XDDP と SPLE の概略

研究会T-14「XDDPとSPLの連携」

SIG-XAS

トピック

1. 派生開発
   - 定義、開発の様子、典型的な状況
2. XDDP
   - 開発の様子、構成要素、プロセス、機能追加と変更、成果物の3点セット、まとめ
3. SPLE
   - 基本形、事業との関係、可変性、フィーチャ、フィーチャモデル、フィーチャとコア資産の関係、プロセスと組織

派生開発

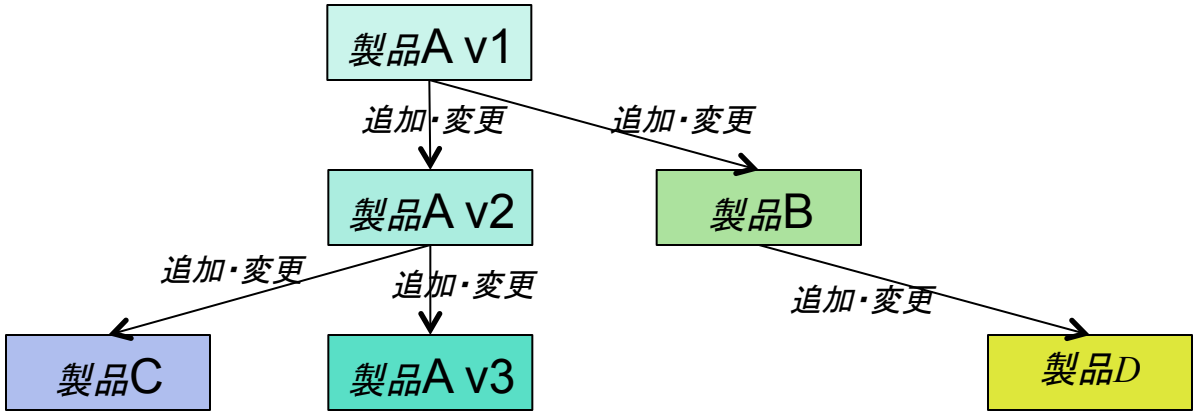
派生開発: 定義

- 派生開発とは
  - 差分開発、流用開発、改良保守、改造開発などとも呼ばれる
  - 正式な定義はない
  - ここでは以下の開発形態を派生開発と呼ぶこととする

過去に構築した、完成済みまたは完成途中のソフトウェアシステムを、一部変更するか、または拡張するか、またはその両方を行なう開発の形態

派生開発の様子

派生開発の典型例



説明:

A, B, C, D: 製品の種類

v1, v2, v3: 同一製品の異なるバージョン

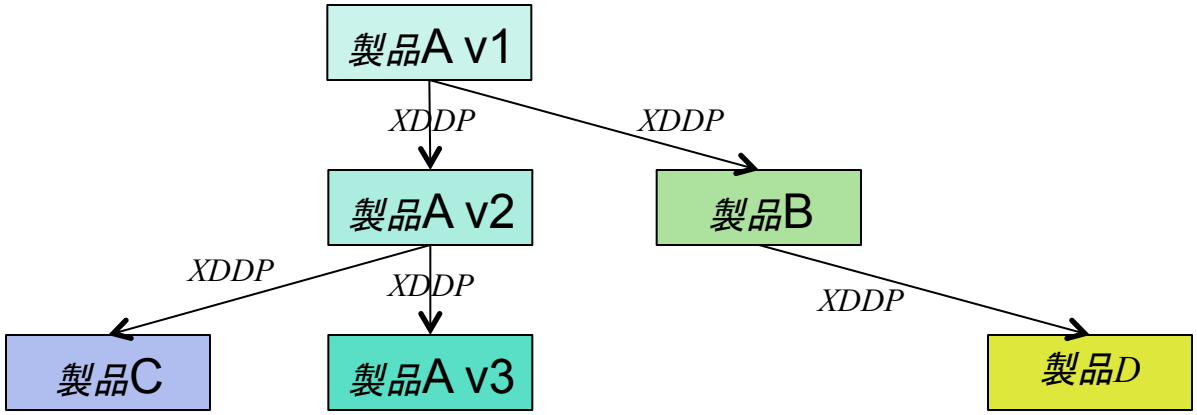
派生開発の典型的状況

- 要件
  - 変更が中心で、機能追加を伴うことがある
  - いま稼動しているシステムに手を加えることになる
- 期間
  - 2週間～6ヶ月と幅はあるが、通常は短いと認識されている
- 制限
  - 今回の担当者が書いたソースコードではないことが多い
  - ソースコードを理解するには、そこにある設計書の内容は不足している
  - ときには、まともな機能設計書も各段階の設計書もない
  - これまでの派生開発のなかで、当初の設計思想は崩されていることが多い
  - 時間的にも技術的にも、「全部を理解」してとりかかることはできない

XDDP

XDDP による開発の様子

XDDP の典型例



説明:

A, B, C, D: 製品の種類

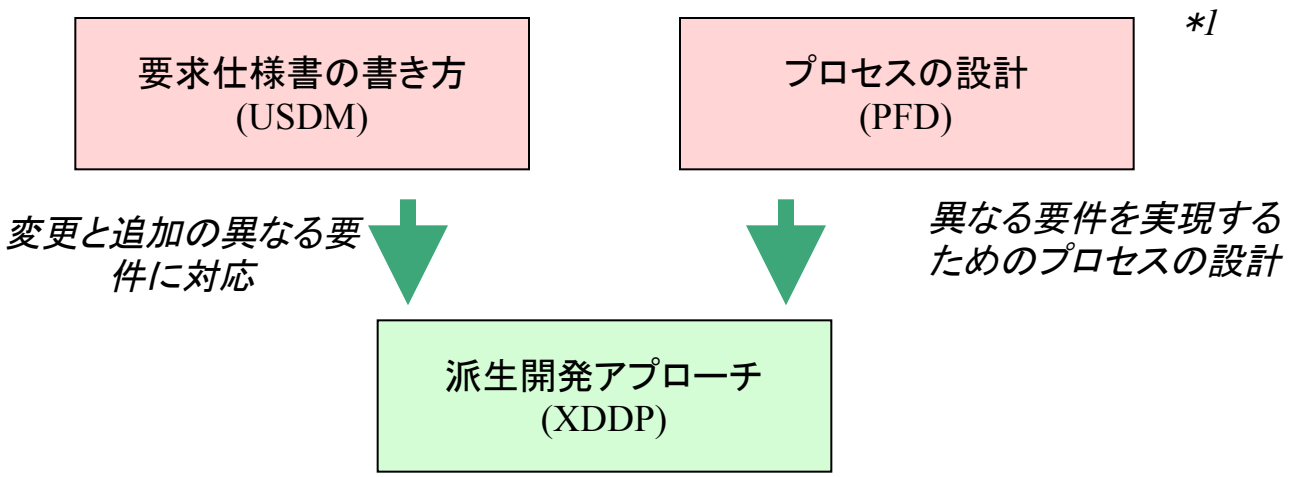
v1, v2, v3: 同一製品の異なるバージョン

XDDP

- XDDPは、最小構成の成果物とプロセスからなる、派生開発専用のプロセスである
- 派生開発では、プログラム上で変更箇所と思える場所を見つけたと思っても、以下のような理由から不具合が発生しやすい
  1. 最適な変更箇所では無かった
  2. 関連する箇所が他にもあることに気が付かなかった
  3. 勘違いをして、間違えた修正をしてしまった
- XDDPでは、以下の事柄を明確にしてからコーティングを行なう
  1. What: 変更要求仕様書で、何を変更するのか？　どういう仕様をどのように変更するのか
  2. Where: トレーサビリティマトリクス（TM）で、変更要求に対して、どこを変更しようとしているのか
  3. How: 変更設計書で、具体的にどのように変更するのか

