ソフトウェアプロダクトライン開発の様々な姿
～ プロセス、組織形態、推進戦略、始め方の違い

2012年07月20日

JASPIC プロダクトライン分科会

- SPLE の概要
- 始める前に(その1):企業間に跨る SPLE 実施の形態
- 始める前に(その2):各企業に適した SPLE 導入の形態
- 始め方:パターン、例、推奨形
- むすび

SPL Engineering とは？

- Software Product Line Engineering（SPLE）
  ＝ ソフトウェアの「**製品系列**」の作り方
- 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

共通化とも

重要：

- 生産性を向上だけでなく品質も維持/向上させる
- 桁を上げる向上は再利用以外ではありえない

キーは共通性、そして可変性

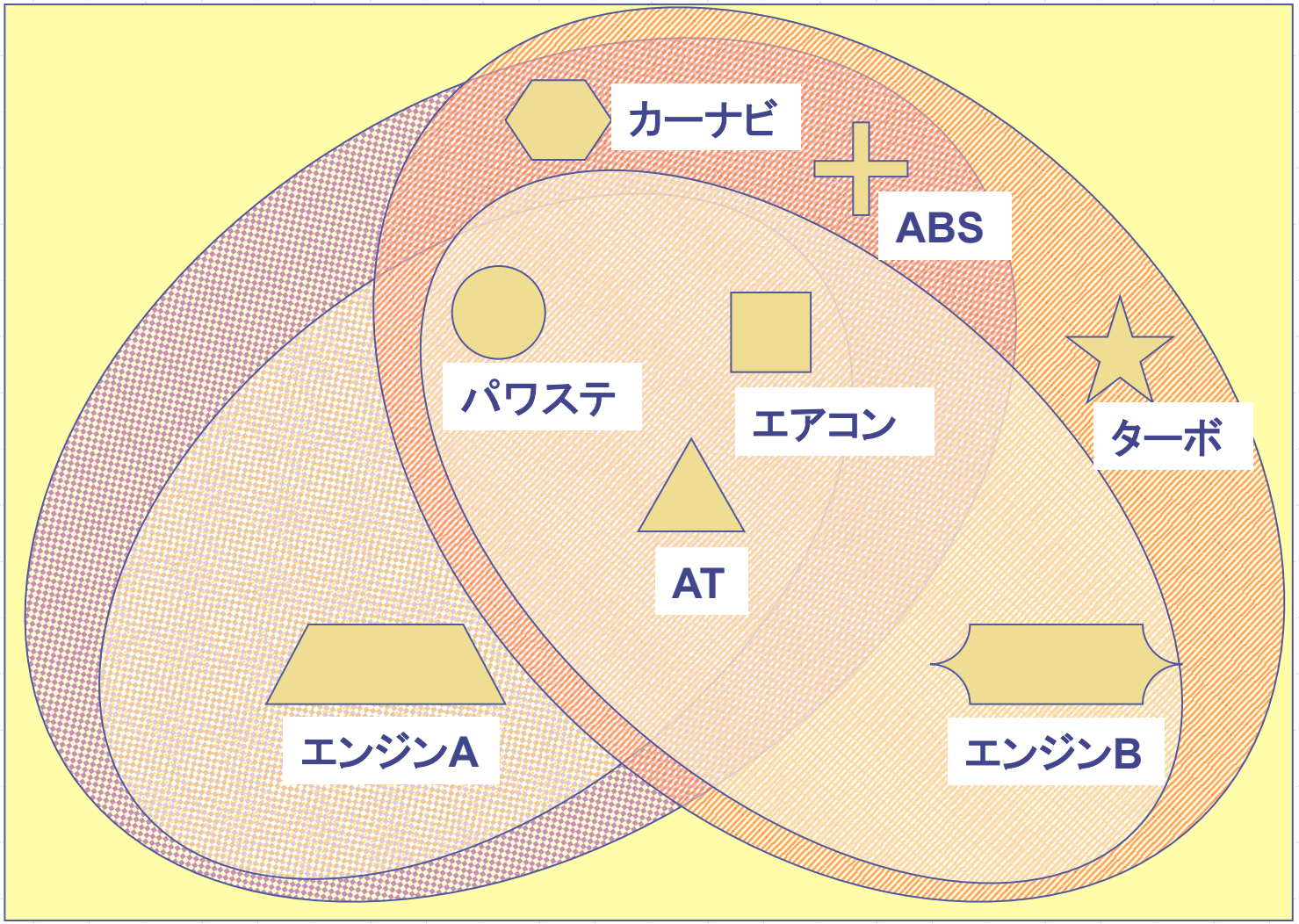
- システムが「類似している」とは**共通性**があること
  - 裏返すとそれらのシステムの間には**可変性**がある
- **共通性**を基に資産を形成し、**可変性**を活用して個々のシステムを構築する

事前にすべて作ってしまわなくても良い

製品と製品系列(例)

