ソフトウェア開発方法論とプロジェクト管理の融合法に関する考察

北陸先端科学技術大学院大学
情報科学研究科
落水　浩一郎

内容

1. ソフトウェアプロセスモデル、ソフトウェア開発方法論、プロジェクト管理技法の発展（または変化）をサーベイしつつ
2. 統合のいくつかの視点に関する話題を提供する

問題提起

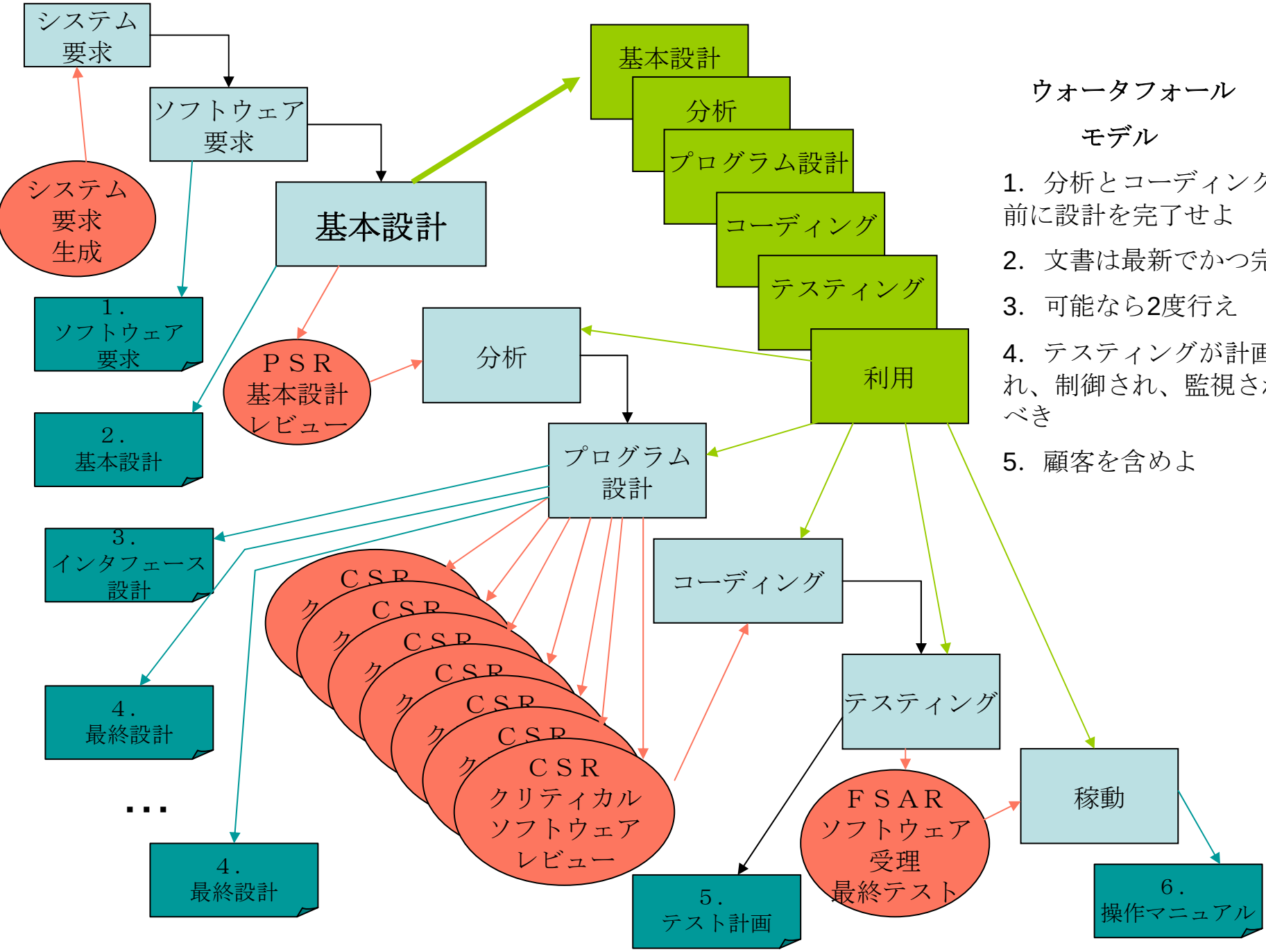
- ソフトウェアプロセスモデル
  - ソフトウェア開発の**戦略**を規定する
- ソフトウェア開発方法論
  - ソフトウェアの持つべき望ましい「**構造**」を定義し、それらを作りこむ手段を検討する。
  - 「構造」の属性の例：正しさの確認容易性、変更波及の局所化、作業分担の容易性、再利用容易性、進化容易性など
- プロジェクト管理
  - プロジェクトチームのコスト、スケジュール；プロダクトの品質などに関して、それらを**計画・見積り、資源を割当、状況を監視**（メトリクス）し**制御**（ベストプラクティス）する手段を提供する
- 乖離？　プロジェクト毎のカスタマイズが困難
- 融合の視点は？