XDDP: 構成要素

- XDDPは、前述のような特徴がある派生開発に対して、「機能追加と変更とでプロセスを使い分ける」という、アプローチをとっている



*1: [Shimizu09] p.24に基づく。

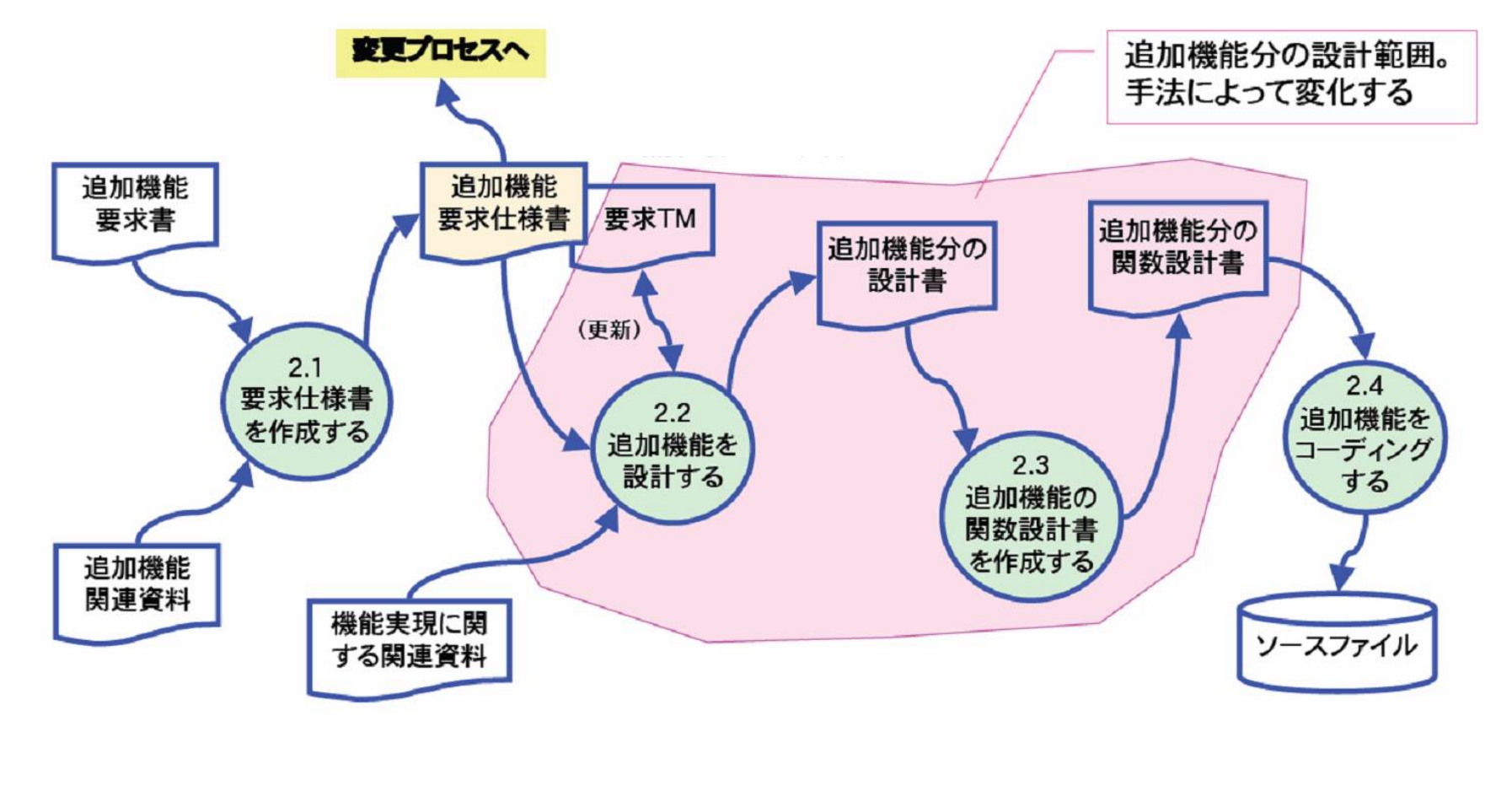
XDDP のプロセス

- 派生開発の要求は、「変更要求」と「機能追加の要求」の2種類がある
- 「変更」と「機能追加」とでは要求の性質が大きく異なるため、XDDPでは「変更用プロセス」と「機能追加用プロセス」に分けている
- 各プロセスは、Process Flow Diagram (PFD) であらわす
- PFDは、構造化分析で使用するData Flow Diagram を、プロセスと成果物の連携を表現しやすいように、改良したものである
- PFDは、楕円でプロセスを表し、書類の記号で成果物を、また矢印で、プロセスへの入力や出力を表す。PFDは、「PFD (Process Flow Diagram) の書き方 第3版」(*1)に詳細な説明がある
- プロジェクト毎の異なる要求を実現するために、以降で示す「変更のPFD」や「機能追加のPFD」を元に、プロジェクトに合わせてPFDを変化させる。つまり、プロセスを設計する(*2)