多様性

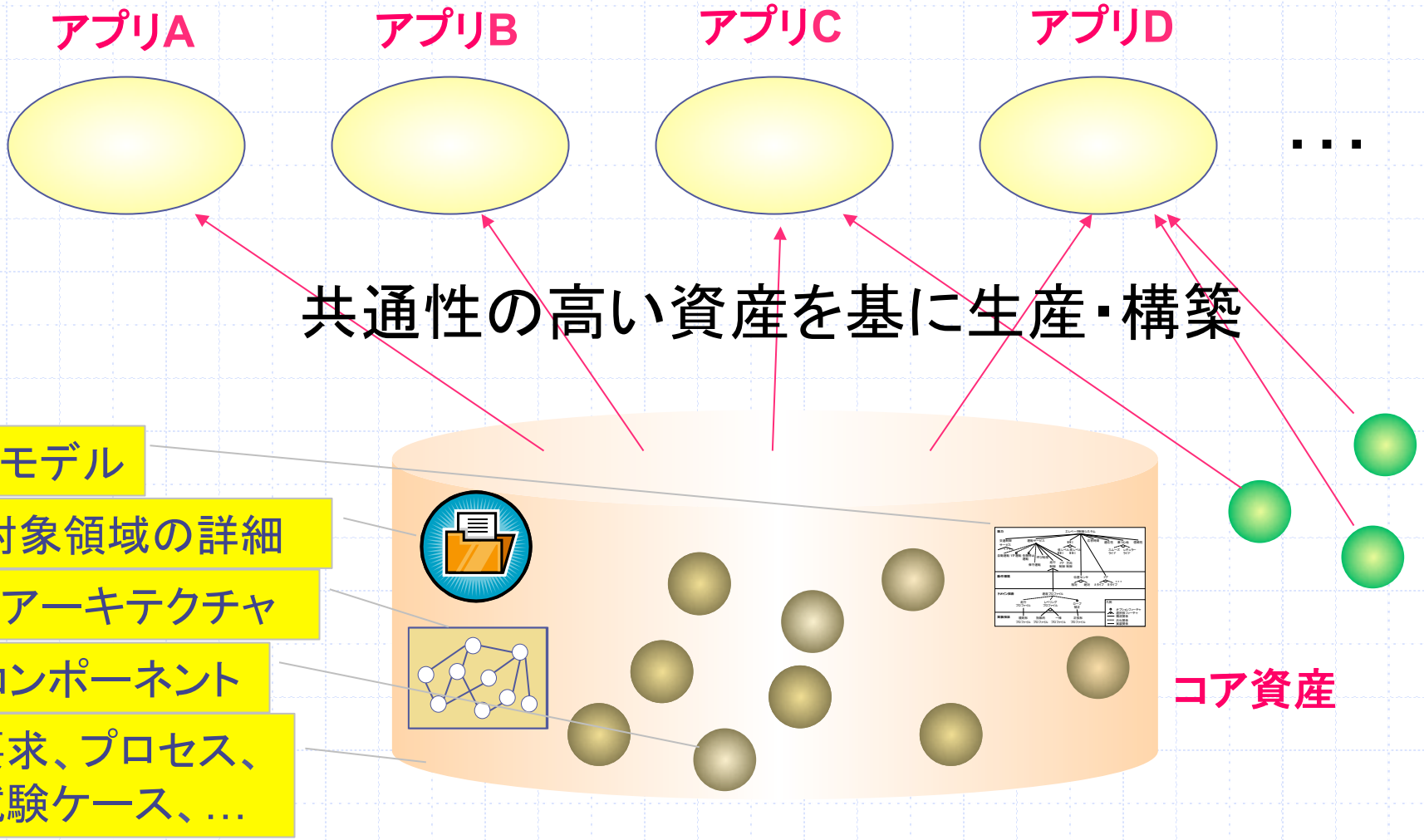
短納期



製品によって含まれるものが異なるが、互いに類似している。
これらをひとくくりにして製品系列(プロダクトライン)と呼ぶ。

製品系列
製品A
製品B
製品C
製品D

SPL アプリケーションの作り方



SPLE 導入の効果

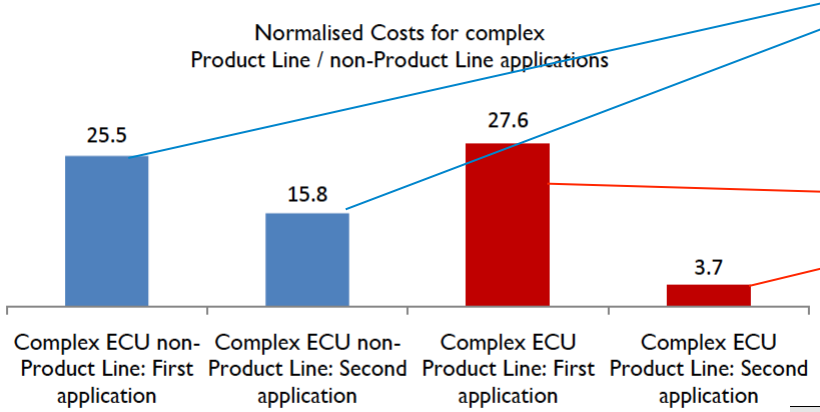
企業名	アプリケーション分野	施策			効果		
		KFS (key for success)	期間	投入資源	尺度	QCDの別	数値
ヒューレット・パッカード	プリンティング・デバイス	・権限を持った技術検討チームと事業検討チーム ●適切な標準化			開発の人的リソース	C	75%減
					開発期間	D	67%減
					欠陥数	Q	96%減
レイシオン	衛星地上管制	●深いドメイン知識の結集 ・アーキテクチャ重視姿勢 ●プロセス成熟度	3年	70人年(*1)	全体コスト開発時)(*2)	C	18.2%減
					全体コスト保守時)(*2)	C	27.8%減
シーメンス医療機器	医療画像処理・管理	・要件合意の確実な形成 ●製品バリエーションの事前計画と合意			開発期間	D	～75%減
					品質コスト(*3)	C	57%減
カミンズ	ディーゼルエンジン制御	●経営層のコミットメント ●開発プロセスの標準化 ・コードのみならず他の全主要成果物の再利用	1年(*4)		開発工数	C	72%減
					ROI(*5)	-	10倍
ボッシュガソリンシステム	自動車エンジン制御	●組織としての展開決定 ●経営層の支援 ・複数部門から各専門職を集めたチーム ・CMMによるプロセス基盤			消費資源	C	30%減
					キャリブレーション工数	C	～20%減

*1: 再利用率および再利用形態からの推定
*2: 開発要員のみ、概数
*3: 将来も含めた推定
*4: テストに掛かるコスト
*5: 全開発プロジェクトを止めて体制移行
*6: 同社による推定

数値は [Pohl05] および [vanderLinden07] から取った。

ゼネラルモータースでの効果

Figure 2: Graph demonstrating cost efficiencies achieved with product line delivery of complex systems against those of non-product line delivery



Source: General Motors

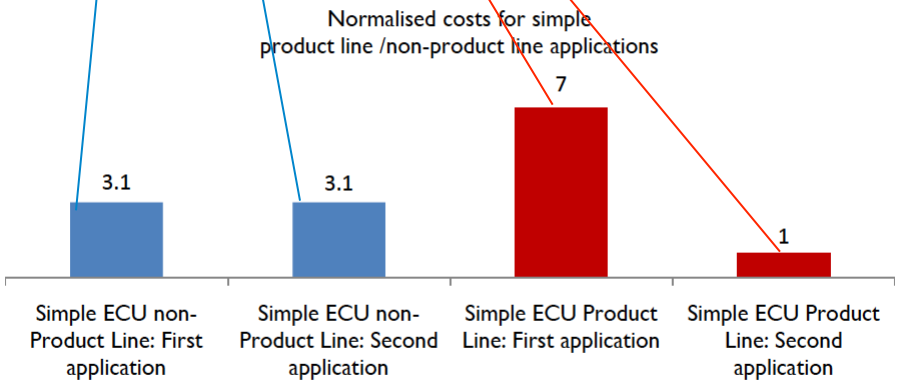
非プロダクトラインで作った製品1本目と2本目のコスト

プロダクトラインで作った製品1本目と2本目のコスト

↑ 複雑なシステムでの例

単純なシステムでの例 →

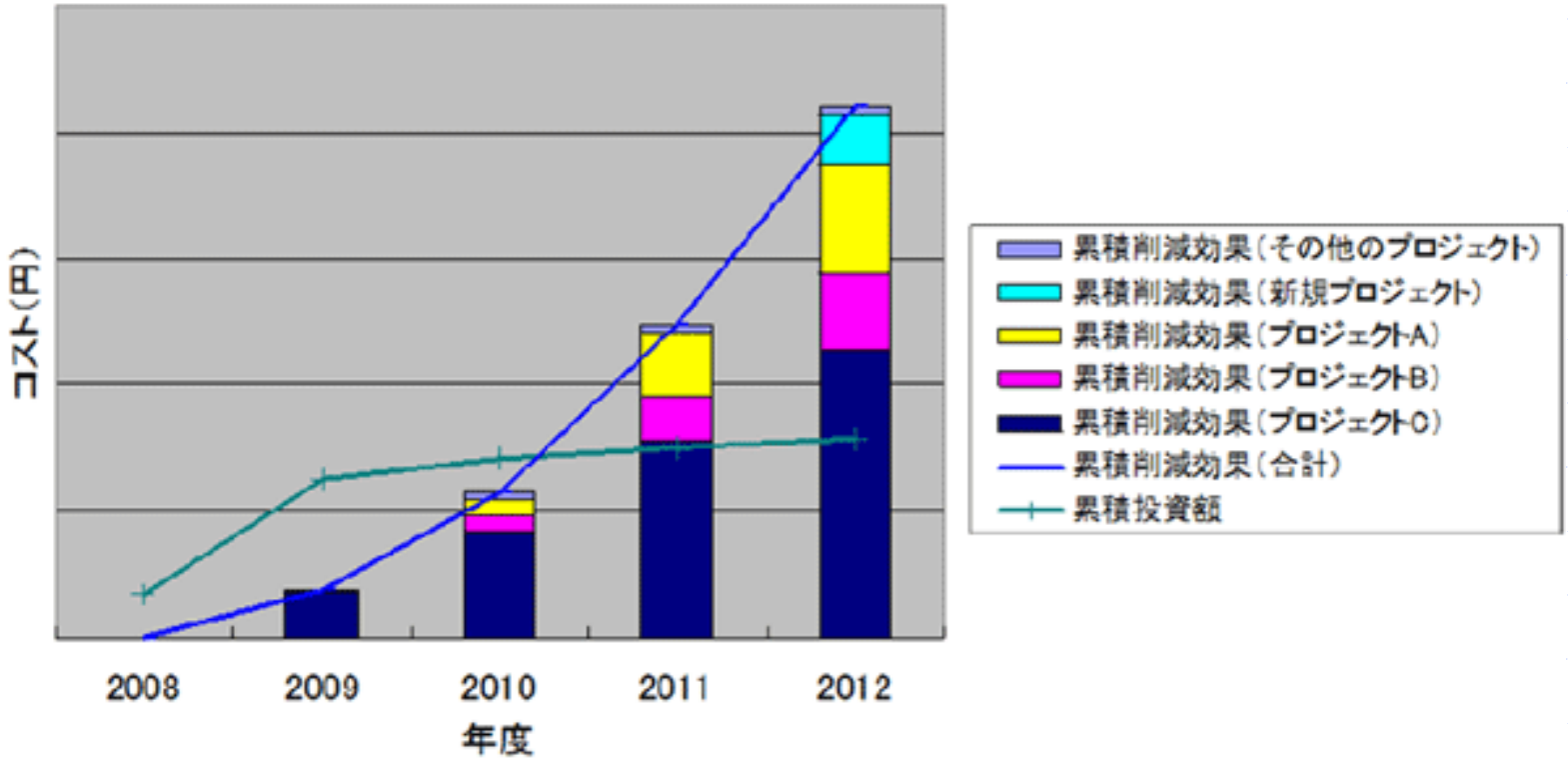
Figure 3: Graph demonstrating cost efficiencies achieved with product line delivery of simple systems against those of non-product line delivery