ソフトウェアプロセスモデルソフトウェア開発方法論、プロジェクト管理技術の発展の歴史（本文付録１参照）

- プログラミング方法論の提案(1970年代前半)
- 設計方法論の開発(1970年代後半)
- 要求定義技術の開発(1970年代後半)
- 定量的プロジェクト管理のはしり(1970年代後半から1980年代前半)
- システム工学の導入によるWaterfall Model の改善(1980年代中期から後半)
- 反復型Waterfall Model の提案(1980年代前半)
- プロトタイピングの出現(1980年代前半)
- 実行可能仕様とフォーマルメソッド(1980年代中期)
- プロセスプログラミング(1980年代後半)
- プロセス改善技術(1990年代初頭)
- CASEツール(1990年代初頭)
- アーキテクチャ中心開発(1990年代中期)
- オブジェクト指向技術（ 1980年代以降）
- ソフトウェアアセスメント技術の成熟（1990年代後半）
- UML (1990年代後半)
- 反復型プロセスモデル（2000年代）
- アジャイル（2000年代）
- ゴール指向要求工学、統合要求工学、COTS（最近）

ソフトウェアプロセスモデルの変化

- Waterfallモデル
  - 契約駆動、内製
- Vモデル（システム工学の導入）
  - 外注
- ミニWaterfallによる反復
  - リスク管理
- プロトタイピング
  - ユーザの取り込み
- 反復型モデル（イテレーティブ＆インクリメンタル）
  - 不確実さの減少、プロジェクト特性の学習



ウォータフォール モデル

1. 分析とコーディングの前に設計を完了せよ
2. 文書は最新でかつ完全
3. 可能なら2度行え
4. テスティングが計画され、制御され、監視されるべき
5. 顧客を含めよ

