ソフトウェア・プロダクトライン・エンジニアリングの理論と実践

～プロダクトラインの基礎とマッサージ・チェア開発への適用事例

**本発表資料は、九州日立マクセル株式会社
安部田章氏との共同の発表資料のうち
前半の橋本担当の部分を抜粋したものです。**

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL INC.

橋本 隆成

アジェンダ

- 発表者経歴紹介
- 第１部：
  - ソフトウェア・プロダクトライン・エンジニアリングの理論
- 第２部：
  - プロダクトライン・エンジニアリング事例～マッサージチェア開発を題材にして実例の解説
- 質疑応答（時間があれば）

発表者略歴

- 橋本 隆成
  - HASHIMOTO Software Consulting International Inc.
  - SEIビジネスパートナー
  - SEI認定CMMIインストラクタ
- 業務略歴
  - 三菱スペースソフトエア（株）にて、弾道計算プログラム、戦闘指揮システム、射撃制御システムなど大規模防衛リアルタイムシステムの開発に従事
  - 日本ヒューレット・パッカードにてネットワークシステムのシステム・エンジニアリング業務に従事
  - （株）オージス総研にてオブジェクト指向を中心としたコンサルテーション、インストラクタ業務に従事
  - ソニー（株）にて携帯電話開発、プレイステーション3、CMMIによるプロセス改善業務、ソニー全社改革推進業務に従事
  - HASHIMOTO Software Consulting International Inc現職

第1部：
ソフトウェア・プロダクトライン・エンジニアリングの理論

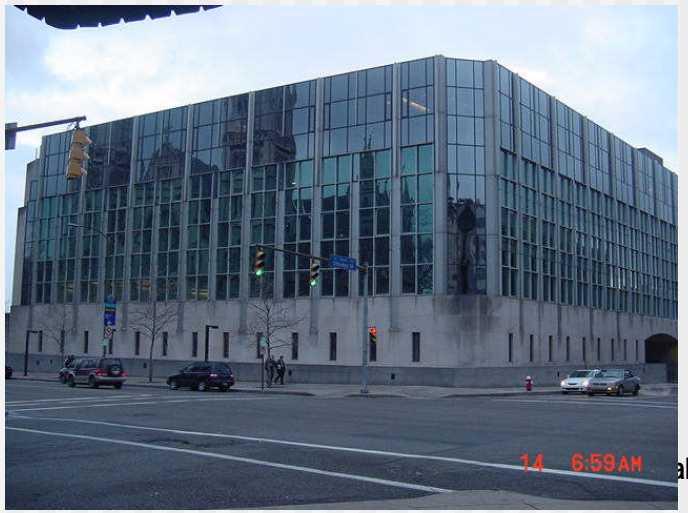
1. ソフトウェア・プロダクトライン・エンジニアリングの基礎
2. ソフトウェア・プロダクトライン・エンジニアリングを支援する技術
3. ソフトウェア・プロダクトライン・エンジニアリングの手法・方法論

ソフトウェア・プロダクトライン・エンジニアリングの基礎

1. プロダクトライン・エンジニアリングとは何か？
2. プロダクトライン・エンジニアリングの狙い
3. プロダクトライン・エンジニアリングが期待される理由
4. プロダクトライン・エンジニアリングの用語
5. プロダクトライン・エンジニアリングの全体像
6. プロダクトライン・エンジニアリングとの構成
7. プロダクトライン・エンジニアリング開発の流れとプラクティスエリアの関係

プロダクトライン・エンジニアリングとは１？

- USカーネギーメロン大学ソフトウェア工学研究所（SEI）で開発・体系化
- 製品がプロダクトラインとしてシリーズ化、ラインナップ構成が存在するものに効果大
  - 特にミッションクリティカルなシステムが多いリアルタイムシステム、組込みソフトウェアの課題が深刻であった
- 近年、多くの研究者たちがプロダクトラインのアプローチを提唱している
  - PLUS
  - PuLSE
  - KobrA
  - Software Factories（MS社）
  - etc

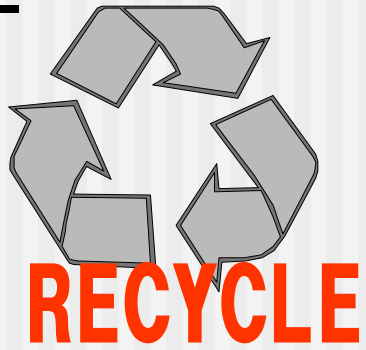


プロダクトライン・エンジニアリングとは？２

- 製造業の製品は組込み・リアルタイムシステムである
  - ライフサイクルが長い
  - 新規の開発よりもシステムの**改修・保守による製品開発**
    - 多くはアーキテクチャをカスタマイズして、製品を開発していくアプローチ
  - **高品質**が要求されるシステム
    - 短期間での開発に限度があり、信頼性が重要視されるシステム
    - リアルタイム性、フォールトトレランス性など難しい要求に対応
  - 製造業は開発する**製品間に多くの共通特長**がある
    - 組織戦略による再利用アプローチ
    - ソフトウエア資産とカスタマイズ
- 市場調査、組織編制、予算の獲得など**非エンジニアリングの活動領域も重視**
  - ソフトウエアの再利用は技術的な問題だけでない
  - ソフトウエア開発は組織・企業の包括的かつ統合的なアプローチ