Source: General Motors

* [Rotibi10] より引用

富士通九州ネットワークテクノロジーズでの効果



[Iwasaki11]より引用。

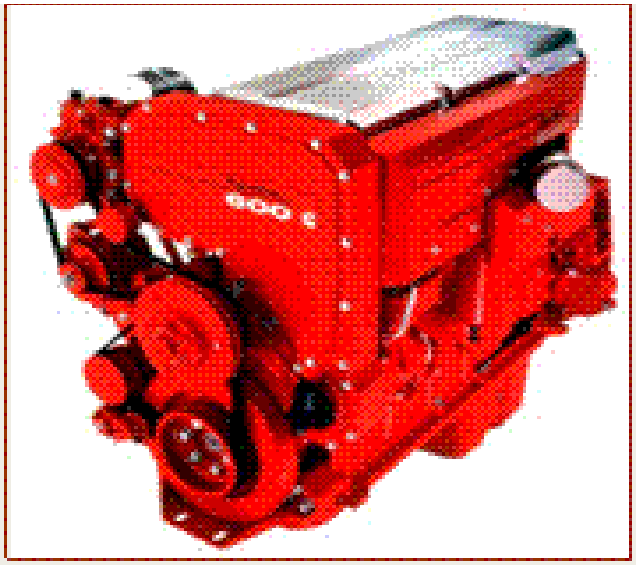
その他の SPLE 適用事例

Carnegie Mellon
Software Engineering Institute

カミンズ社：ディーゼルエンジン制御システム

1,000以上の個別のエンジン制御アプリケーションからなる20以上の製品群において、

- 製品サイクルタイムが250人月から数人月に短縮
- ビルドおよび統合にかかる時間が1年から1週間に減少
- 品質目標以上の品質を達成
  顧客満足度が高い
- 製品スケジュールが守られている



- http://www.sei.cmu.edu/productlines/essentials/essentials_11.htm

参考：SPLE適用事例サマリ 1/2

導入企業	ビジネス	開発者数	導入時の開発状態	導入による主な効果
AKVAスマート(*1)	魚の養殖産業向け給餌制御・養殖管理ソフトウェア	6～10人	レガシーシステムのリプレース	・70%以上のコード削減 ・使用感の統一 ・技術基盤およびコードスタイルの共通化 ・再利用、保守、統合の容易化
Boschガソリンシステム(*1)	エンジン制御システム	～1,000人	既存資産をベースにした戦略重視型	・較正工数（20%）および保守工数の削減 ・リソース消費を20～30%削減 ・市場の多様性に応じた製品系列の定義
Cummins(*2)	エンジン制御システム	不明	多数の類似のソフトウェアを個別に開発	・製品サイクルタイムが250人月から数人月に短縮 ・ビルドおよび統合にかかる時間が1年から1週間に減少 ・品質目標以上の品質を達成、高い顧客満足度
DNVソフトウェア(*1)	海運、石油・ガス、鉄道等産業向け情報処理システム	100人	三つの清算センタを共通支援	・ソフトウェア基盤の共通化 ・早期市場投入、ライフサイクルコスト削減と高品質 ・マーケティング、販売、開発の連携の強化
日立製作所(*3)	エンジン制御システム	不明	レガシーシステムからの再利用可能資産の切り出し	・既存資産の可視性の向上（予定） ・品質を維持した再利用性の向上（予定）
九州日立マクセル(*4)	マッサージチェア	不明	変更可能性の高い部分をなるべく集約したアーキテクチャ	・基本アーキテクチャを温存したままシリーズ展開 ・顧客満足度指標を容易に既存および新規の資産に対応付け
マーケットメーカー・ソフト(*1)	株式市況および金融市況の管理・表示ソフトウェア製品	25人	スクラッチからの戦略的多品種開発	・市場投入期間が1/2～1/4（推定） ・製品5本出荷後に従来開発方法より安価に ・保守コスト～60%削減
ノキア・モバイルフォーン(*1)(*2)	モバイル電話機製造	＞1,000人	既存資産をベースにした戦略重視型	・ソフトウェア進化の理解向上 ・共通性に対する洞察の深化
ノキア・ネットワークス(*1)	ネットワークインフラストラクチャ、通信・ネットワークサービス基盤、およびオペレータおよびサービスプロバイダ向け	＞1,000人	既存資産をベースにした戦略重視型	・高複雑度のシステムの管理 ・利用可能資産の再利用と可視性向上

*1: [vanderLinden07]に基づく。一部[JASPIC08]を参考にした。
*2: [Clements01]に基づく。
*3: [Yoshimura07]に基づく。
*4: [Abeta08]に基づく。

参考：SPLE適用事例サマリ 2/2

導入企業	ビジネス	開発者数	導入時の開発状態	導入による主な効果
野村総合研究所 (*5)	流通業向けエンタープライズシステム受託開発	不明	過去の経験の蓄積でシステム構築	・開発および保守時の読解工数の削減 ・可変性管理の容易化 ・設計再利用による生産性向上 ・類似性がやや低い他業種にも応用
フィリップス ・コンシューマー ・エレクトロニクス テレビ向けソフトウェア (*1)	末端消費者向け電子製品	250人	新アーキテクチャ、リバース・エンジニアリング ・コード	・全ての中～高価格帯テレビ向けソフトウェアを一つのソフトウェアプロダクトラインで対応 ・マーケティングによる望ましい可変性を生み出す能力 ・製品開発におけるソフトウェア開発のクリティカルパスからの脱出 ・6年後にも新規アーキテクチャは不要　以前は5年以内に再開発）
フィリップス医療システム (*1)	X線、MRI、CT、超音波等の医療向け画像機器	＞1,000人	既存資産をベースにした戦略重視型	・工数が1/2～1/4に低減 ・再利用機能に関して市場投入期間を50%以下に短縮 ・再利用機能に関して製品欠陥密度を50%に低減 ・使用感の共通化 ・製品計画およびロードマップ使用の向上 ・一つの製品から他の製品へフィーチャを容易に
シーメンス医療ソリューション (*1)	X線管、MRI・CTスキャナからインフラストラクチャサポートまでを含む病院および医療従事者向けハードウェアおよびソフト	100人	ボトムアップ	・再利用レベル :～50% ・開発サイクル期間短縮 :～25% ・品質コスト削減 :～57%
Telvent (*1)	エネルギー、交通管制、運輸、環境の四つのセクター向けのソリューション	＞1,000人	多くの顧客要件変更に設定で対応	・サーバコア資産を他市場へも振り向け ・実行時可変性の導入 ・参照プロセスフレームワークの改善 ・コア資産のロードマップの集中管理
東芝 (*6)	原子力・火力発電所向け監視制御システムの開発	不明	流用開発？	・仕様多様化・複雑化に対応 ・生産性、品質の維持 ・共通部、可変部、新規部を扱うプロセスの明確

*1: [vanderLinden07]に基づく。一部[JASPIC08]を参考にした。
*5: [Ishida07]に基づく。
*6: [Takizawa09]に基づく。

従来の「再利用」例　1/2

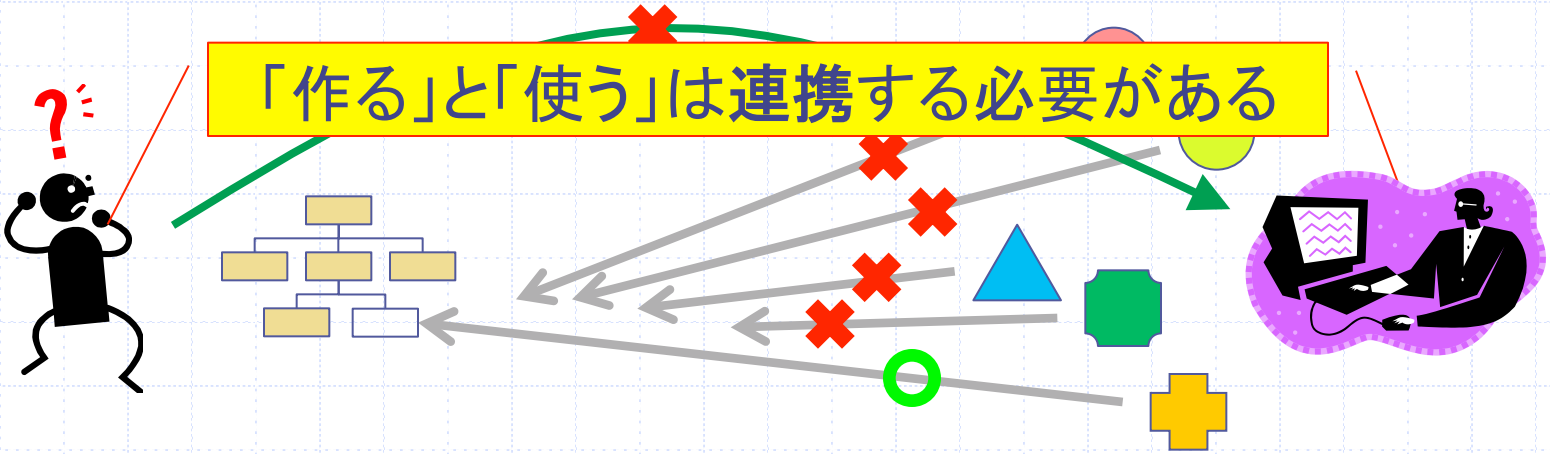
- 流用開発
  - 流用元を探す手間、どこをどう変えるかを考える手間
  - 流用元へのフィードバックが不十分/無し