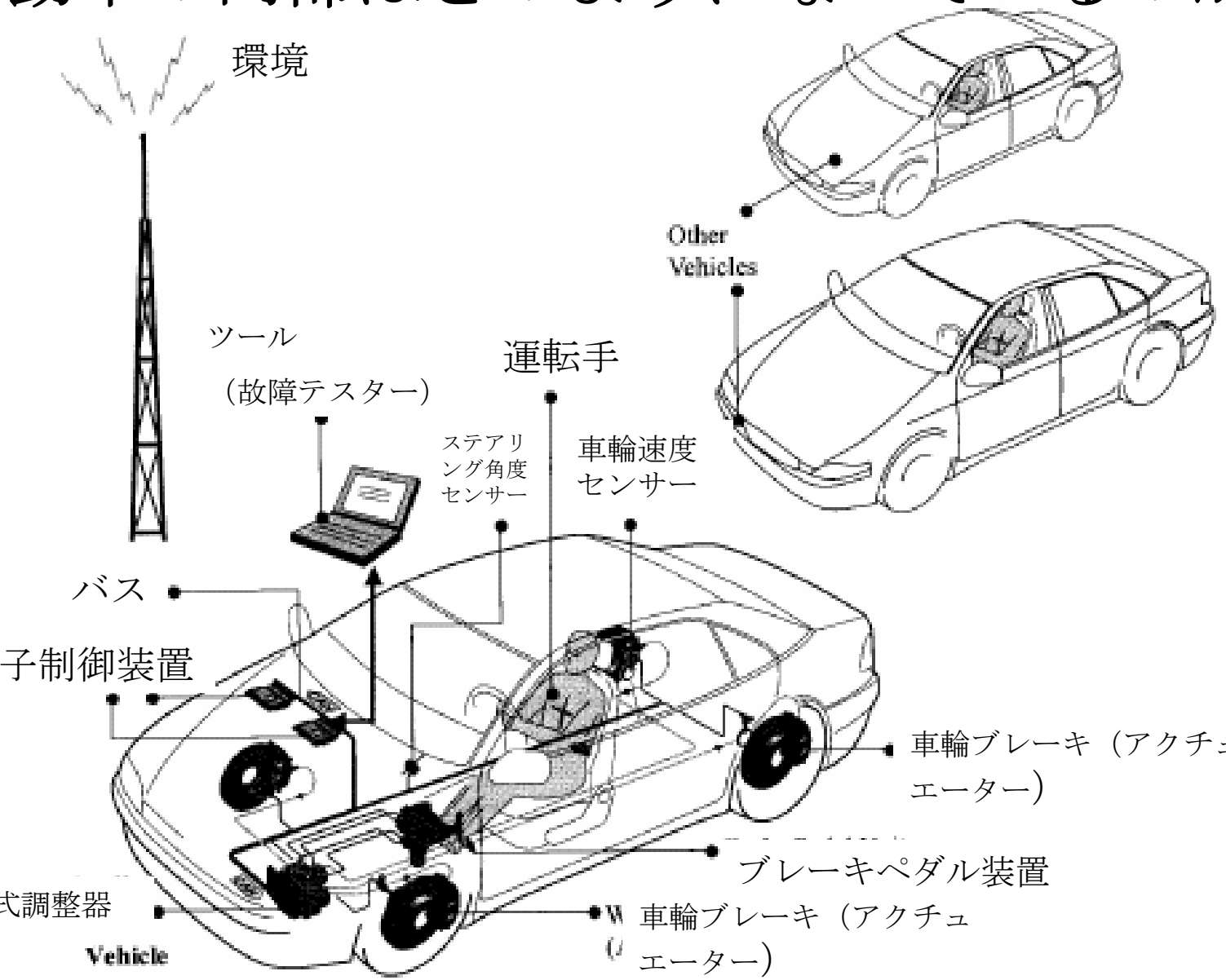
ウォータフォールモデル構築に至る論理の展開

- モデルの構築
  - （プログラム）「分析」と「コーディング」から始める
  - 大規模ソフトウェア開発を制御するために必要なフェーズ群を付加する　（システム要求定義、ソフトウェア要求定義、プログラム設計、テスティング、稼動）
  - フェーズの実行順序に関する情報を付加し、ベースラインの移行法に関する仮定を導入する
- 問題点の指摘と対応策の提案
  - 「テスト」→「設計」→「要求」となる、大幅な手戻りに関する問題の指摘と対応策の提案
    - 基本設計フェーズの導入
    - パイロットモデルの開発（基本設計以下を2度行う）
    - 設計結果の文書化
    - テストの計画、制御、監視
    - 顧客を含める

Vモデル

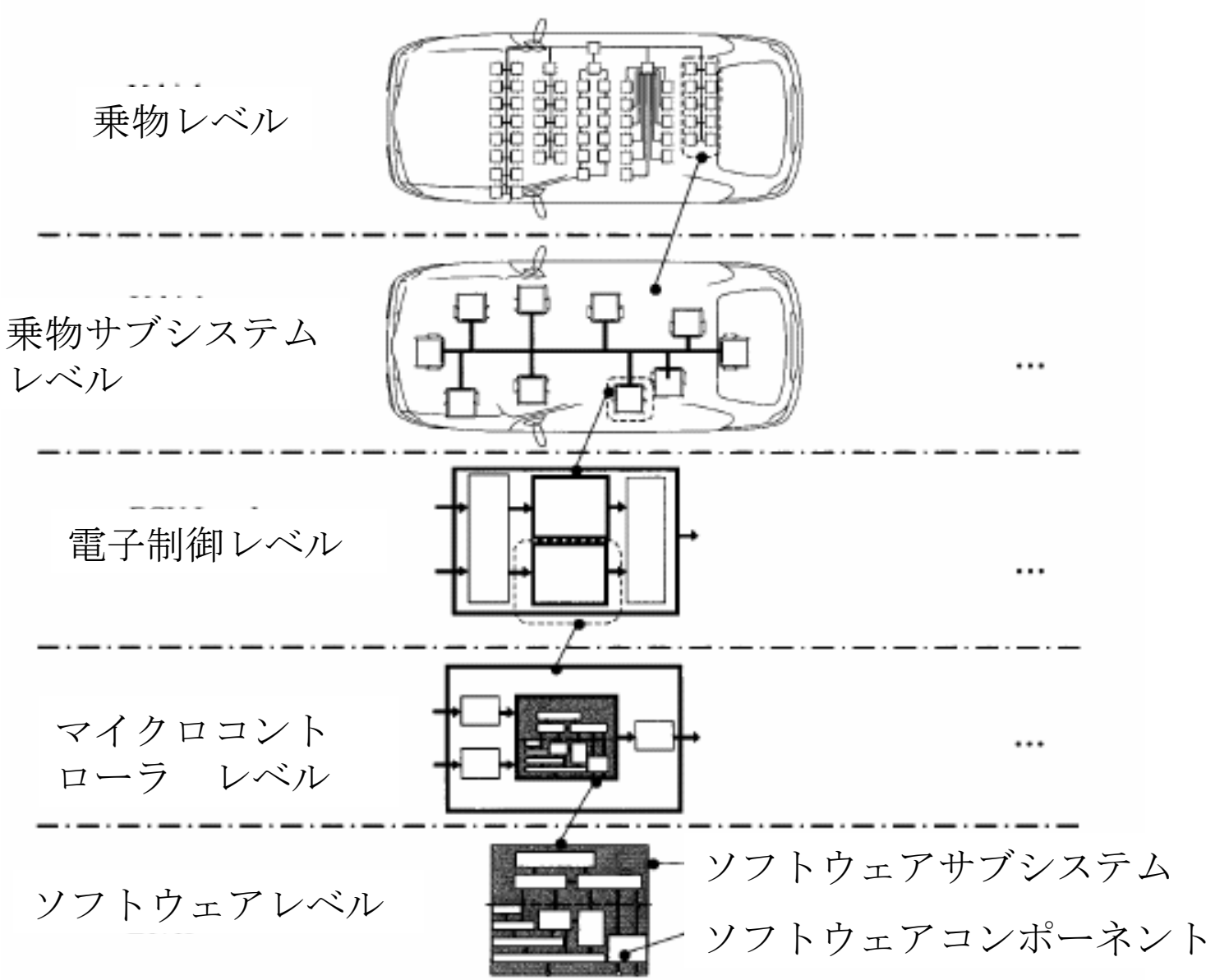
- システム工学的アプローチの導入
  - システム要求を定義する
  - システム要求をサブシステムに配分する
  - 詳細なコンポーネントを定義する
  - コンポーネントを検証する、サブシステムを検証する