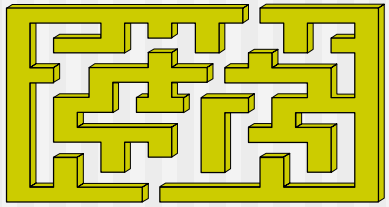
プロダクトライン・エンジニアリングの狙い

- 開発期間の短縮
  - 市場調査～要件開発～テスト・保守までをカバー
- 生産性の向上
  - システマティックなSWの再利用に期待
- 品質の向上
  - SW資産の再利用により新規開発は最低限にする
  - SW資産の再利用による品質の向上
- 開発エンジニア、テスト担当者の不足の軽減
  - 重複・無駄な作業の排除による担当者の拡散を防止
  - SW資産の再利用とカスタマイズによる作業負荷の削減
- 効果なソフトウエア資産の再利用
  - 統合的な技術戦略
  - 組織体制、マネージメントの重視
  - パターンによるアプローチ

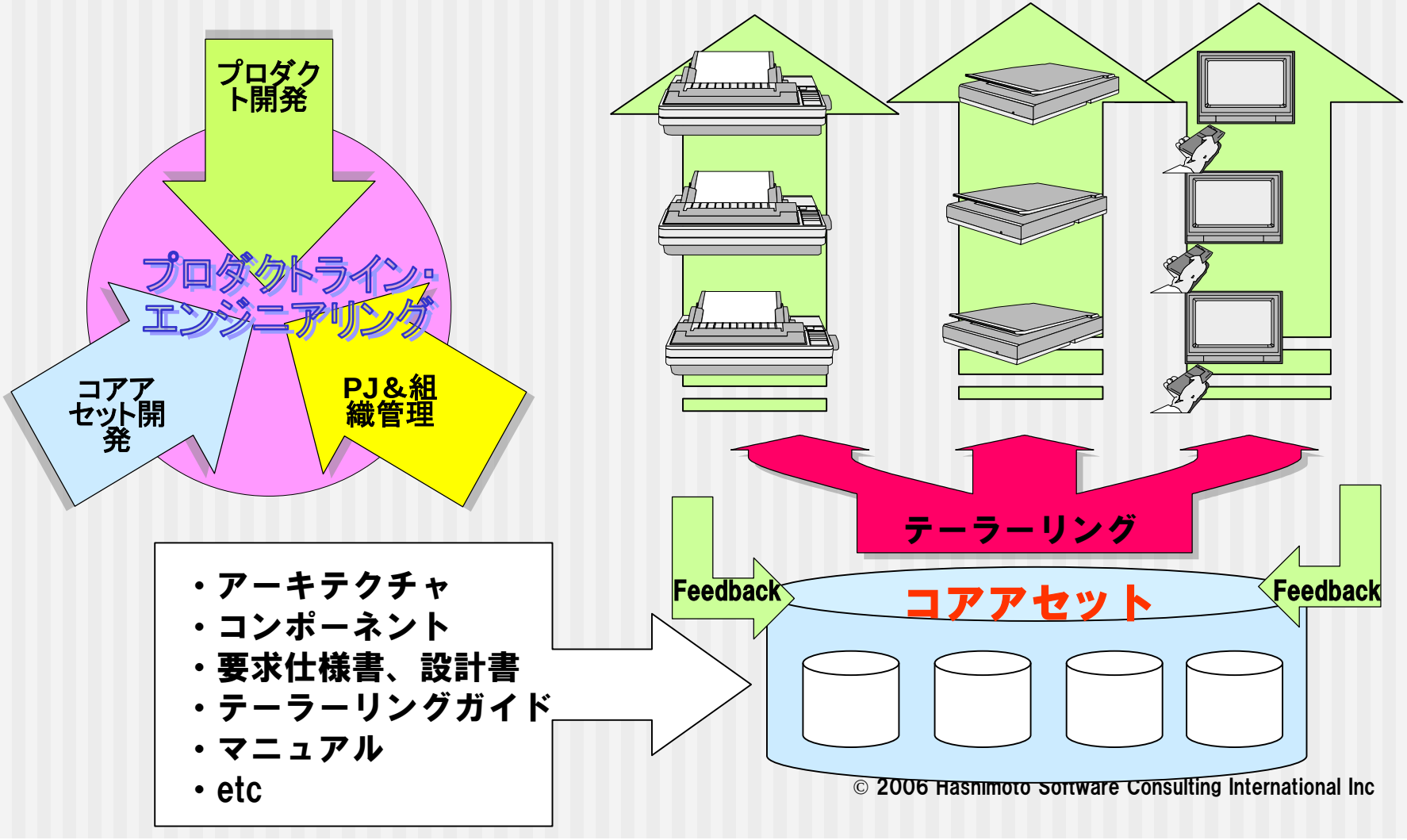


プロダクトライン・エンジニアリングが期待される理由

- 従来の手法、方法論ではソフトウエア開発の課題の解決に**不十分**であることが分かってきた
  - ソフトウエアの開発技術が発展上であった
    - ソフトウエアの大規模化・複雑化
    - ソフトウエア利用の拡大
  - 開発手法・方法論、改善活動の手法が現場での製品開発作業の一部しかカバーしていない
    - オブジェクト指向技術・コンポーネント指向開発
    - SEI-CMMI
    - ISO9000など
  - 日本および国際情勢、インターネットの普及などソフトウエア開発を取巻く社会環境の変化
    - 人口の減少への開発戦略の変化
    - BRICs
    - 分散開発