*1: [Shimizu09a] に基づく。
*2: [Shimizu07] p.355に基づく。

XDDP: 機能追加の PFD

- 以下に追加プロセスの PFD を示す



[Shimizu07] p.111に基づく。

XDDP: 機能追加の PFD

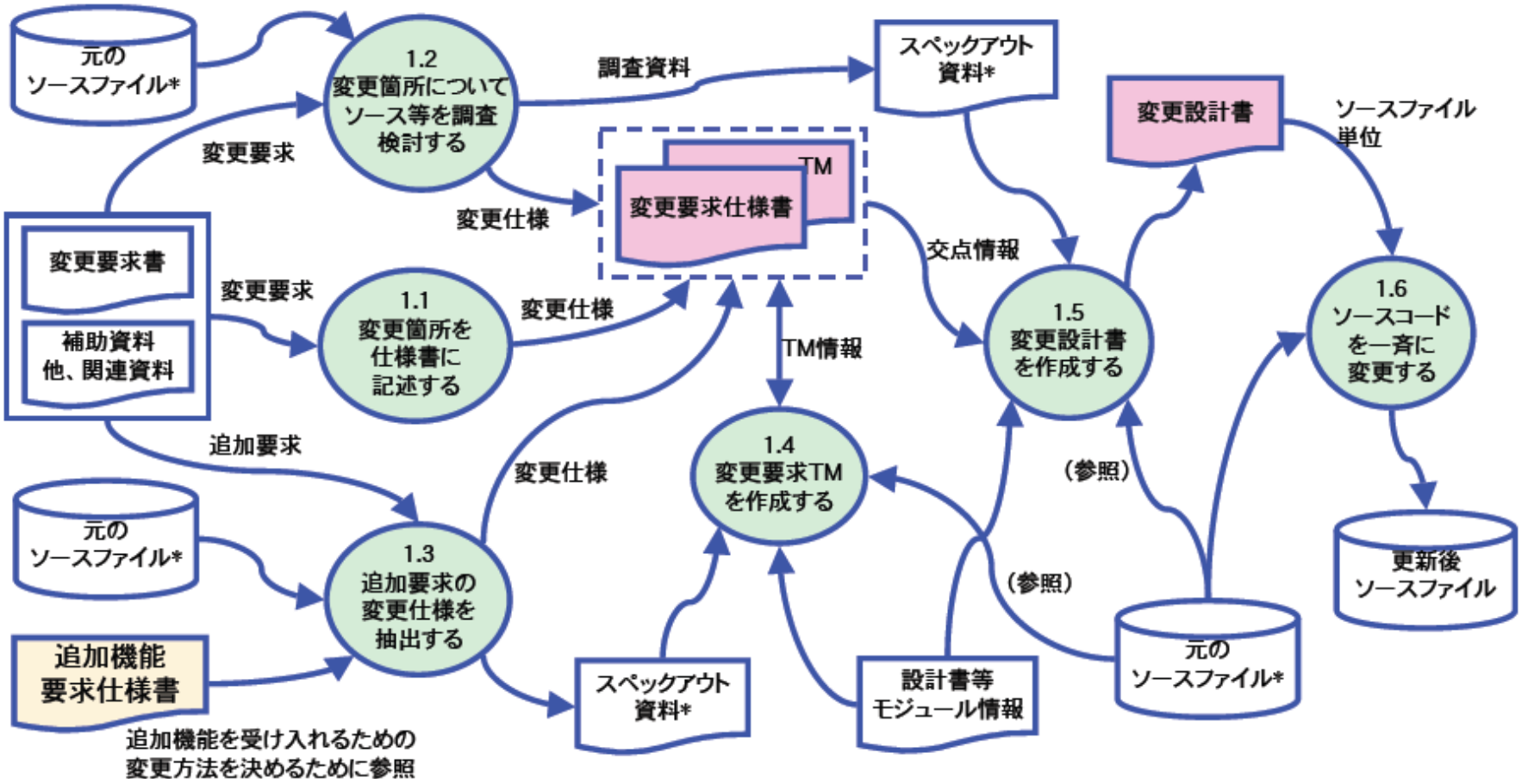
前ページのプロセスを以下に説明する

- プロセス2.1 要求仕様書を作成する
  - 案件によっては、この段階で機能の追加方法などの調査が行われる
  - このプロセスで作成する追加機能要求仕様書を変更プロセスで使う
- プロセス2.2 追加機能を設計する
  - 追加機能を設計した際に、要求TMの更新も行う
  - 既存ドキュメントで要求TMが作成されていない場合も、こうした機会に作成するとよい
- プロセス2.3 機能追加の関数設計書を作成する
  - プロセス2.2も含め、このプロセスは、手法によって変化する
- プロセス2.4 追加機能をコーディングする
  - 通常の新規開発と同様に、追加機能をコーディングしソースファイルを作成する

[Shimizu09] p.35より引用。

XDDP: 変更の PFD

- 以下に変更プロセスの PFD を示す



図中の「＊」は複製表示で一つのPFD上に同一の成果物が他にも表示されていることを示す(*1)

[Shimizu07] p.108より引用。
*1: [Shimizu09a] p.20より引用。

XDDP: 変更の PFD

前ページのプロセスを以下に説明する

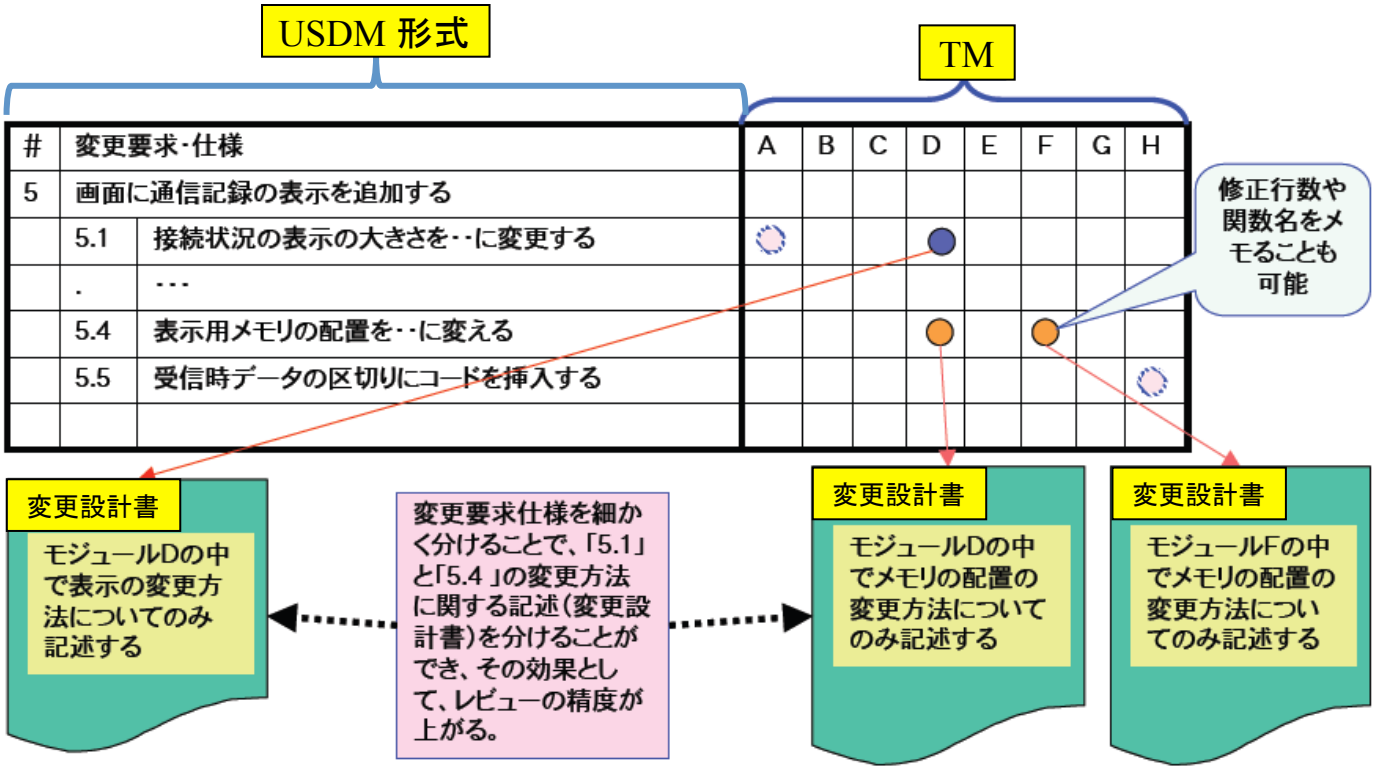
- プロセス1.1 変更箇所を仕様書に記述する
  - 機能仕様書などの文書から変更箇所（仕様）が特定され、変更仕様書の該当する箇所に記述する
- プロセス1.2 変更箇所についてソース等を調査検討する
  - それぞれの変更要求に対してソースコードを調査して変更箇所を探し、発見した箇所を変更仕様として変更仕様書の該当箇所に記述する。その際の調査資料をスペックアウト資料として残す
- プロセス1.3 追加要求の変更仕様書を抽出する
  - 追加機能を受け入れるための変更箇所（仕様）を探し、それを変更仕様書の該当箇所に記述する。ソースコードの調査結果は資料として残す
- プロセス1.4 変更要求TM(トレーサビリティマトリクス)を作成する
  - すべての変更仕様に対して具体的にソースコードとの対応付けを行う
- プロセス1.5 変更設計書を作成する
  - 変更仕様のレビューを終えた後、改めて具体的な変更方法（差分）を記述する
- プロセス1.6 ソースコードを一斉に変更する
  - 変更設計書のレビューを終えた後、ソースコードを一斉に変更する

[Shimizu09] p.35より引用。

XDDP: 成果物の 3点セット

- 従来の派生開発では以下の事柄が曖昧になりがちであった
  1. 変更要件をどのように理解したか
  2. 変更要件に対してどこを変更しようとしているのか
  3. 変更箇所をどのように変更するのか