自動車の内部はどのようになっているのか



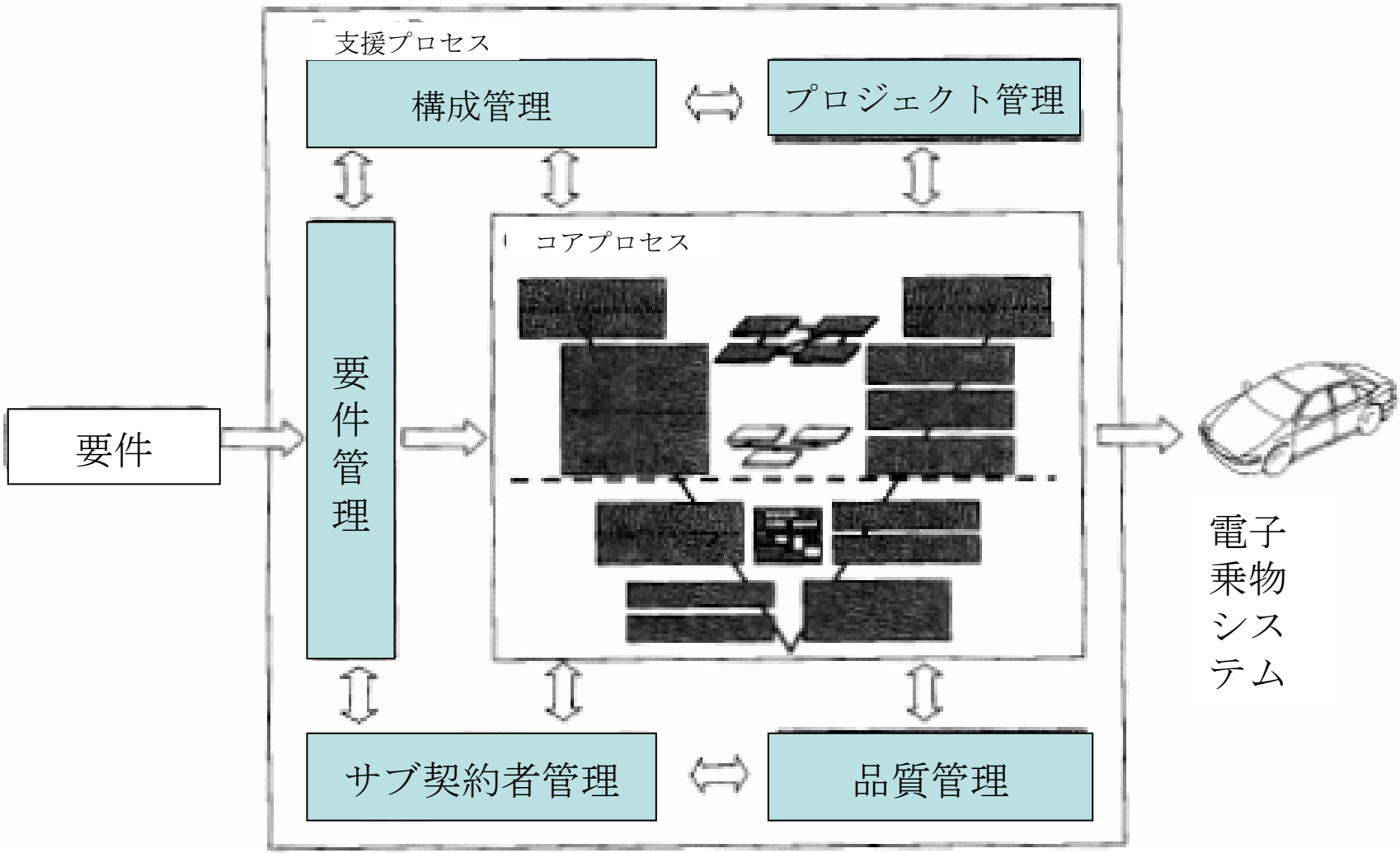
Jorg Schauffele,　Thomas Zurawka, “Automotive Software Engineering”, SAE Intnl. 2005.