  =「見つかれば使う」という程度の「再利用」

- 部品開発
  - 使われる仕組み・努力が不足/不適切なまま作る

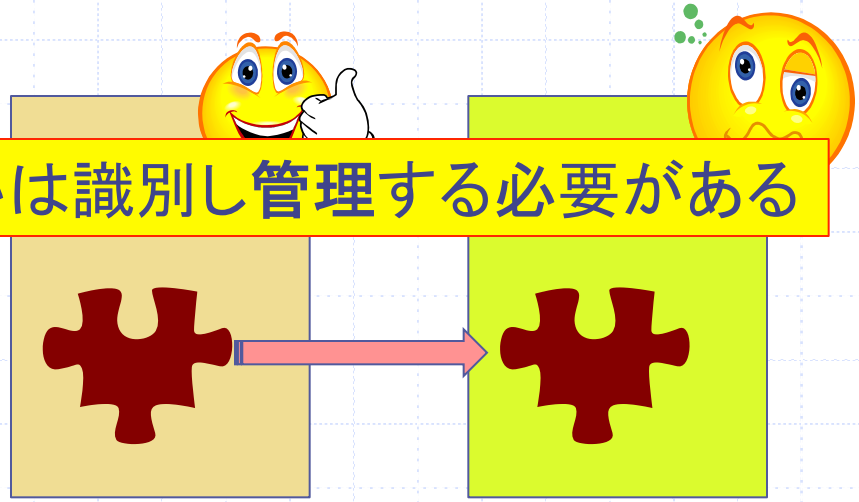
    →知らない・使いにくいので使われない

  =「使うべき」ものを作るが使われる保証はない



従来の「再利用」例　2/2

- 無制約の改変
  - どこをどう変えるかは一技術者の裁量で決まる
  - 多種多様の改変が発生して、何が根幹で何が枝葉かわからなくなり、保守・拡張・さらなる再利用が困難になる
- 無計画な再利用
  - 仕様が似ているから(より典型的にはコードが似ているから)流用する
  - 想定外の事故が起こる（例：Ariane 5, Therac 25）

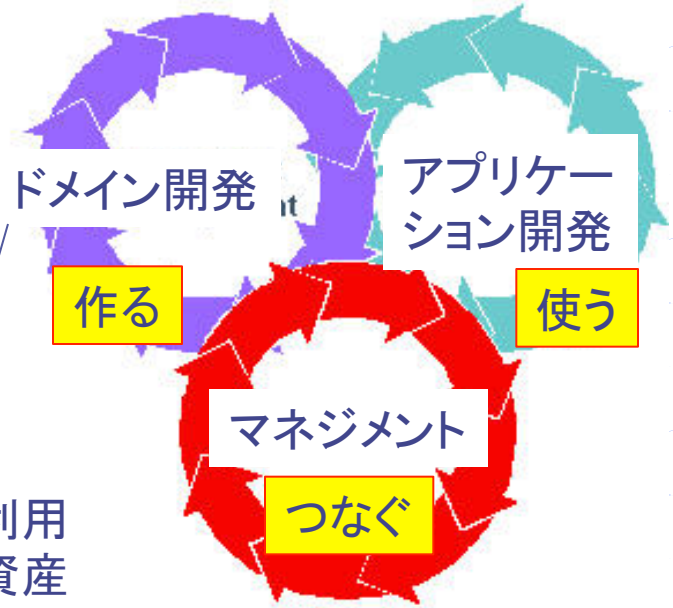


どこをどう変えるか・変えないかは識別し管理する必要がある

SPLE の特徴

連携

- 再利用可能資産の構築とその活用が連携している
  - 何度も使うとわかっているものを作るから高効率が保証できる



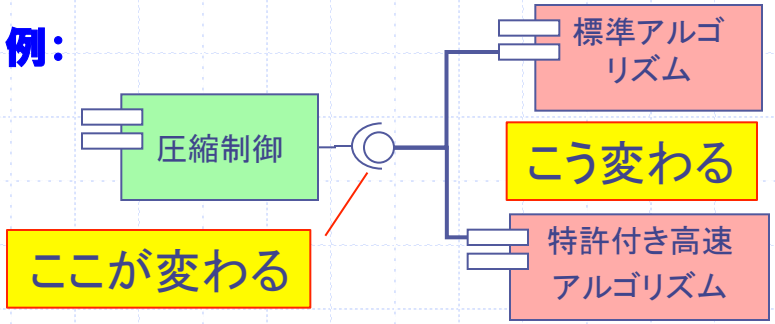
＝再利用可能資産

米 Software Engineering Institute の図を編集

差異の管理

- アプリケーション間で異なる点を識別・共有・管理する
  - 品質を確保した改変が計画できる

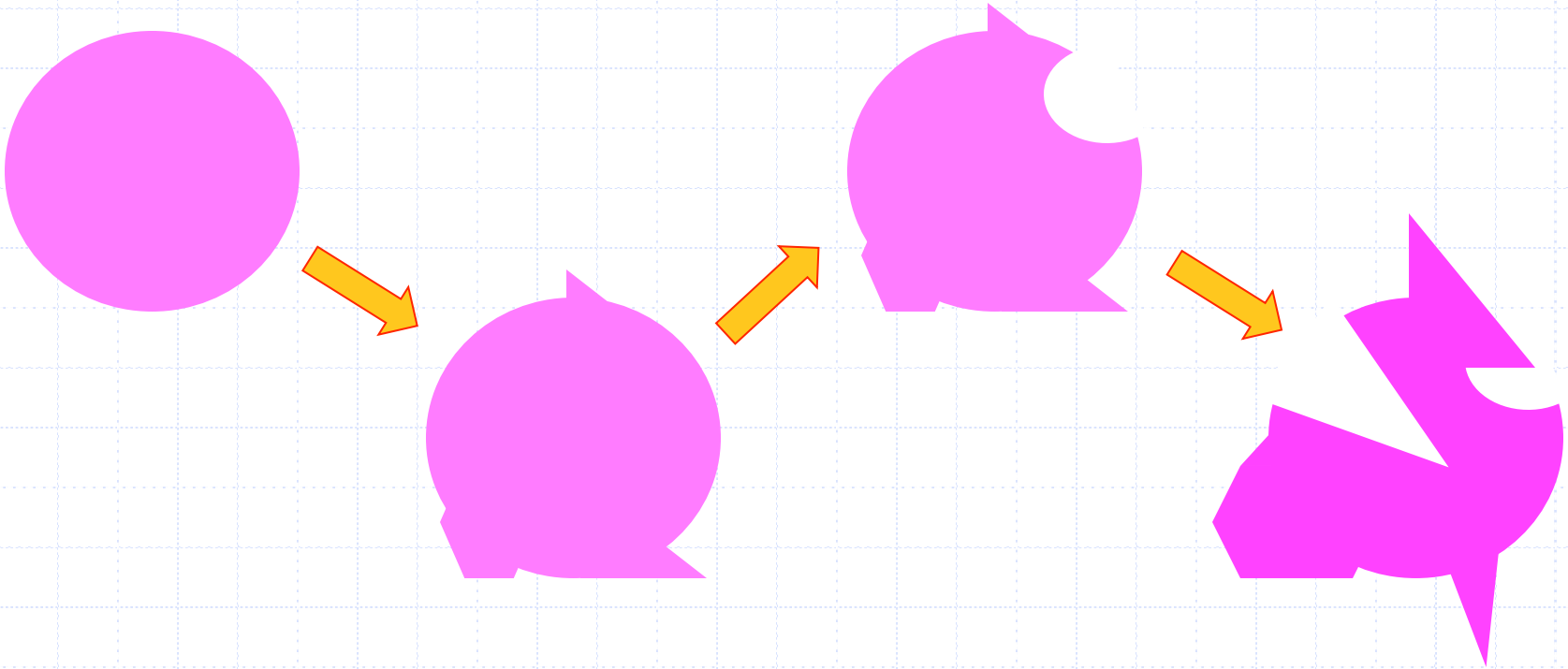
例:



- アーカイブ圧縮のアルゴリズムが速い版と普通の版を計画する
- 変更は、圧縮を制御するコンポーネントの要求インタフェースにおいて、コンポーネントを入れ換えることで行なう

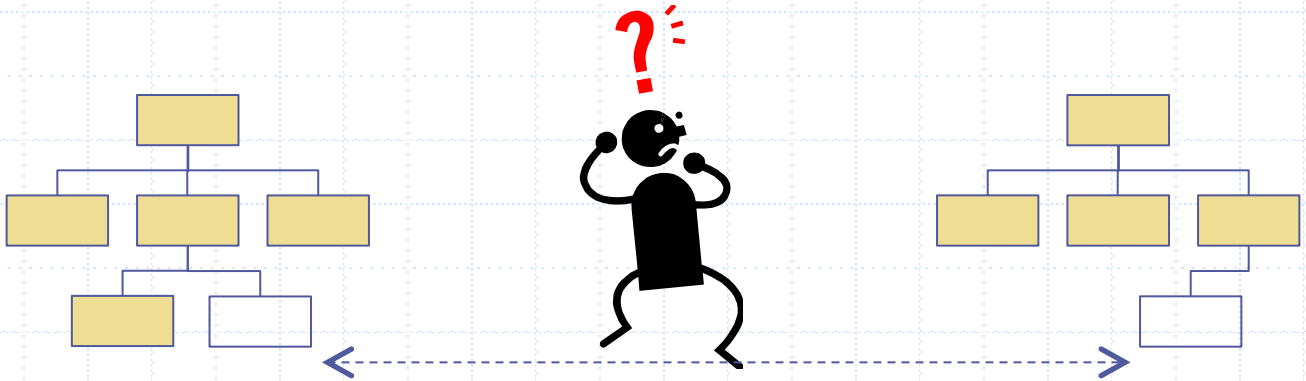
「可変」とはいっても…

- 何でも変えられる（無統制）のは良くない



「共通」とはいっても…

- 結果として共通になるのではなく、共通にする努力をしなければ高効率・高品質が担保できない
  - 「再利用率」は効率を表わさない
- そもそも共通にして良いのかどうかの分析が必要