- XDDP では以下の手段を用いることにより、関係者間での適切なレビューおよび合意形成ができるようにする
  1. 「何を」(What) 変更するかを USDM 形式の変更要求仕様書で
  2. 「どこを」(Where) 変更するのかを TM で
  3. 「どのように」(How) 変更するのかを変更設計書で具体的に



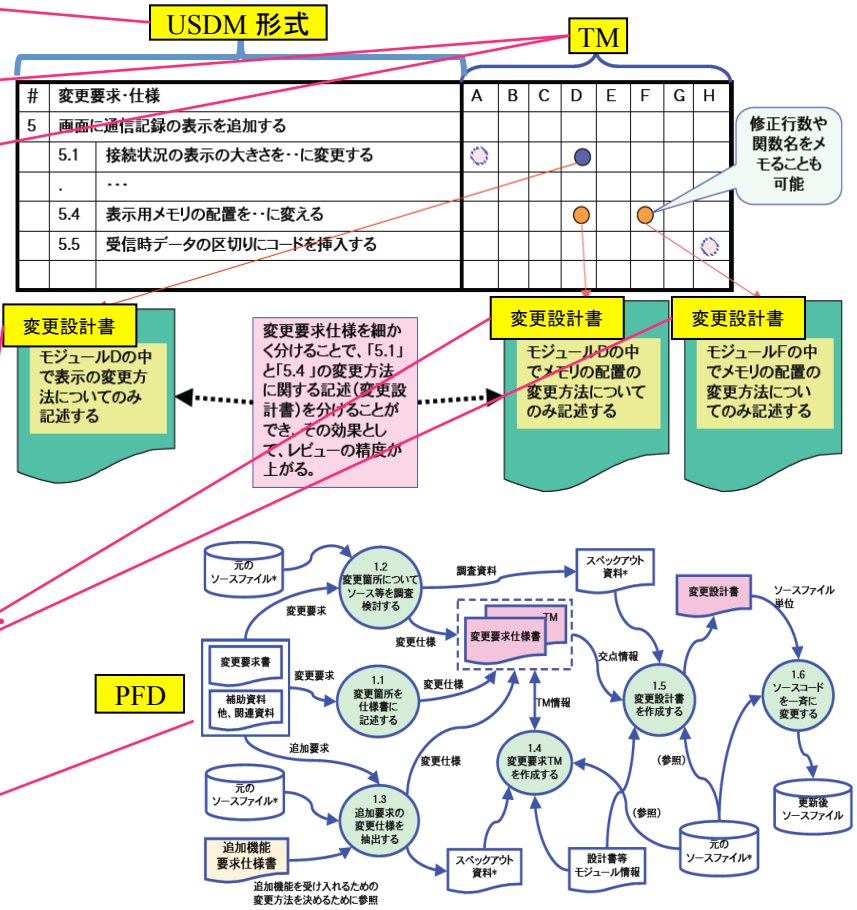
これらの「変更要求仕様書」「TM」「変更設計書」を XDDP の「成果物の3点セット」と呼ぶ

XDDP: まとめ

手戻りを起こさず最初から正しく作るために以下が用意されている

- 仕様の抜け・漏れをなくす
- 追加と変更を明確に区別する
  - 追加：
    - 単に新しく作ればよい
  - 変更：
    - ひとつの仕様に対して複数ありうる
    - 複数の仕様が同じ箇所の変更で実現されることがありうる

  そのために…
    - 変更はその箇所と内容をすべて明らかにしてからコーディングする
- そしてこれらのプロセスを状況に応じて PFD で設計・再設計する



[Shimizu09] p.73 および [Shimizu07] p.108より引用。

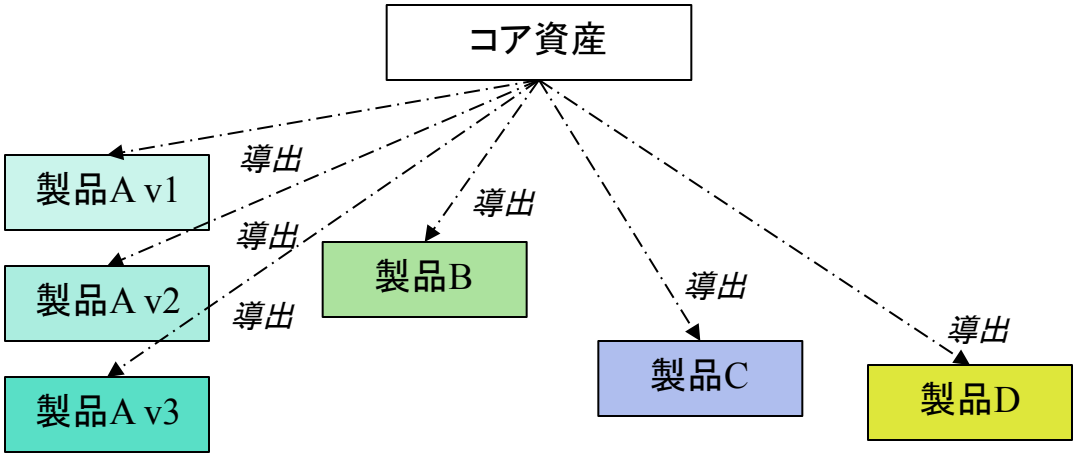
SPLE

SPLE

- Software Product Line Engineering (SPLE)
  ＝ ソフトウェアの「製品系列」(SPL) の
  作り方 (E)
- ソフトウェア再利用を体系的・計画的に行ない、
- 類似するソフトウェア集約型システム群の開発において低コスト、高品質、短納期を実現するためのパラダイム
- 「どのように」作るかだけでなく、「何を」作るかが視野に入ってくる

共通化とも

SPLEの基本形



説明:

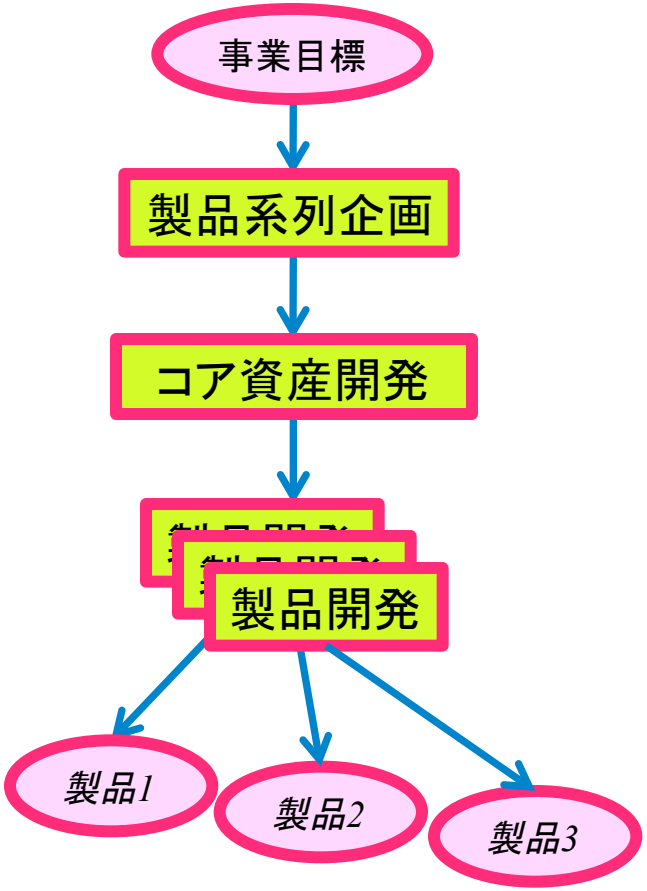
A, B, C, D: 製品の種類

v1, v2, v3: 同一製品の異なるバージョン

事業と SPLE

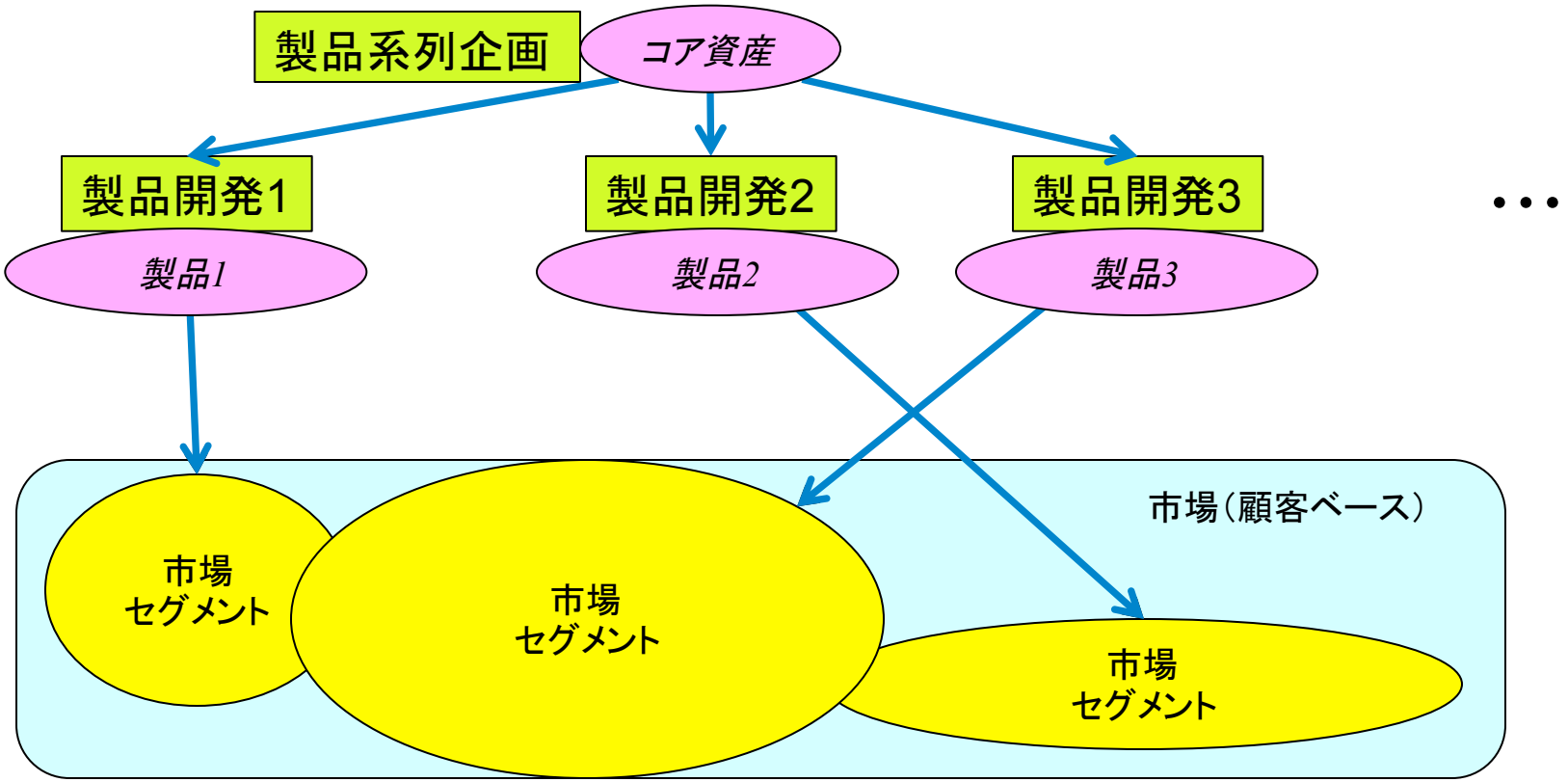
開発は二段階に分かれる

- 製品系列を企画するプロセスで対象市場を網羅する、戦略的な製品構想を確立した後、開発は二段階に分けて行なわれる
- 「コア資産開発」で製品の間の共通性と可変性（差異のある点）を正しく識別し、それら共通性と可変性を正確にソフトウェアで実現して、
- 「製品開発」で、個々の製品の大半をコア資産から再利用し、製品ごとに異なる対象市場セグメント向けの部分のみ新規開発する



事業と SPLE

- 対象市場を網羅する製品群を構想し、
- 各製品を開発する順序/時期を適切に計画し、
- コア資産を基にして製品を開発する



可変性を扱う技術

- SPLE に特有の技術というものは少ない
  - 要となるのは要件開発、アーキテクチャ設計、他いくつかの支援技術
  - 技術以外に重要なのは…
    - 超上流での市場・顧客絞り込み
    - コア資産開発と製品開発の連携
- 可変性を扱う技術は比較的特有

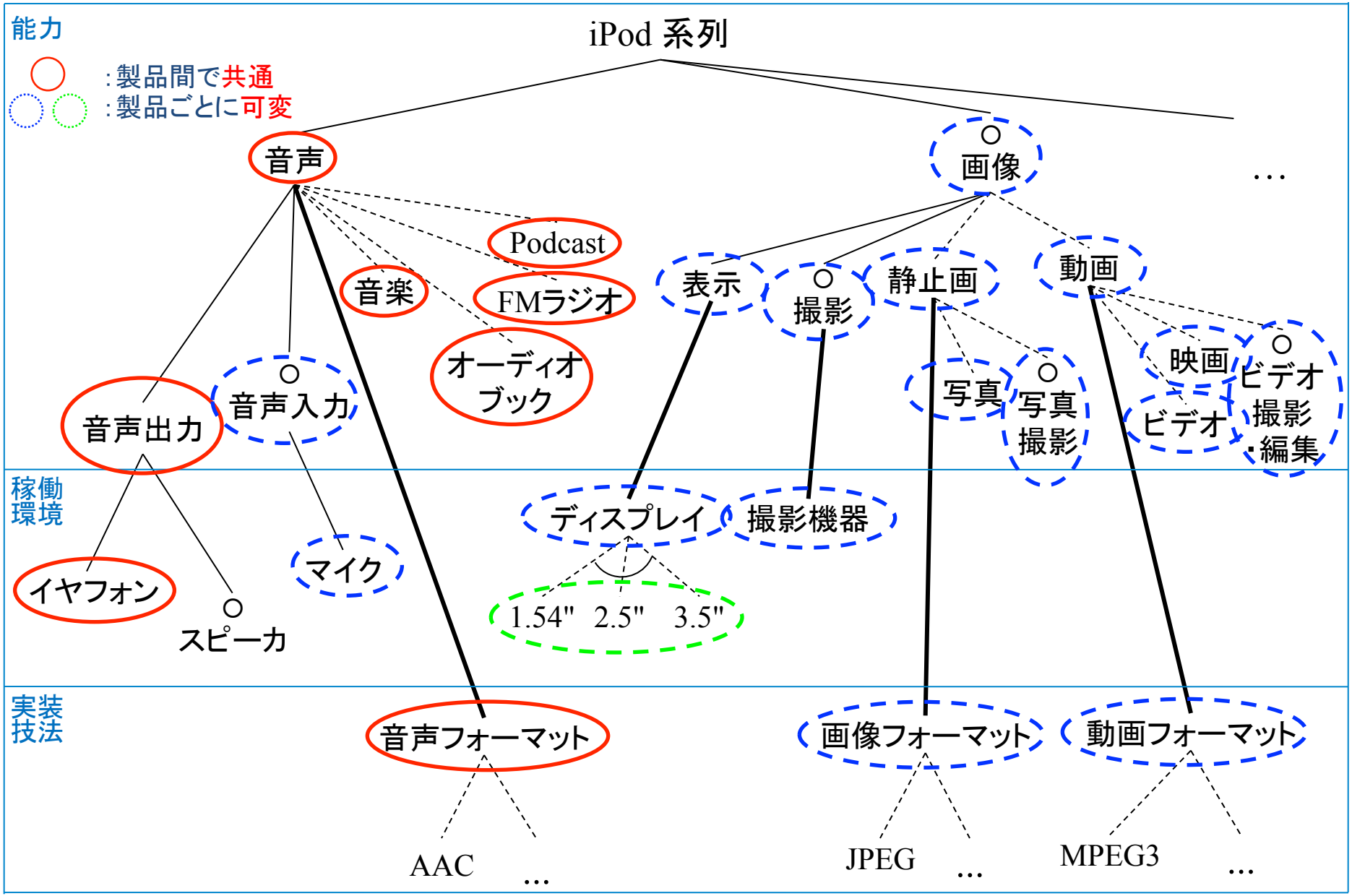
  ※可変性：製品間の差異
  - 共通性・可変性分析
  - 可変性モデリング
    - フィーチャモデリングを含む