プロダクトライン・エンジニアリングの全体像



プロダクトライン・エンジニアリングで用いる用語

- コアアセット（コア資産、組織資産）
  - プロダクトラインエンジニアリングで共通資産としてのアーキテクチャ、コンポーネント、ドキュメントなど
- テーラーリング（カスタマイズ）
  - コアアセットを各製品（プロダクト）に利用する際に、製品にあうようにする活動
- プロダクト
  - 各製品
- プロダクト・ファミリー
  - コアアセットで開発される共通性のある製品群
- 共通特長
  - プロダクト・ファミリーで見られる共通した機能・非機能
- 可変性
  - 共通特長に対して、要求（機能、非機能）の変更
- スコープ
  - プロダクトライン・エンジニアリングを適用する範囲
  - スコープが決定したら、ドメインエンジニアリングを行なう
- ドメイン・エンジニアリング
  - 体系的な再利用を行なうために、開発対象を特徴などからいくつかのドメインに分割し、開発するシステムのドメインに共通性と可変性を分析する作業。
  - オブジェクト指向では、堅牢的なアーキテクチャ開発のために、概念モデルなどを作成する

プロダクトライン・エンジニアリングの構成１

- SEIのソフトウエア・プロダクトライン
  - 包括的にプロダクトラインの活動をカバーしている
  - 他のプロダクトラインの手法・方法論が基礎としてい
  - 具体的なHowToよりは活動のWhatを明確にしている
    - 自分たちのやり方に落とし込む必要がある
    - 他のプロダクトラインの手法・方法論がHowToを紹介
    - ただし、色々なアプローチを文献として紹介している
- プロダクトラインの活動を3つのカテゴリー、29のプラクティス・エリアで構成
  - ソフトウエアエンジニアリングプラクティスエリア
  - プロジェクト管理プラクティスエリア
  - 組織管理プラクティスエリア
  - CMMIのような成熟度、能力度レベルは存在しない
- 適用のためのパターンを示している
  - ３つのカテゴリー、29のプラクティス・エリアが静的構成
  - パターンは活動に対する複数のプラクティス・エリアを使った動的構成

プロダクトライン・エンジニアリングの構成２

ソフトウェア・エンジニアリング

１．アーキテクチャ定義
２．アーキテクチャ評価
３．コンポーネント開発
４．ＣＯＴＳの利用
５．既存資産の発掘
６．要求エンジニアリング
７．ソフトウェアシステム統合
８．試験
９．関連ドメインの理解

プロジェクト管理

１．構成管理
２．データ収集/メトリクス/追跡
３．（ソフトウェア）内作/購入/発掘/外注の分析
４．プロセス定義
５．スコープ定義
６．プロジェクト計画策定
７．プロジェクトリスク管理
８．ツールによる支援