SPLE による品質、費用・工期の向上

費用・工期

- **共通性**を活用し、異なるところだけを別途用意する
- 費用が減るように再利用を**計画**し、計画に沿って再利用する

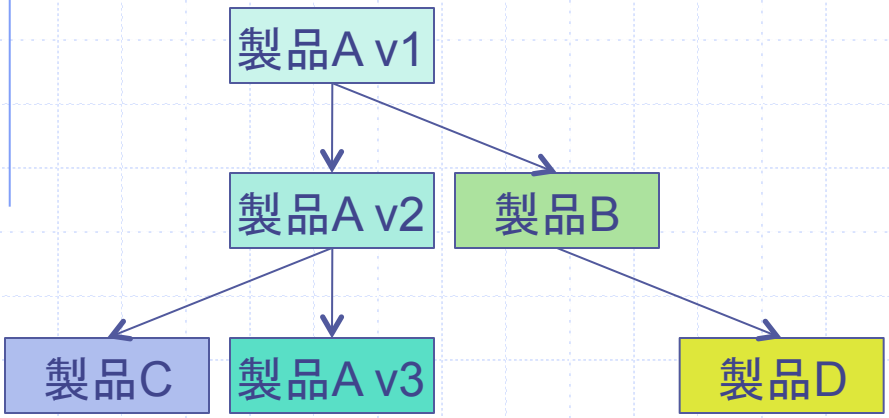
品質

- 何回もの検証
  - 再利用可能資産開発時と製品製造時（複数回）
- 製品開発からの**フィードバック**
  - 再利用の度に改善機会がある

派生開発と SPLE

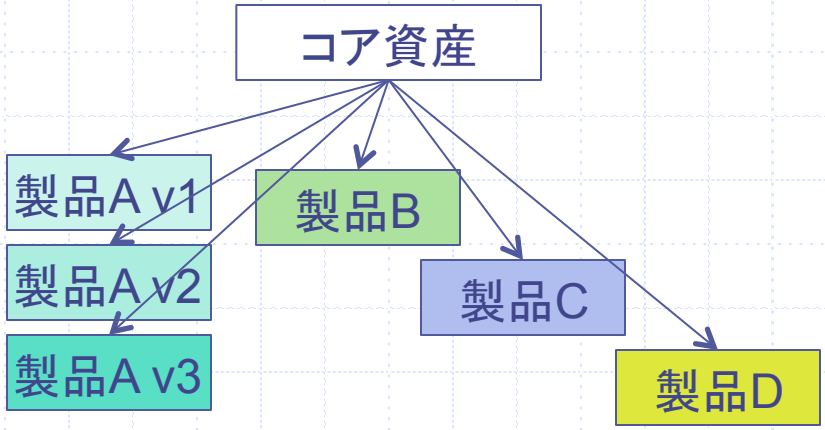
派生開発：差分開発、改造開発、改良保守、等ともいう

派生開発の典型例



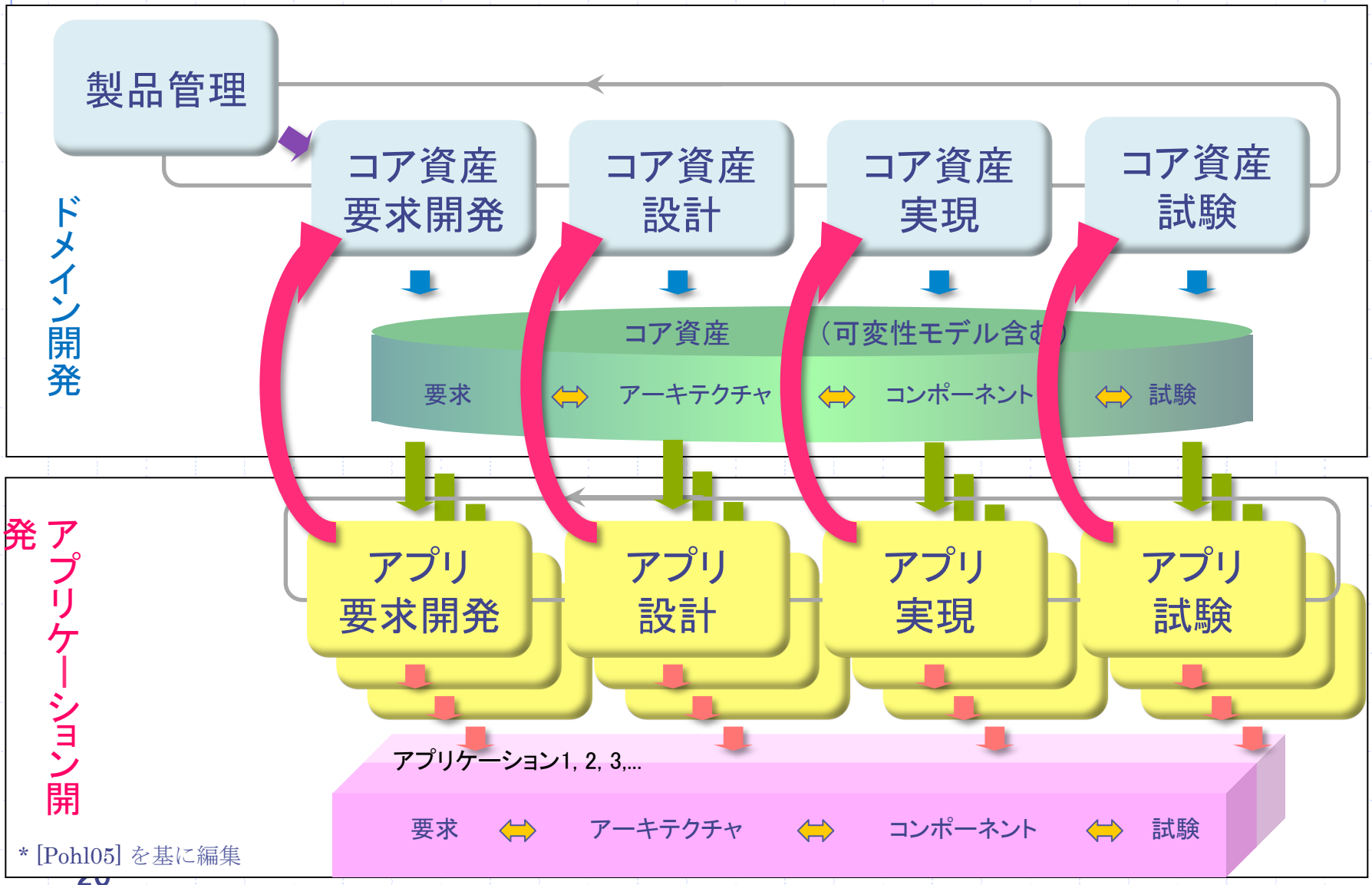
- 過去のシステムを基に再利用を検討する
  - 作ったものを再利用する

SPLEの基本形



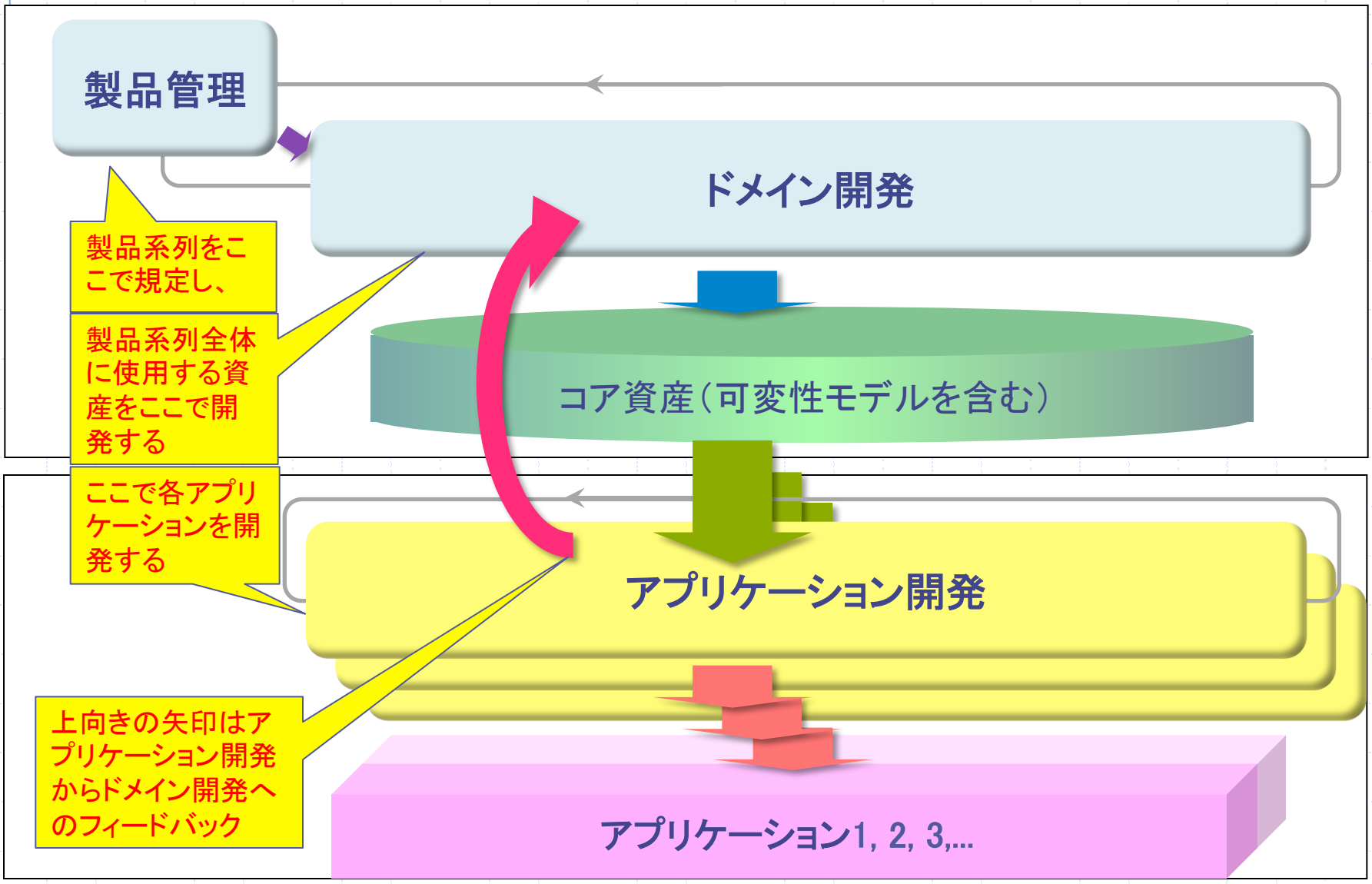
- 将来のシステムを基に再利用を検討する
  - 再利用するものを作る

SPLE のプロセス：二段階開発



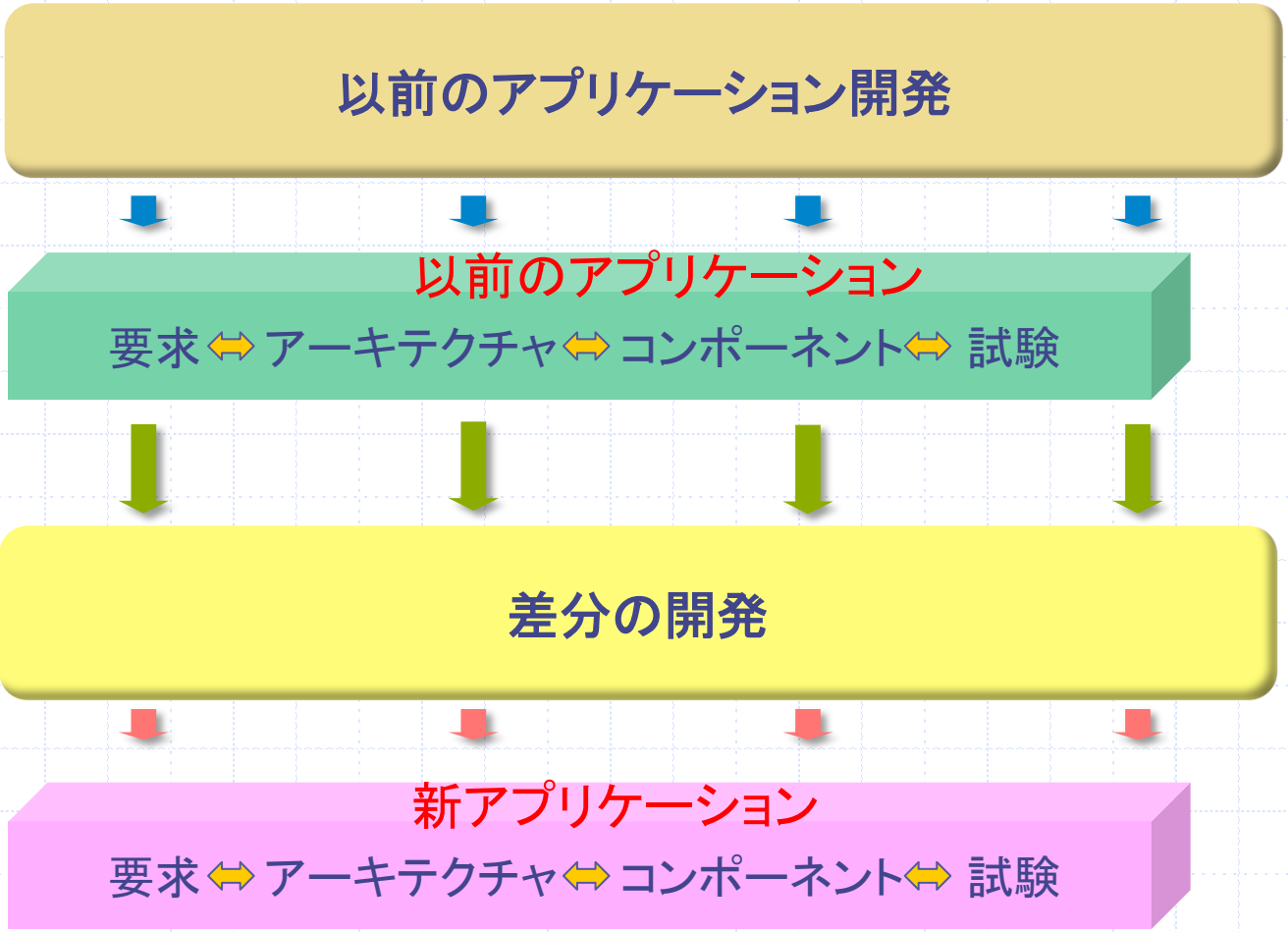
* [Pohl05] を基に編集

SPLE のプロセス：二段階開発



派生開発

基となる資産（以前のシステム）の開発は終了している



派生開発と SPLE の違い：元資産の更新

- SPLE でのコア資産の保守・進化ではバックワードコンパチビリティを持たせるが、派生開発では「資産」が別のシステムであるので新システムとの整合性は問題としない
  - 派生開発では、派生元にフィードバックはしない
  - SPLE ではコア資産にフィードバックする