「フィーチャ」で共通性と可変性を捉える

- SPL において何が可変かを表現するものとして、「フィーチャ」がよく使われる
  - 対象ドメインにおける専門家や末端利用者の「語彙」に相当
  - または、「マニュアルに出てくる語彙」
    - ユーザ向け/運用管理者向け/保守担当者向け/…
- 製品の利害関係者にとって意味のある**機能または品質**を一言で表わしたもの
  - 要件として展開される
  - 製品系列全体に共通のフィーチャ
  - 製品ごとに選択/非選択されるフィーチャ
  - 製品によってパラメータを調整するフィーチャ

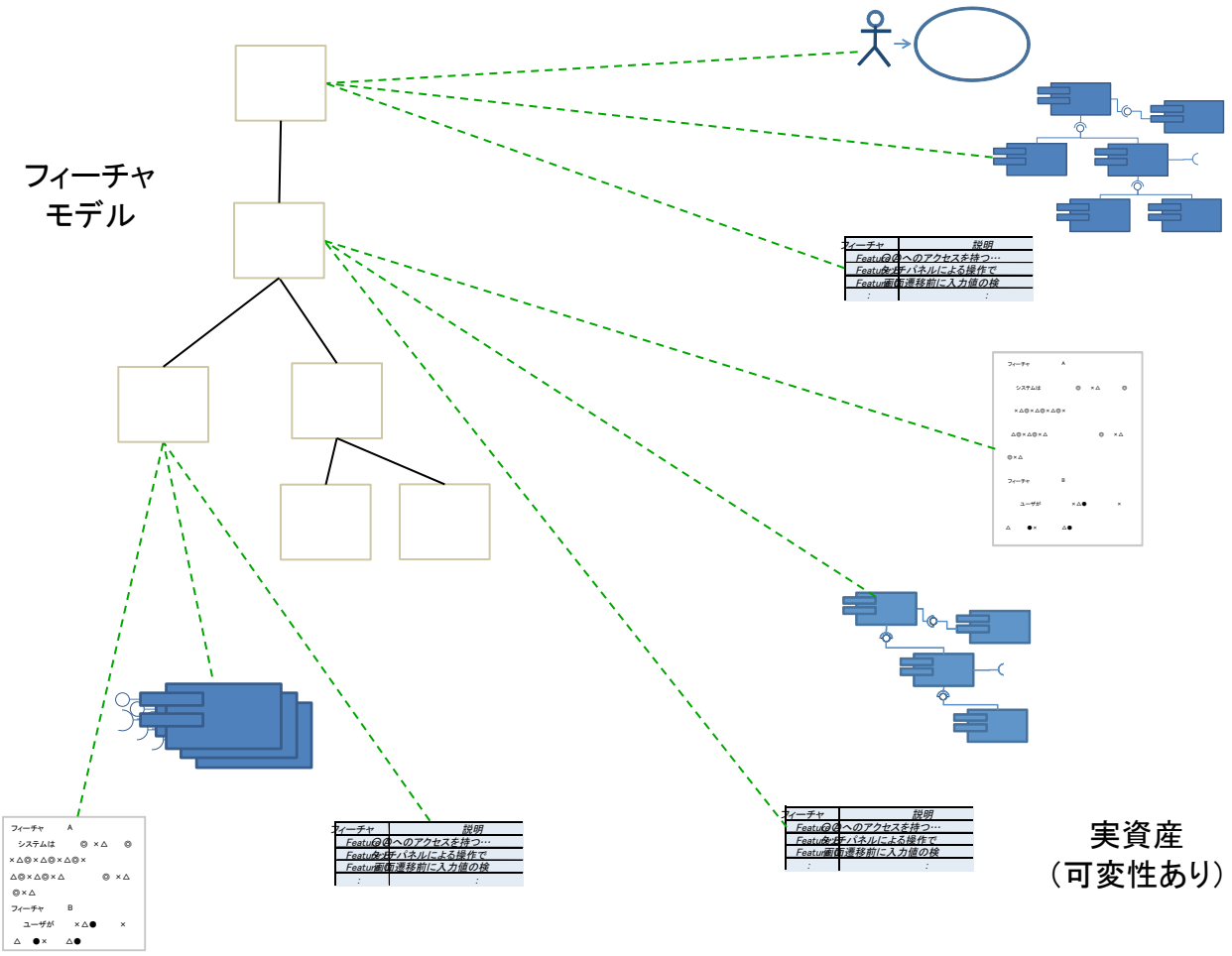
  などがある

フィーチャモデル例


能力
:製品間で共通
:製品ごとに可変
iPod 系列
音声
画像
…
Podcast
音楽
FMラジオ
オーディオブック
表示
撮影
静止画
動画
音声出力
音声入力
写真
写真撮影
映画
ビデオ
ビデオ撮影・編集
稼働環境
ディスプレイ
撮影機器
イヤフォン
マイク
スピーカ
1.54" 2.5" 3.5"
実装技法
音声フォーマット
画像フォーマット
動画フォーマット
AAC
…
JPEG
…
MPEG3
…

フィーチャモデルの構築とコア資産の関連付け

- 共通部分の扱い
  - フィーチャモデル上でフィーチャを作る（可変性なし）
  - そのフィーチャに共通部分の資産を対応付ける
- 可変部分の扱い
  - フィーチャモデル上でフィーチャを作る
  - フィーチャのタイプをその可変性を表わすものにする（オプション、多者択一、多者択多など）
  - 各フィーチャを各資産の可変部分に対応付ける