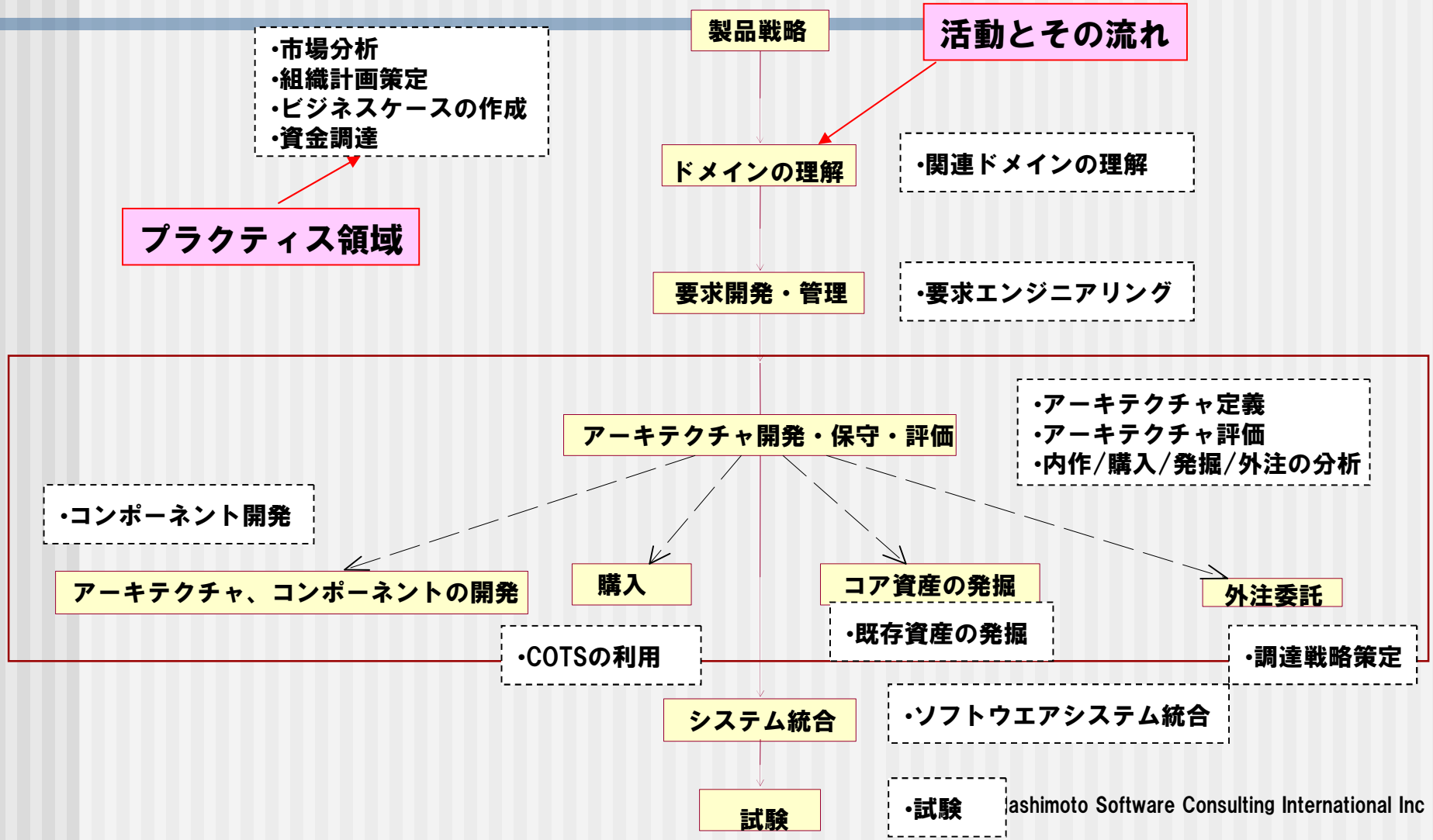
組織管理

１．ビジネスケースの策定
２．顧客インタフェース管理
３．調達戦略策定
４．資金調達
５．プロダクトラインの着手及び制度化
６．市場分析
７．プロダクトライン運営
８．組織計画策定
９．組織リスク管理
10．組織編成
11．技術予測
12．トレーニング

プロダクトライン・エンジニアリングの構成３

- プラクティスエリアの記述構成
  - プラクティスエリアの名称
    - （例：コンポーネント開発CD）
  - 導入部
    - プラクティスエリアの活動の目的、用語解説
  - 解説部
    - プロダクトライン固有の見方
    - コアアセット開発への適用
    - プロダクト開発への適用
    - 特定プラクティス
    - プラクティスのリスク
    - 追加資料
    - 検討課題