- SPLE の概要
- 始める前に(その1):企業間に跨る SPLE 実施の形態
- 始める前に(その2):各企業に適した SPLE 導入の形態
- 始め方:パターン、例、推奨形
- むすび

背景、目的、ゴール

背景

- 近年、1社で開発工程の全てを担当するのは現実的ではない。最も可能性の高い開発形態として、グループ会社で開発を分業している場合のSPLEの適用方法について考察する。

目的

- 上流開発を親会社が、下流を子会社が受け持つという形態でのプロセスの分担を検討する。

ゴール

- 親会社と子会社とでSW開発を行う場合のSPLE適用の形や注意点をまとめる。

サブプロセスの内容とSPL固有技術の必要性

SPL技術		要求開発	設計	実現	試験
ドメイン開発	必要	共通要求定義 可変要求定義 共通性分析 可変性分析 可変点と変異体定義 ドメイン可変性のモデル化	参照アーキテクチャ設計 可変性の追加	可変コンポーネントのインターフェース設計 可変コンポーネント設計	不変・可変試験設計 （不在変異体の特定）
	不要	共通部と変異体の仕様記述	参照アーキテクチャの詳細設計（インターフェース、可変性の固定方法）	不変コンポーネントのインターフェース設計 不変コンポーネント設計 コンポーネント詳細設計 実装	試験仕様策定 試験
アプリケーション開発	必要	ドメイン要求からの可変性の固定 要求差分の分析 可変点、変異体の追加 アプリケーション可変性のモデル化	ドメイン設計からの可変性の固定 アプリケーション固有の変異体追加 コンポーネント構成の決定	コンポーネントの可変性の固定（構成・設定支援）	ドメイン試験仕様からの可変性の固定 変異体不在試験仕様策定 アプリケーション依存関係試験仕様策定
	不要	追加した可変点と変異体の仕様記述	アプリケーションアーキテクチャの設計	アプリケーション固有インターフェース・コンポーネントの設計・実装 （差分の設計・実装） アプリケーション構築	試験設計 差分の試験仕様策定 試験

サブプロセスの成果物

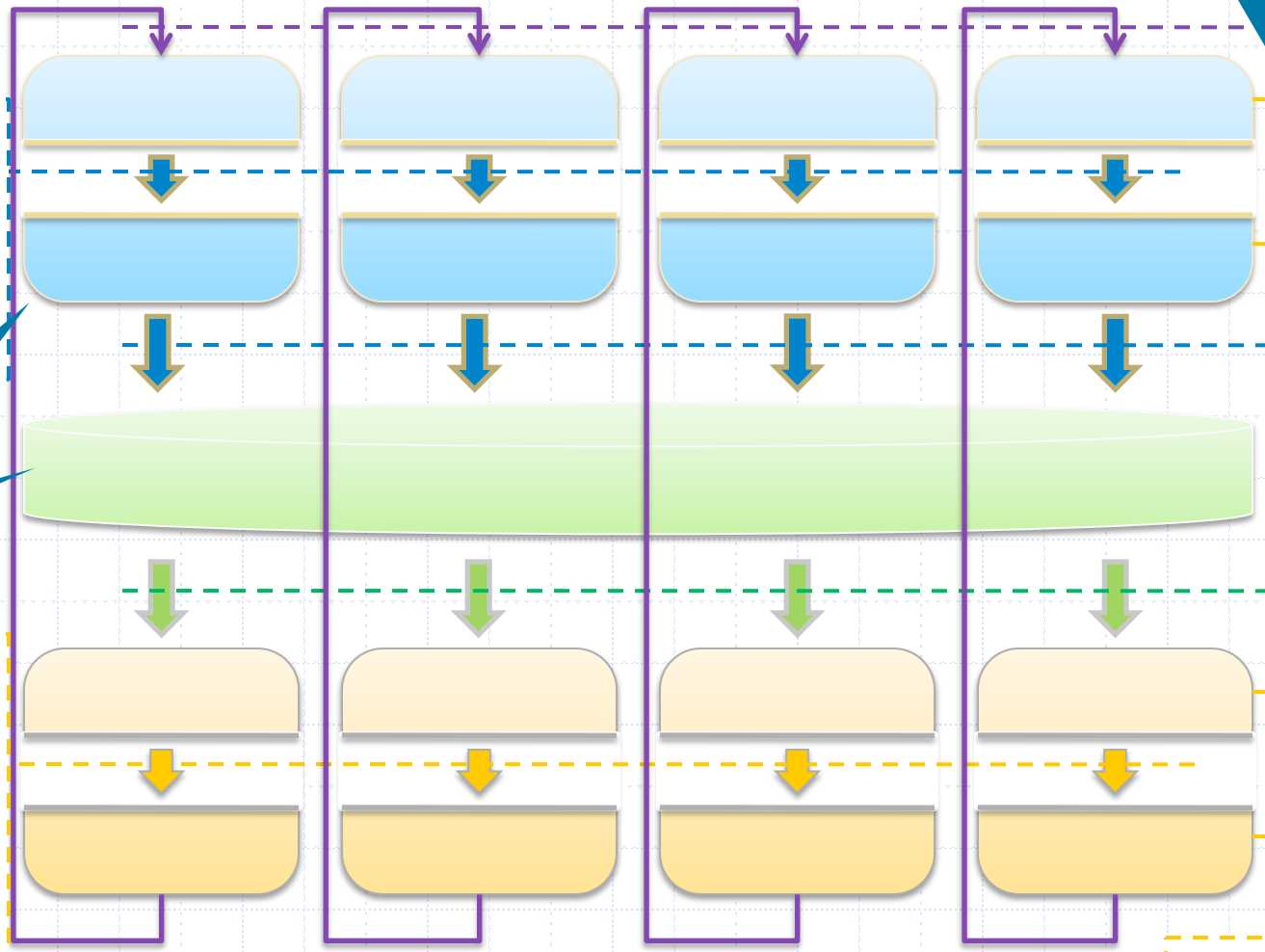
SPL技術		要求開発	設計	実現	試験
ドメイン開発	必要	直交可変性モデル＋要求仕様書（共通要求概要、変異体要求概要）	参照アーキテクチャ設計書	可変コンポーネントのインターフェース仕様書 可変コンポーネント設計書	不変・可変試験設計書 （不在変異体の特定）
	不要	要求仕様書（共通要求、変異体要求）	参照アーキテクチャ詳細設計書	不変コンポーネントのインターフェース仕様書 不変コンポーネント設計書 コンポーネント詳細設計書 ソースコード・オブジェクト	試験仕様書 試験成績書
アプリケーション開発	必要	（アプリケーション要求仕様書（追加概要含む）＋アプリケーション追加部直交可変モデル） or （アプリケーションベース要求仕様書）	（アプリケーションベースアーキテクチャ設計書）	構成設定書	アプリケーション試験設計書 変異体不在試験仕様書 アプリケーション依存関係試験仕様書
	不要	アプリケーション要求仕様書（アプリケーション追加部含む）	アプリケーションアーキテクチャ設計書	アプリケーション追加コンポーネントのインターフェース仕様書 アプリケーション追加コンポーネント設計書 ソースコード・オブジェクト	アプリケーション追加試験設計書 アプリケーション追加試験仕様書 試験成績書