電気・ソフトウェア系の設計



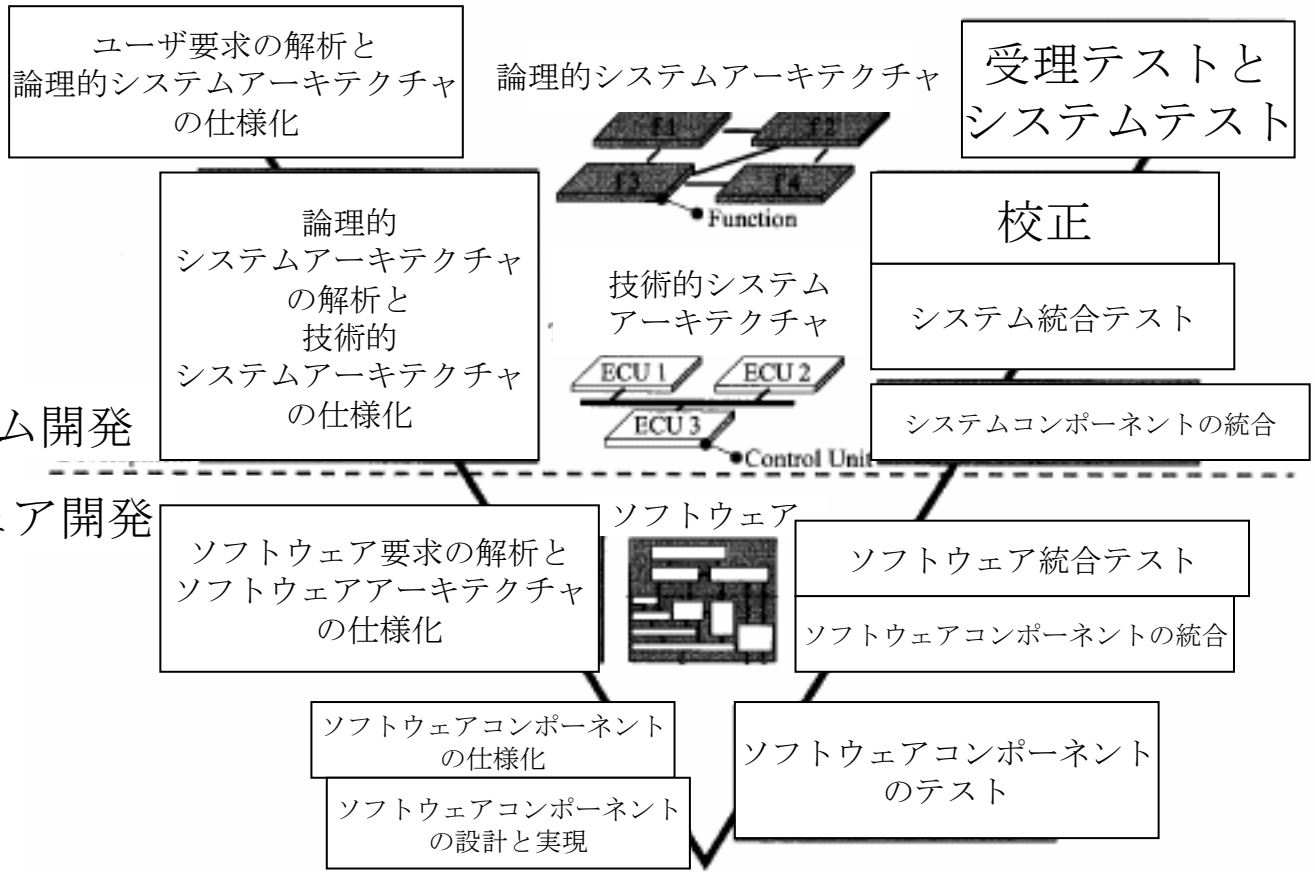
Jorg Schauffele,　Thomas Zurawka, “Automotive Software Engineering”, SAE Intnl. 2005.

車載システム開発法（BMWの例）



Jorg Schauffele,　Thomas Zurawka, “Automotive Software Engineering”, SAE Intnl. 2005.