プロダクトライン開発の流れとプラクティスエリアの関係



プロダクトライン・エンジニアリングを支援する技術

1. 従来再利用が加速しなかった原因
2. ドメインエンジニアリング
3. Feature Orientedアプローチ
4. クロスカットの関心の分離

従来再利用が促進しなかった原因１～コピー&ペースト

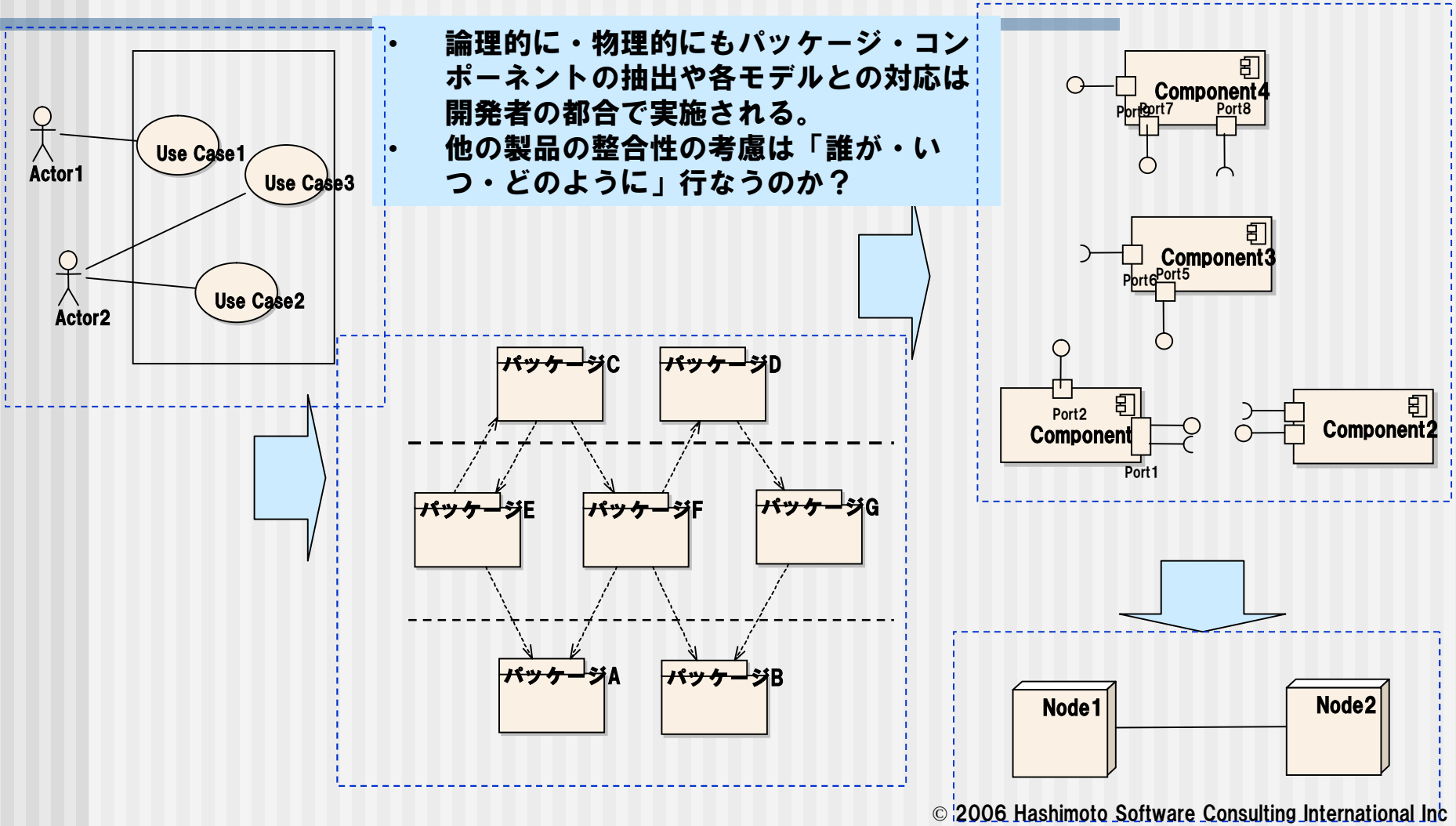
- 古くは「コピー&ペースト」の再利用（？）
  - ソフトウエア開発の歴史と同時に行なわれた方法
    - 現在でも行なわれている
    - これを止めろというのは難しい
  - 管理不能
    - コードのカジュアルなコピーとカスタマイズ
    - エンジニアの個人的な能力とやり方に依存
    - コードの利用の履歴は残らない
  - 不具合も同時にコピー
    - 十分に品質が確認されたものをコピー&ペーストするわけではない
    - コピー&ペーストされたコードが、また、コピー&ペーストされる

従来再利用が促進しなかった原因２～手法・方法論

-
- 開発などの技術が発展途上だった
  - クラス単位の再利用は難しい
  - カプセル化、継承、委譲などへ過度の期待
  - パッケージ、サブシステム単位の再利用にも課題
  - パターンも限界がある
- ソフトウエアの部品化への課題
  - コンポーネントベース開発・再利用が加速していない
    - コンポーネントの定義も漠然としていた
    - コンポーネントの抽出と要件との対応には課題が多い

パッケージ、コンポーネントの抽出と要件との対応

- 論理的に・物理的にもパッケージ・コンポーネントの抽出や各モデルとの対応は開発者の都合で実施される。
- 他の製品の整合性の考慮は「誰が・いつ・どのように」行なうのか？



従来再利用が促進しなかった原因３～複雑性の問題

- １つの開発技術、ソリューションだけではソフトウエア開発・保守上の課題が解決できない
  - マルチパラダイムの必要性
    - ドメイン・エンジニアリング
    - フィーチャー指向分析
    - オブジェクト指向技術
    - アスペクト指向技術
    - etc
  - 横断的な関心事の分離の問題
    - 拡散（散らばり）
    - もつれ合い
- アーキテクチャのカスタマイズ技術
    - 可変性への対応
    - 統合（複合化）技術
    - バインディング・タイムの選択

アーキテクチャのカスタマイズ技術の課題

- 可変性の実現方法と選択
  - 可変性を明確にした後、設計で実現方法の選択と決定
  - 実現方法にはオブジェクト指向の継承・委譲・ポリモフィズムなどが一般的
- 統合（複合化、バインディング）技術
  - アーキテクチャを構成するコンポーネント、可変性の実装と統合をどのように実現させるかの技術
- バインディング・タイム
  - アーキテクチャを構成するコンポーネント、可変性を実現方法に応じて決定される
    - コンパイル時
    - リンク時
    - コード動作時