適用するとどんな姿になるのか？

- 目標の姿
  - いつ、誰が、どのように
    - 開発方式と組織体制の典型パターンと例

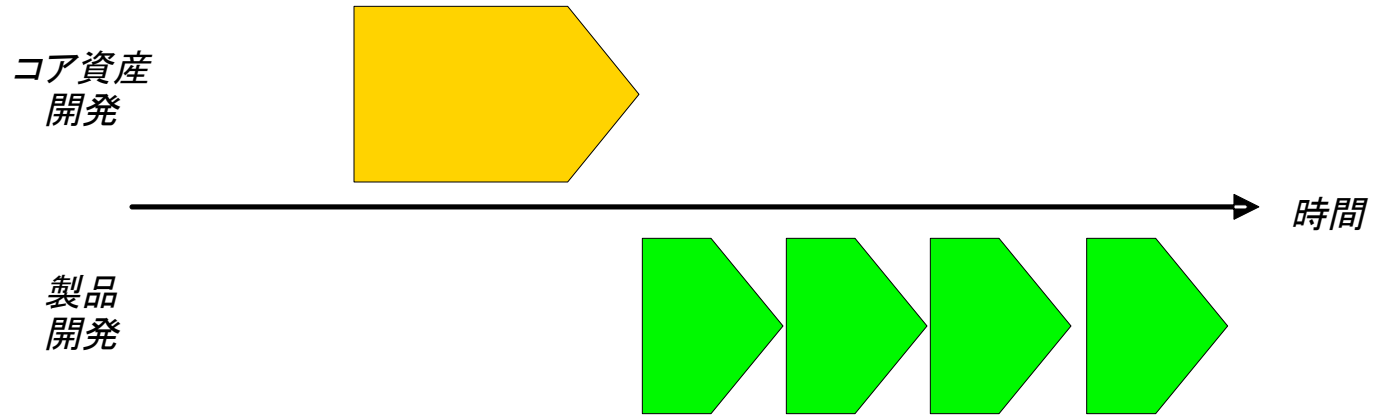
ソフトウェア製品系列開発の方式

—いつ資産を作り、いつ使うか

- 事前準備式 (Pro-Active)
- 都度対応式 (Reactive)
- そしてこれらの中間/折衷型
  - 例: 抽出式（Extractive）

事前準備式開発

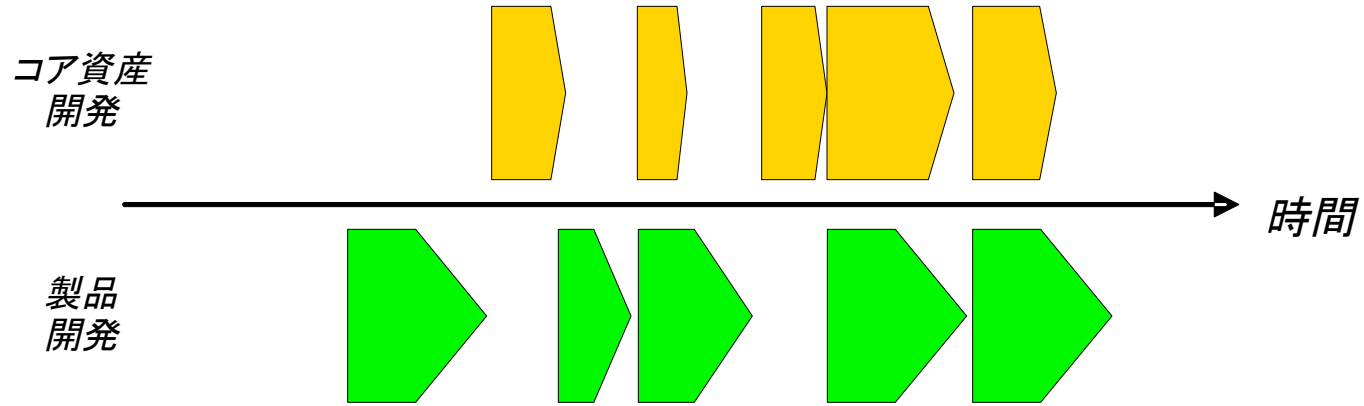
- まずコア資産を開発する
- 個々の製品の要求に基づいてコア資産を選択し適応させて、必要な追加を行ない、製品を開発する