車載システム開発へのVモデルの適用

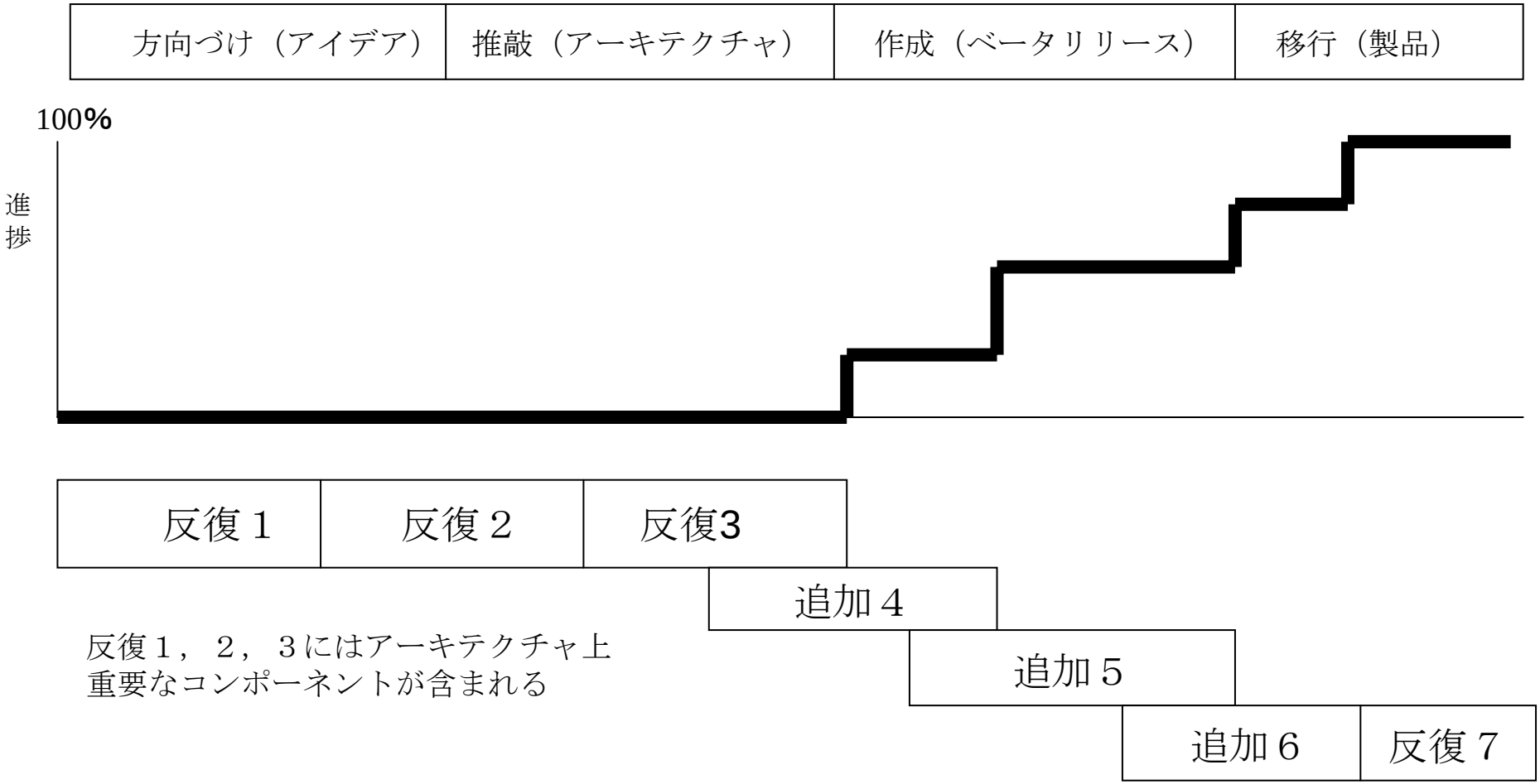


Jorg Schauffele,　Thomas Zurawka, “Automotive Software Engineering”, SAE Intnl. 2005.

反復モデル

- いくつかの問題を早期に洞察すればするほど、それを修正できる可能性がましていくというのが大規模システム構築における基本原則である（カンター）。
- プロジェクト開始時には、向かっている目標がわからない。インクリメンタルな開発は、一度開発サイクル全体を経験させ、新しく発見した知識をすぐ使えるようにする。プロジェクトには突発事項が待ち構えている。インクリメントによって、そのような突発事項に早く気付くことができ、各インクリメント境界で作業方法を変更する機会が得られる（コーバーン）。

統一プロセス



反復１，２，３にはアーキテクチャ上
重要なコンポーネントが含まれる

反復７では新しいコンポーネントを追加せず、アップグレード、修正、機能拡張のみを行う

Walker. Royce著、日本ラショナルソフトウェア監訳、「ソフトウェアプロジェクト管理」。ピアソン・エデュケーション、2001.