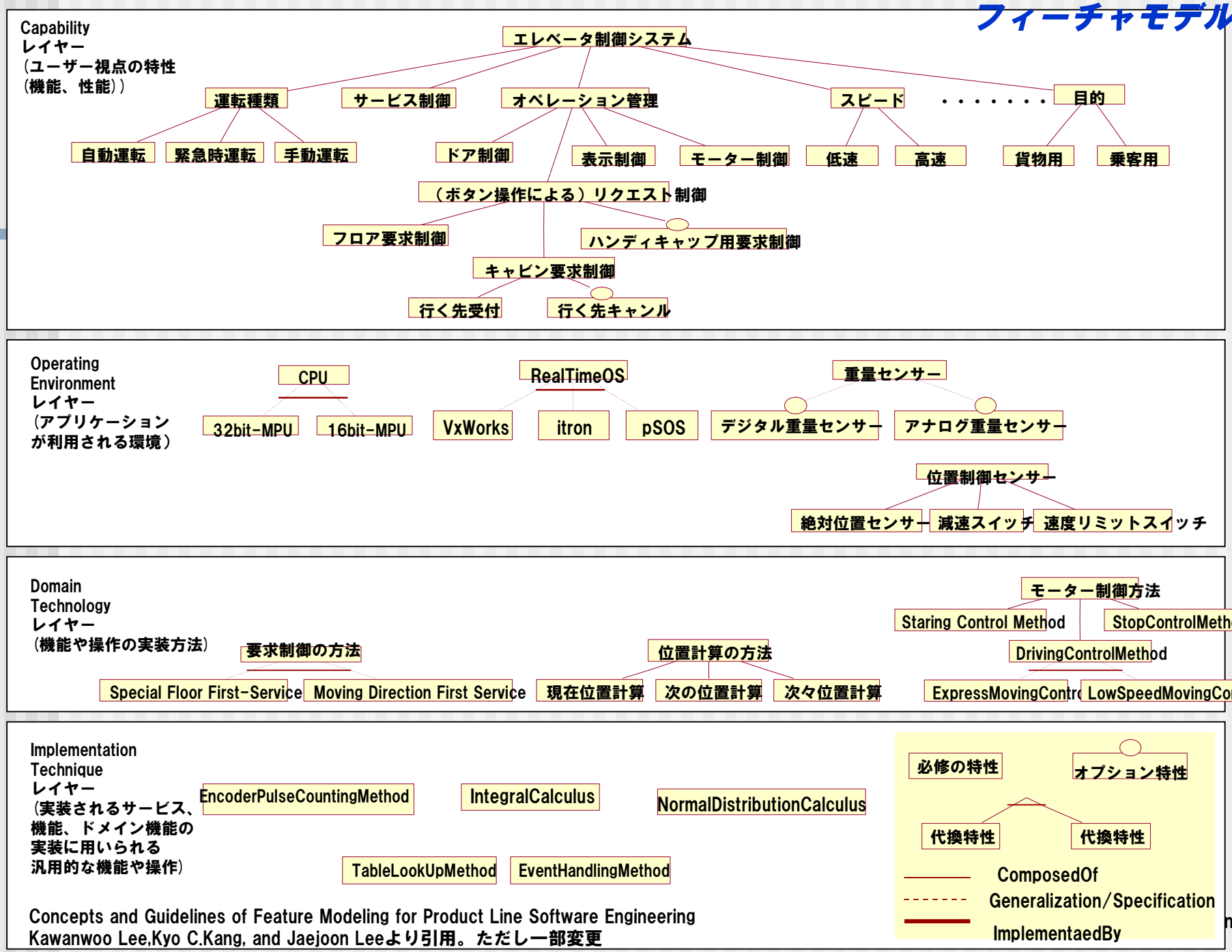
プロダクトライン・エンジニアリングの再利用戦略１

- ソフトウエアの**複雑性**への積極的な対応
  - **マルチパラダイム戦略**の必要性
    - 要求エンジニアリング
      - ユースケースアプローチ
      - フィーチャ指向アプローチ
      - ドメイン・エンジニアリング
    - オブジェクト指向技術・コンポーネント技術
    - プロダクトライン・エンジニアリングの考え方
    - マルチポイントビューの考え方
  - **スコープの限定（プロダクト・ファミリー）**
    - **汎用的**なソフトウエア部品の開発と利用は困難である
    - **コントラクト**と**コンテキスト**を考慮した“契約に基づいた設計”の重要性
      - 非機能要求の明示
    - **組込み・リアルタイムシステム**ではこの問題は顕著

フィーチャ指向アプローチ

- 再利用性のための共通性・可変性の抽出や体系化に有効な技法
  - 従来の要求仕様書、ユースケースモデリングを補う
  - フィーチャモデルで表現する
  - FORDおよびFORMが有名
- 機能・非機能の両方を「特性」として扱う
  - ユーザー視点、エンジニアからの視点の両方で表現
  - ユースケースはここでの「特性」をどのように利用するかという点になる

フィーチャモデル



Concepts and Guidelines of Feature Modeling for Product Line Software Engineering
Kawanwoo Lee,Kyo C.Kang, and Jaejoon Leeより引用。ただし一部変更