- 初期投資の額・期間が大・長
- 関連ドメインが成熟（安定）していて、そのドメインを熟知し初期投資の余裕のある組織向き

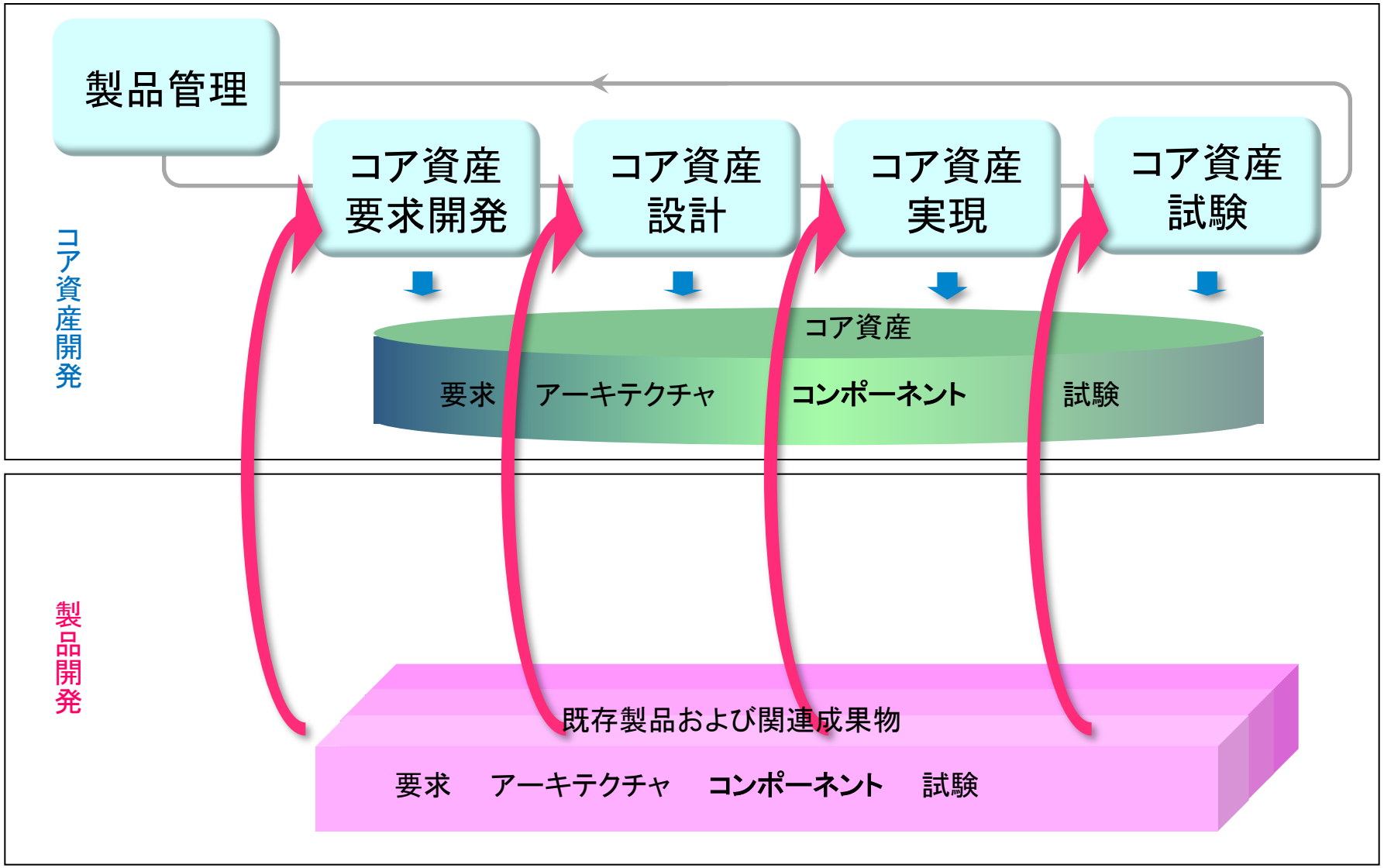
都度対応式開発

- コア資産の開発を製品の後に（または並行して）開発する
- このサイクルを、個々の製品開発時に繰り返し、コア資産を更新・統合して行く



- 初期投資の額・期間が比較的小・短
- 関連ドメインが安定していない、またはドメインをあまりよく知らず、将来予測が困難な場合向き

抽出式開発プロセス



[Pohl05] のフレームワークの部分を基に作成。

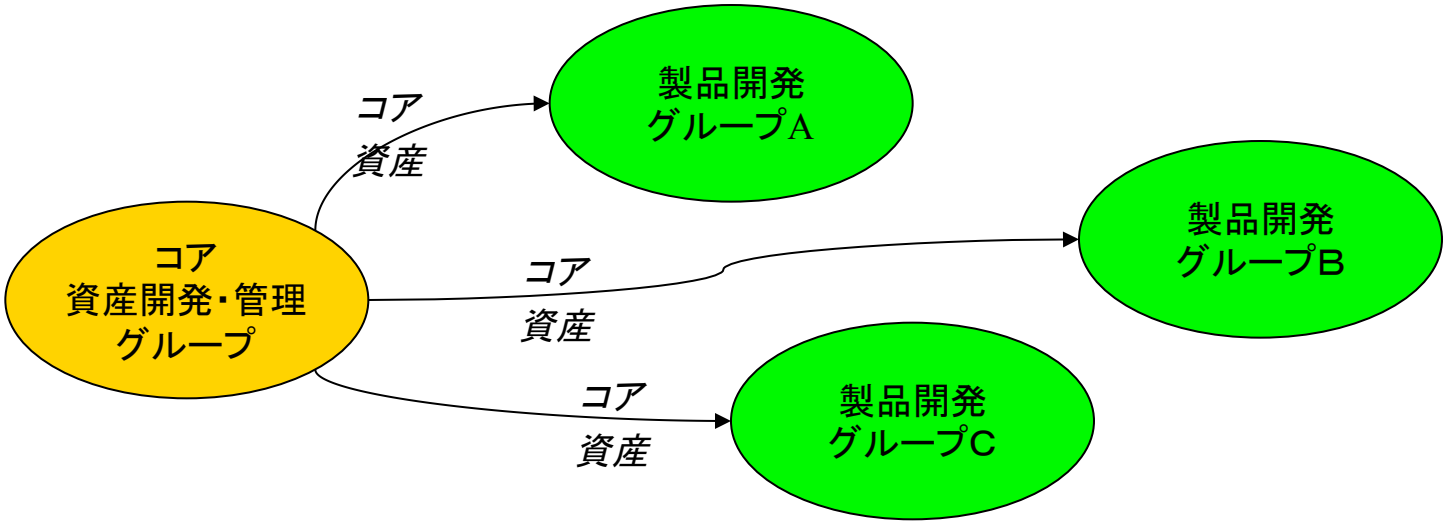
ソフトウェア製品系列開発の組織体制

—誰が資産を作り、誰が使うか

- 中央統御型 (Centralized)
- 協調開発型 (Collaborative)
- そしてこれらの中間/折衷型
  - 例: 非分離開発型

中央統御型体制

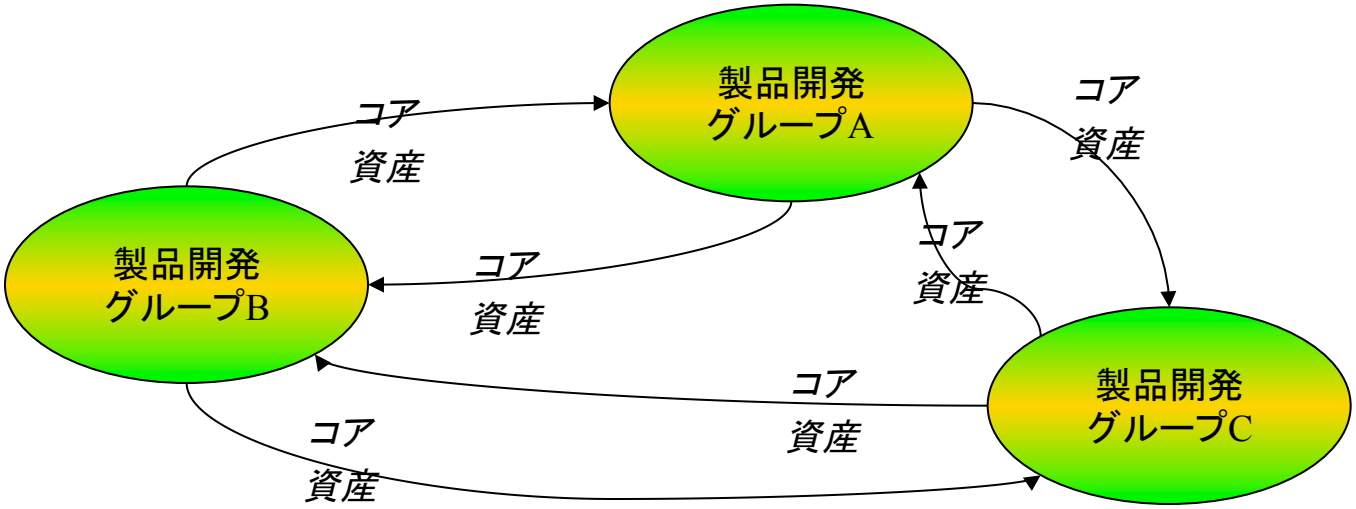
- 特定部門がコア資産の開発・管理を受け持つ
- 個々の製品の開発は他の担当部門が行なう



- 製品系列およびコア資産の管理・制御が一元的に行なえる
- 初期投資が大きく、利益回収までの期間が長い

協調開発型体制

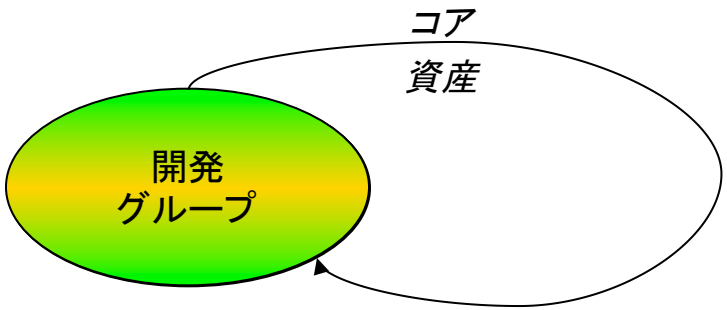
- コア資産と個々の製品の開発は部門ごとに分散
- 各部門で開発したコア資産を全部門で共有



- コア資産の費用は分散される
- 他部門のニーズに合うコア資産の開発や相互協調は容易でない

非分離開発型

- コア資産と製品開発のチームを分離しない
- 同じチームが両開発を行なう



- 製品系列が１本または少数、あるいは開発に関わる要員が少ない場合など

以上です。
ご質問はパネリストの皆さんへ

参考文献

[Shimizu07]

清水 吉男、「『派生開発』を成功させるプロセス改善の技術と極意」、技術評論社、2007

[Shimizu09]

清水 吉男,「派生開発を制覇しなければ明日はない ー派生開発プロセス[XDDP]のすべてー」、SQiPシンポジウム2009 ハーフデイチュートリアル講演会資料、2009/9/9

[Shimizu09a]

System Creates, Inc. 、「PFD(Process Flow Diagram)の書き方 第3版」、“硬派"のホームページ、2009/9/21。 http://homepage3.nifty.com/koha_hp/

[Pohl05]

Klaus Pohl, Günter Böckle, Frank van der Linden, “Software Product Line Engineering - Foundations, Principles, and Techniques,” Springer Verlag, Heidelberg, Germany, 2005. ［邦訳：林 好一、吉村 健太郎、今関 剛 訳『ソフトウェアプロダクトラインエンジニアリング──ソフトウェア製品系列開発の基礎と概念から技法まで』エスアイビー・アクセス］、2009