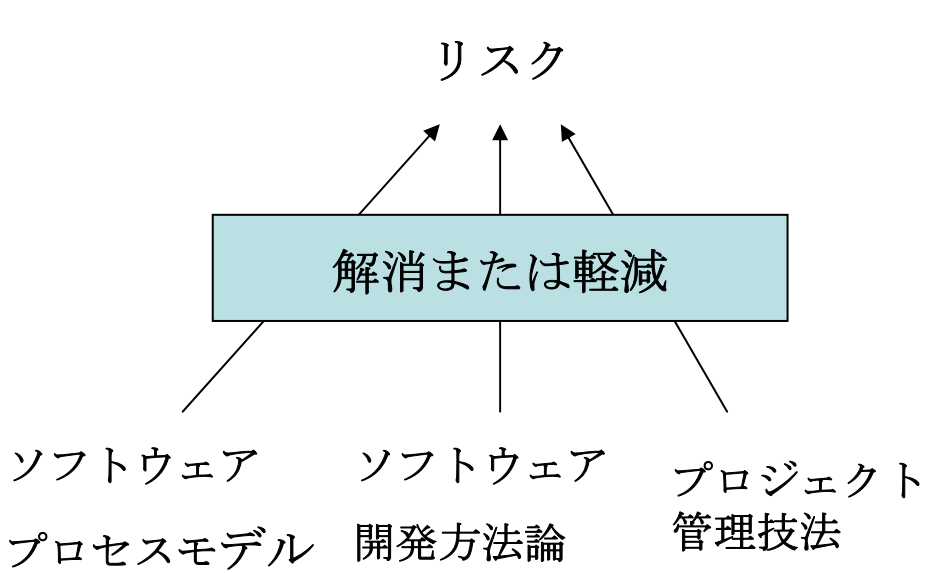
ソフトウェア開発方法論の発展の歴史

- 構造化プログラミング
  - 正しさの確認、作業の独立性
- 情報隠蔽モジュール
  - 変更波及の局所化
- 構造化分析・設計
  - 変更波及の局所化
- 要求定義
  - ユーザ要求の記述法
- 実行可能仕様とフォーマルメソッド
  - 自動化、検証、証明
- オブジェクト指向
  - 変更波及の局所化、再利用、拡張容易性
- ゴール指向要求工学，統合要求工学，COTS
  - 納期短縮

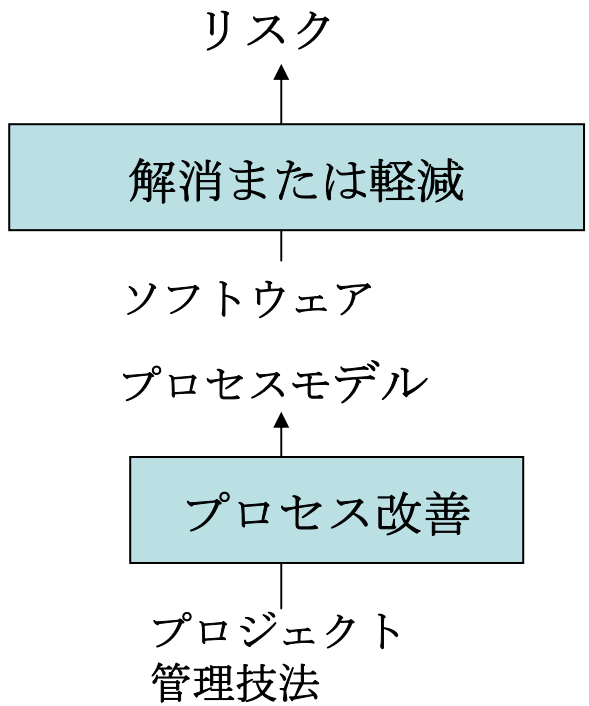
プロジェクト管理技術の発展

- 種々のメトリクス
  - コスト見積り、危険要因の検出、テスト打ち切り時期の判定
- 測定手段
  - ファンクションポイント
- CMM
  - 成熟度とベストプラクティス
- ソフトウェア評価
  - ベースラインとベンチマーク
- PMBOK
  - 知識体系の整備

リスクによる統合の例
（本文　付録２、付録３参照）



アプローチ**A**



アプローチ**B**

アプローチA

（**Caper Jones,** ソフトウェア病理学、1995）

- 委託開発あるいはアウトソーシングソフトウェアのリスク
  - 高い保守コスト　60%
  - 受託開発企業と顧客の軋轢　50%
  - 徐々に増大するユーザ要求　45%
  - 契約外の検収基準　30%
  - ソフトウェアおよび成果物の法律的な所有権　20%