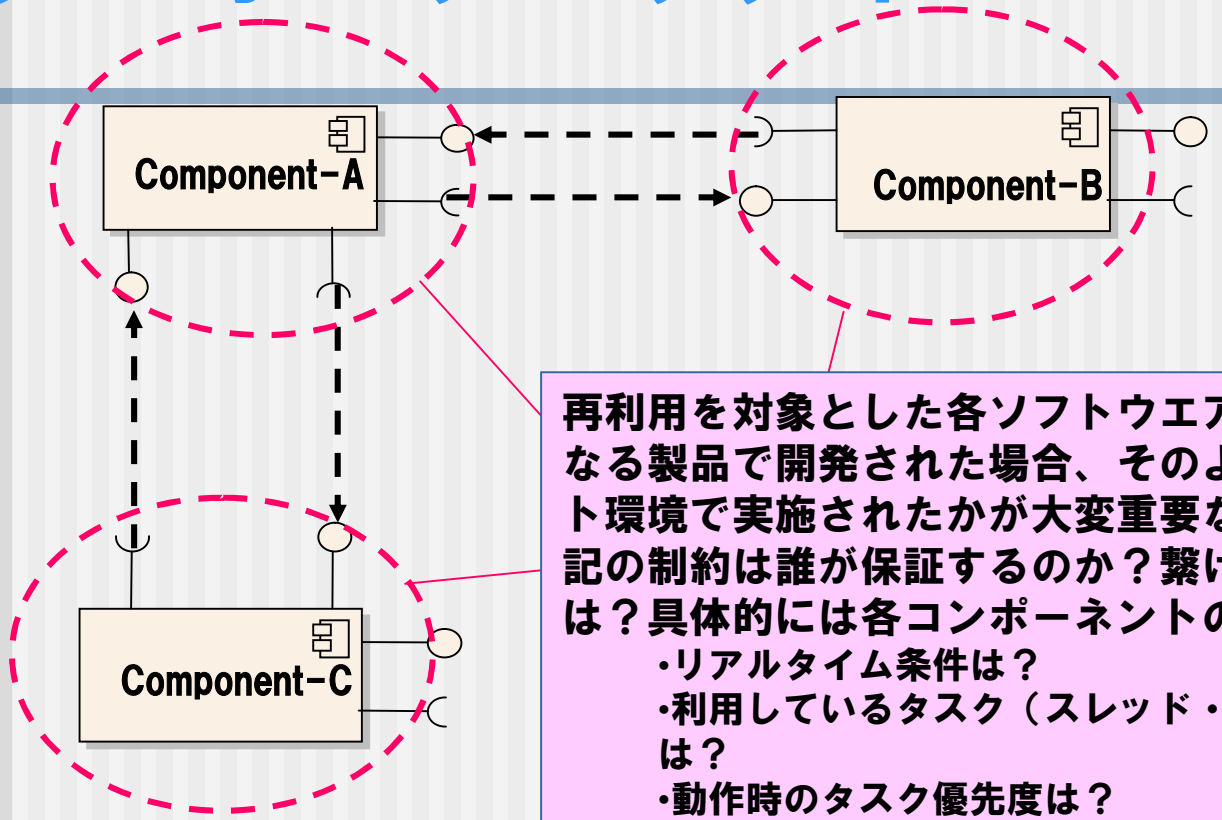
プロダクトライン・エンジニアリングの再利用戦略２

- マルチパラダイム戦略
  - オブジェクト指向技術
    - 可変性の実現に継承・委譲・ポリモフィズム、etc
  - 関心の分離
    - アスペクト指向
    - アーキテクチャの階層化
    - etc
  - ドメインエンジニアリング
  - コンポーネント技術
  - パターン

プロダクトライン・エンジニアリングの再利用戦略３

- コントラクトとコンテキストの明確化
  - コントラクト
    - 契約に基づく開発と利用
  - コンテキスト
    - 提供側・利用側の非機能的、開発の制約、状況
- 従来のコンポーネントの再利用は、単純なコントラクトのみが強調されている
  - 機能（責務）
  - インターフェース
  - 事前条件、事後条件
- ソフトウエア部品は下記のことも大きな提供側のコントラクト/利用側のコンテキスト条件である
  - リアルタイム性
  - 動作時のタスク優先度
  - リエントラント性
  - コードサイズ
  - インターオペラビリティ的動作保障
  - その他利用時の制約

コントラクトとコンテキストとインターオペラビリティ



再利用を対象とした各ソフトウエアコンポーネントが異なる製品で開発された場合、そのような開発環境、テスト環境で実施されたかが大変重要な意味をもつ。場合下記の制約は誰が保証するのか？繋げたときに動作保障は？具体的には各コンポーネントの

- ・リアルタイム条件は？
- ・利用しているタスク（スレッド・プロセス）のメカニズムは？
- ・動作時のタスク優先度は？
- ・リエントラント性のメカニズムと場所は？（クリティカルセクションなど）
- ・コードサイズの制約は？
- ・インターオペラビリティ的動作保障の方法は？
- ・その他利用時の制約は存在するのか？

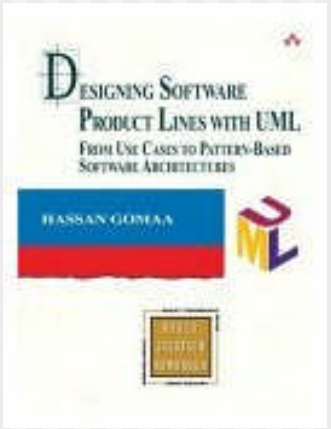
スコープの限定による再利用（ファミリー内）

プロダクトライン・エンジニアリングの手法・方法論

1. PLUS
2. PuLSE
3. FODA、FORM

ＰＬＵＳ１

- PLUS(Product Line UML-based Software Engineering)
  - Hassan Gomaa(George Mason Univ.)
  - DARTS、ADARTS,CODARTS、COMETなどリアルタイムシステムで貢献で有名
  - エンジニアリング(要求定義・分析・設計）領域に特化している
    - 「アーキテクチャ定義」
    - 「コンポーネント開発」
    - 「要求エンジニアリング」の３つ
  - PLUSは、フィーチャ、コンポーネント、クラスをよく定義された手順に従って発見し、構造化することを支援する手段を提供
  - 既存資源からの発掘やCOTSの利用法も想定



