効率性	保守性	移植性	安全性	
1	計算資源の配置	本体部と周辺機部にわかれ、前者は各デバイスを制御し業務機能を実行、後者はICｶｰﾄﾞ処理の業務機能を実行	ー	本体と周辺機間の通信速度が低い	ICｶｰﾄﾞ業務の変更が本体と周辺機双方に影響し保守性が低い	ー	ｾｷｭﾘﾃｨ性が高いﾃﾞｰﾀ処理が適切に局所化されている	ICカード業務用に画面プロセスを別に1つ起動し、周辺機が1画面デバイスとして操作する
4	イベント処理	実行表示されている画面クラスに直接イベントが集約する	画面切り替わりでｲﾍﾞﾝﾄ消失する問題	ー	ｲﾍﾞﾝﾄ追加による影響把握が困難	ー	ー	ｲﾍﾞﾝﾄのｷｭｰ管理とﾃﾞｨｽﾊﾟｯﾁを行うを行うｲﾍﾞﾝﾄ管理部を設ける

実施結果：各ステップの成果

OMRON
Sensing tomorrow™

Step1 モデル化

・改札機、券売機、窓口処理機のそれぞれのモジュール構成、タスクやイベント構成をUMLモデル化。



Step2 特徴抽出

・駅務業務や組み込みシステムとして求められるアーキテクチャの設計項目として11項目を設定。

・各アーキテクチャの設計解の一覧表を作成。

・3機種のアーキテクチャの類似する部分と異なる部分を特徴化した。

Step3 品質観点の評価

・製品の品質情報の収集
  ・市場不具合の発生分布
  ・類似コードの存在
  ・修正が困難な部位
  ・性能上の問題 など

・5つの品質項目ごとにアーキテクチャの設計解を評価。

・各アーキテクチャの強みと弱みが明確になった。

Step4 アーキテクチャモデルの改善案

・弱みとなる箇所に対してリファクタリングの必要性と改善策をコスト試算も含め提示した。
（銀行ATMシステムのアーキテクチャに存在する設計解も参考にした）

・さらに、ドメインに共通なアーキテクチャ特性（階層分割、共通な機能要素など）を示した。

考察

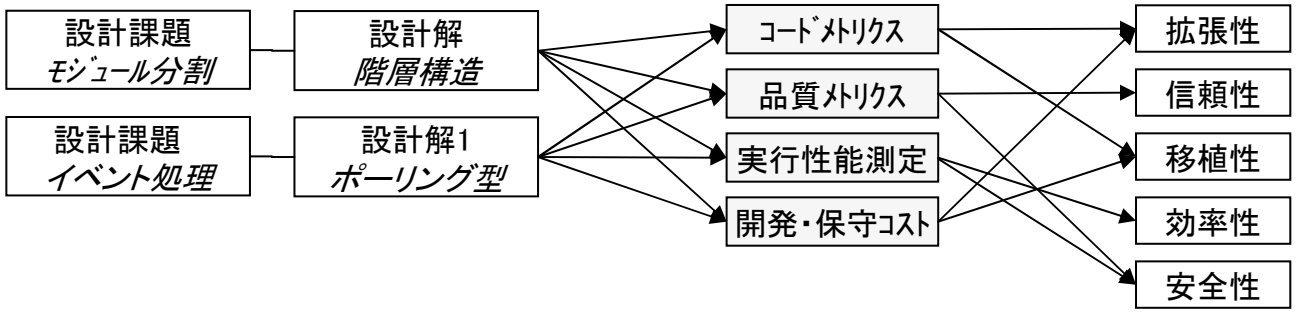
OMRON
Sensing tomorrow™

1. アーキテクチャ分析結果の用途について

- アーキテクチャリファクタリングのプラニング
  - 保守のコストが増えてきた、重大な障害が多く発生する、バグが除去しにくいなどの問題が顕在化した時に、リファクタリングのプランの材料を提供する。
- 組織的な知識共有
  - 異なる製品の開発者がお互いのアーキテクチャ情報を共有する。またマネージャ、開発部門以外の関係者に、アーキテクチャの基本知識を与える。

2. 分析方法の改善について

- 定量的データに基づくアーキテクチャの品質評価
  - アーキテクチャの設計解と、それが原因となるバグ件数や過大な修正工数、性能劣化など定量データとの影響関係を定式化することで、評価の客観性が高められる。



最後に

OMRON
Sensing tomorrow™

まとめ

- 事業ドメインの主要製品のアーキテクチャ群を対象する、4つのステップから構成されるアーキテクチャ分析を示した。
  - アーキテクチャのモデル化
  - アーキテクチャレベルの設計課題に基づく特徴化
  - 複数のアーキテクチャの強みや弱みを品質項目に基づき評価
  - 複数のアーキテクチャ間で比較することで、より良い改善策を検討
    - ⇒ 得られた改善策は、ドメイン共通なアーキテクチャの要件に含まれる
- 上記のステップを駅務機器事業ドメインで実施して、比較的短い期間であったが、分析報告を作成した。

成果の利用

現在、この分析の成果は、

1. 既存アーキテクチャのリファクタリングへのインプット
2. 事業ドメインに共通なアーキテクチャの検討着手

に利用されている。

参考文献

[1]ソフトウェアプロダクトライン―ユビキタスネットワーク時代のソフトウェアビジネス戦略と実践, Paul Clements, Linda Northrop, 前田 卓雄 (翻訳), 日刊工業新聞社 (2003/10)

[2]ソフトウェアアーキテクチャドキュメント, Paul Clements, Len Bass (著), David Garlan (著), James Ivers (著), Felix Bachmann (著), 前田 卓雄 (翻訳), 日刊工業新聞社 (2005/10)

[3]実践ソフトウエアアーキテクチャ, Len Bass, Rick Kazman, Paul Clements,前田 卓雄 (翻訳), 日刊工業新聞社 (2004/10)

[4] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural Mismatch: Why Reuse Is So Hard," IEEE Software, Nov./Dec. 1995,pp. 17-26.