アプローチA
（Caper Jones, ソフトウェア病理学、1995）

- 高い保守コスト
  - 構造的な複雑度解析ツール
  - リストラクチャリングツール
  - リバースエンジニアリングツール
  - 欠陥多発モジュールの解析と欠陥除去手法
- 受託開発企業と顧客の軋轢　管理しにくい
- 徐々に増大するユーザ要求
  - プロトタイピングの利用
  - JADの採用
  - 契約の基本を、納入されるべき成果物のファンクションポイント数とファンクションポイントあたりの予想コストにおく
- 契約外の検収基準　言及なし
- 法律問題と訴訟問題　管理しにくい

アプローチB

（Walker Royce, ソフトウェアプロジェクト管理, 2001）

- **アーキテクチャ先行アプローチ**　開発プロセスをアーキテクチャ先行アプローチに基づいて構築する。設計を推進する要求事項、アーキテクチャにかかわる重要な設計上の決定などを、リソースをフルスケールの開発にあてる前に達成する。
- **反復型ライフサイクルプロセス**　問題についての理解、効果的なソリューション、効果的な計画をいくつかの反復にわたって拡大する反復型プロセスを採用する
- **コンポーネントベースの開発**　ソースコードとカスタム開発の量を減らす
- **変更管理環境**　共有の成果物に基づいて作業を進めるための客観的に管理されたベースラインを設ける
- **ラウンドトリップエンジニアリング**　要求仕様、設計モデル、ソースコード、実行可能コード、テストケースなど、異なる形式の開発情報を自動化し同期を取るのに必要な支援環境を利用して変更の自由度を向上させる
- これらが、従来の開発プロセスが持つ上位１０のリスクを次のように解消するとする

融合にあたっての考慮点(1/3)

- スキルレベルが異なる様々な人間集団を異なる組織にわたって協調させるためには、まず、適切なソフトウェアプロセスモデルを採用する必要がある。
- 人間集団の動きを統一するだけでは品質は作りこめない。目標とする品質を作りこむためには、開発手段の標準化が必要であり、適切なソフトウェア開発方法論を採用する必要がある。

融合にあたっての考慮点(2/3)

- 人間集団の動きを統一し、開発手段を標準化しても問題は残る。
- ソフトウェア開発でつねに問題になるのは、コスト超過、スケジュール遅延、品質の不十分性であり、これらは開発プロジェクトに依存する。

融合にあたっての考慮点(3/３)

- 開発の進行につれて見積りや計画段階では読みきれなかった事柄が顕在化してくる（状況）。これらの状況を検知し（学習し）、必要な対応策を施すため、プロジェクト管理技術の適切なサブセットの選択が必要となる。
- これらは、特定のソフトウェアプロセスモデルやソフトウェア開発方法論のもつ弱みを補うものであり、かつ、プロジェクトチームが抱える問題点を補うものでなければならない。

融合にあたっての問題点

- 最新の話題となっているものを単純に組合わせるだけでよいのなら
  - 反復型ソフトウェアプロセスモデル
  - オブジェクト指向やコンポーネント指向ソフトウェア開発方法論
  - PMBOK中の適切なプラクティス
- 話は簡単であるが実際にはそうはいかないだろう

ソフトウェアプロセスモデルはそれぞれ押さえどころが異なる

- Vモデルは外注を考慮にいれたモデルであるがウォータフォールモデルの欠陥を引き継いでいる
- 反復型プロセスモデルは、ウォーフォールモデルの欠陥をカバーする。しかし、アウトソーシングやオフショア開発に適するか否かは不明である。

ソフトウェア開発方法論とソフトウェアプロセスモデル

- 特定のソフトウェア開発方法論は、それが開発された時代に支配的であったソフトウェアプロセスモデルに基づいて構築されていることが多い
- 特定のソフトウェア開発方法論の採用は特定のソフトウェアプロセスモデルの採用を意味する

プロジェクト管理技術とソフトウェアプロセスモデル

- プロジェクト管理技術の中には特定のソフトウェアプロセスモデルに依存するものがある
  - 例えば、PMBOKのタイム管理はウォータフォールモデルに依存する要素が多い

特定のプロジェクトへの依存

- 開発の進行につれて顕在化するリスクやその度合いは特定のプロジェクトに依存するように思える。このため、どのようなプロジェクト監視技術を重点的に採用するかはプロジェクト毎に異なるように思える

とりあえずの攻め口
組合わせを特徴づけるパラメータは？

- ソフトウェアプロセスモデル
  - 市場の特性の分類
  - 顧客と受託会社の関係
  - 外注、内製
- ソフトウェア開発方法論
  - ドメインの特性
  - 開発者担当者の技術レベル
- プロジェクト管理技法
  - 開発に利用できるリソース量
  - 開発者の技術レベル
  - データ収集のタイミングと内容の分類、
    結果のフィードバック法の分類