ＰＬＵＳ２

- ＰＬＵＳのカバーする範囲
  - アーキテクチャ：コンポ型、メッセージ型を使って提供
  - コンポーネント：ユースケース記述をもとに抽出
  - 要求エンジニアリング：
    - ユースケース⇒フューチャ⇒コンポーネント
  - 構造、方法論、テスト手段（ただしこれは弱い）をサポート
  - ユースケースから機能要求をカバー
    - ただし非機能要求をカバーしていない
  - 構成管理
  - プロジェクト管理、メトリックス、追跡
  - スコープの定義
    - コアアセットの特定（ただしこれは軽い扱い）
  - 組織管理プラクティス
    - 調達戦略
    - マーケティング
    - 組織編成など（弱い）
  - 市場、コスト見積技術など

PLUS手順	目的	インプット	アウトプット
スコーピング	プロダクトラインの開発対象の範囲を設定。	問題領域（マーケティングプラン）	採用する変種
ユースケースモデリング	プロダクトの典型的な使われ方を分析。	問題領域	ユースケース図と記述（変化点付き）
		採用する変種	
フィーチャモデリング	ユースケースと変化点をフィーチャーでまとめる。	ユースケース図と記述（変化点付き）	フィーチャー一覧
			フィーチャーユースケース関係表
			フィーチャー依存図
静的モデリング	システムの静的な構造をモデル化する。	問題領域	クラス一覧
	組み込みでは、主にデバイスとの関係をモデル化。	ユースケース図と記述	概念静的モデル
	必要に応じてエンティティ導子の関連を分析		コンテキストクラス図
			エンティティクラス図
動的相互作用モデリング	外界と内部オブジェクトの通信をモデル化。	ユースケース記述	コミュニケーション図
		フィーチャー一覧	
		コンテキスト図	
		エンティティクラス図	
状態機械モデリング	状態を持つオブジェクトをモデル化。	コミュニケーション図	ステートチャート図
フィーチャー／クラス依存関係モデリング	フィーチャーとクラスの対応表を作成。	フィーチャー一覧	フィーチャー／クラス依存関係表
		フィーチャー・ユースケース関係図	フィーチャーでタグ付けされたコミュニケーション図
		クラス一覧	フィーチャーでタグ付けされたステートチャート図
アーキテクチャ設計	分析結果に最適なアーキテクチャを設計	コミュニケーション図	コンポーネント構造図
			メッセージインターフェース図
			平行コミュニケーション図
			アーキテクチャ図
コンポーネント設計	各コンポーネントのインターフェースを設計	アーキテクチャ設計の出力	コンポーネントインターフェース図

ＰｕＬＳＥ

- PuLSE（Product Line Software Engineering）はIESEで開発された
  - プロダクトライン開発に必要な活動、基盤を体系化
    - 基盤（infrastructure）：コア資産
    - 技術ンポーネントを6つに分類している
      - プロダクトラインの展開を行なうために必要となる技術的なノウハウを提供している
      - PuLSE－BC（Baselining & Customaization）
      - PuLSE－Eco （Economic Scoping）
      - PuLSE－CDA （Customizable Domain Analysis）
      - PuLSE－DSAA （decision model）
      - PuLSE－I （Instantiation）
      - PuLSE－EM （Evolution & Management）

FODA

- FODA（Feature-Oriented Domain Analysis:機能（特性）ドメイン分析）
  - Kwanwoo Lee, Kyo Chul Kang他
  - ドメイン分析のプロセスとプロダクトを特定・定義する
  - ドメインのアプリケーション(製品)で、ユーザーが共通して求めている機能と可変性を検討する
  - 分析作業以降は他のドメイン、ドメインに共通する機能と派生機能を明確にし、モデルとして表現する
- FORM（A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures）

参考文献

- Designing Software Product Lines With UML: From Use Cases to Pattern-Based Software Architectures ：Hassan Gomaa , Maryann Barber（著）Addison-Wesley Pub
- 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（著）, 前田 卓雄（翻訳）日刊工業新聞社
- ソフトウェアファクトリー―パターン、モデル、フレームワーク、ツールによるアプリケーションの組み立て：Jack Greenfield 他著 野村 一行（翻訳）日経BPソフトプレス
- マルチパラダイムデザイン：ジェームズ・O. コプリン（著）, James O. Coplien（原著）, 金沢 典子 , 羽生田 栄一 , 平鍋 健児（翻訳）ピアソンエデュケーション
- Kawanwoo Lee,Kyo C.Kang, and Jaejoon Lee Concepts and Guidelines of Feature Modeling for Product Line Software Engineering
- Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe, FORM：Feature-Oriented Product Line Engineering, IEEE Software, Vol.9, No.4, pp.58-65
- Axel van Lamsweerde and Emmanuel letier, Integrating Obstacles in Goal-Driven Requirements Engineering. In Proc. Of 20th International Conference on Software Engineering, pp. 53-63, 1998