ここの直交可変モデルはアプリケーション実現のために、やむなくコア資産に手を入れなくてはならない場合に必要になる。

外注依頼の注意事項

ハンバーガーモデル

* [Pohl05] を基に編集



コア資産の試験、アプリケーションを考慮した試験

アプリケーション成果物からドメイン資産へのフィードバック判断

可変性をどう実現するか、の指示。
製品戦略的に外注に出して良いのか。

コア技術

コア資産

納品品質の確保
可変性を持っている

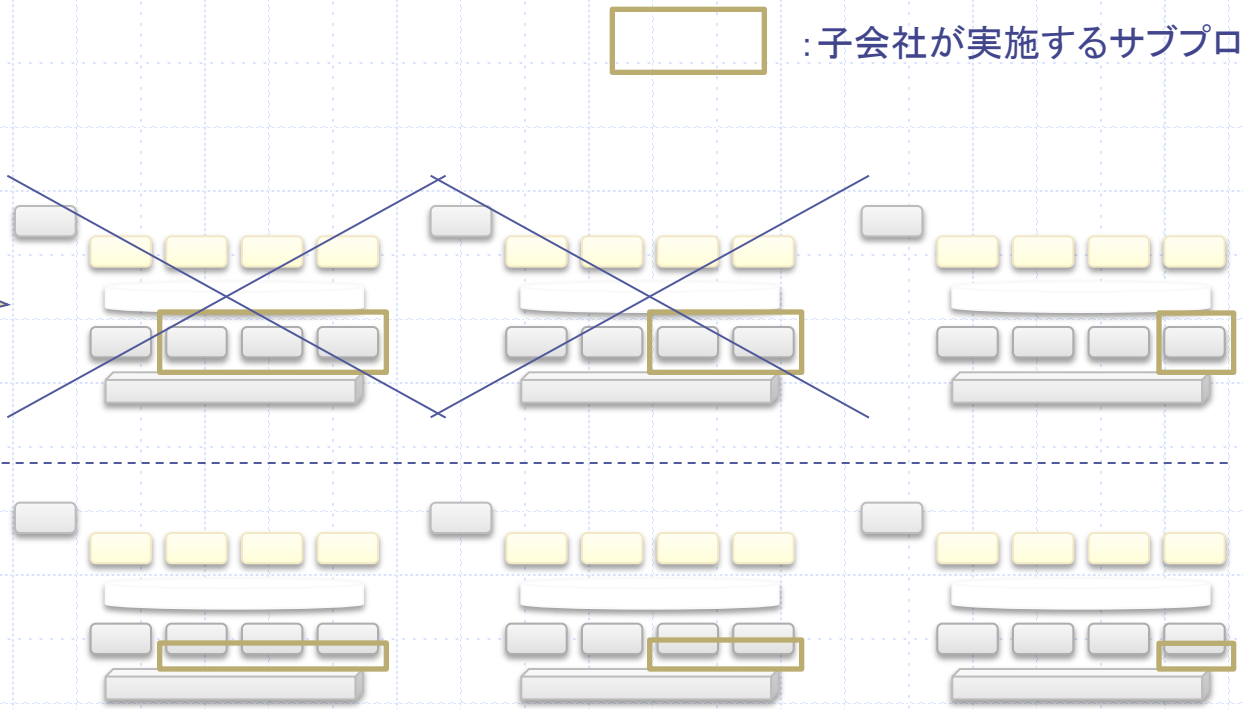
アクセス、あるいは提供方法の明確化
ドメイン資産の改変利用を勝手にさせない

可変点を固定する手段の明確化
可変点の固定指示、あるいは固定した成果物を出す。
コア技術かどうかの観点も必要

試験設計が重要

－契約時の注意－
コア資産に対するソフトウェア寿命。コア資産故、1製品寿命より長期に利用される。SPL特有ではないがコア資産のロイヤリティにも注意。

コア技術を自社以外に見せたくない開発パターン

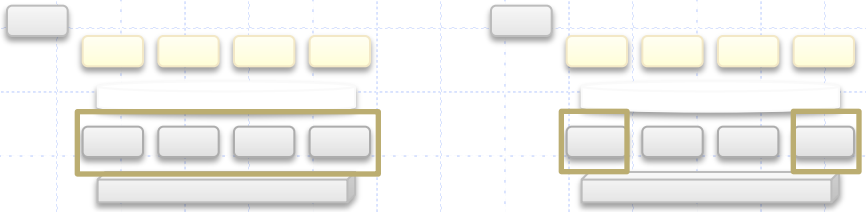


コアはライブラリなどのバイナリ形式で渡す。可変点の固定もコアソースに触れない形で実現する。

コアアーキテクチャの部分がブラックボックスでも、可変点の部分が見えていれば左上のパターンもありうる。が、アーキテクチャを知らなければその製品の新規に関わるアーキテクチャを設計できない。

コア開発拠点と現地法人による開発パターン

：子会社が実施するサブプロセス

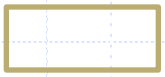


コア資産を元に現地法人が現地需要に合った製品を開発する。

FCや規格対応ものには適さない。

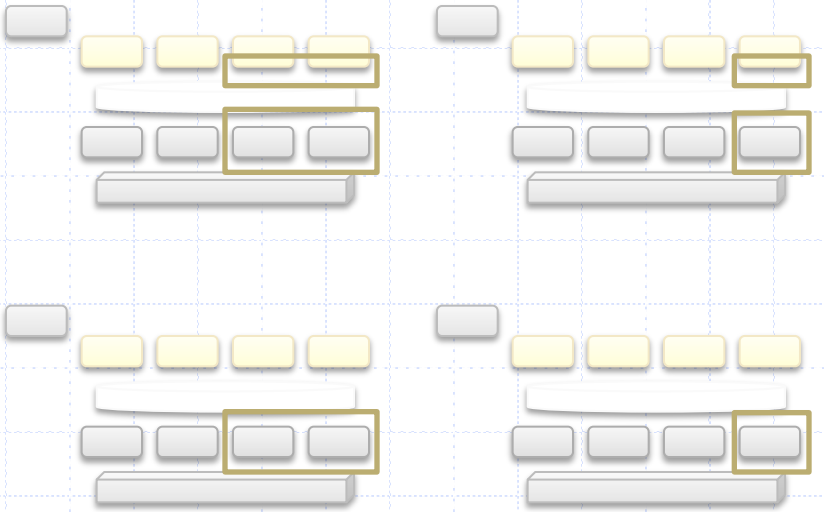
事業上、Worldwide戦略として、一部を固定して出す必要があるため。

研究立ち上げパターン

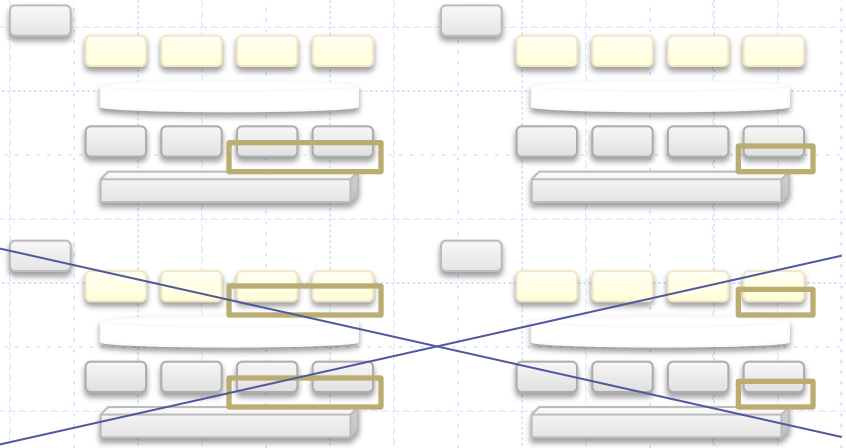


：子会社が実施するサブプロセス

子会社が高いSPL技術を持っていなければならない。



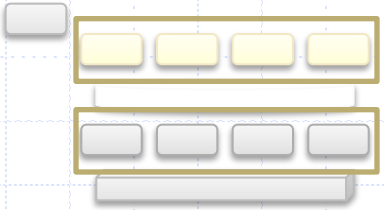
SPL技術不要



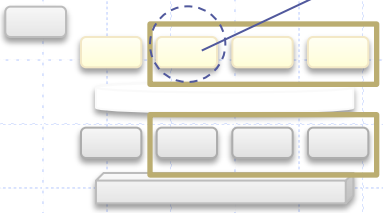
どうして×か。

研究開発には試行錯誤が必要。コア資産を意識していなくても、コア資産になる可能性もある。

開発任せパターン

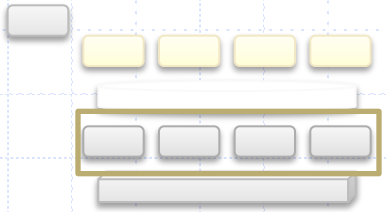


親会社は開発手法に
感知しないパターン



ODM開発パターン

可能ならば
レビューし、
将来に耐
えられる設
計になって
いることを
確認したい。



現地法人開発パターン

:子会社が実施するサブプロセス

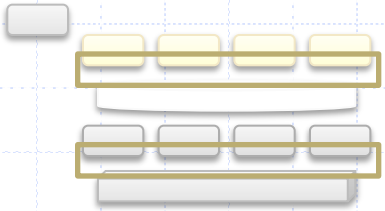
可変点要求開発技術を子会社に握られるのは、親会社は避けたほうがよい。

親会社は企画が実現できればよい、子会社は技術があれば満足、といったwin-win関係だと問題はない。

その他パターン

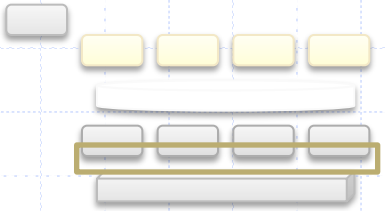
：子会社が実施するサブプロセス

SPL技術不要



現地法人でのコア資産開発。現地でのFC、規格対応での製品開発。自動製品構築。

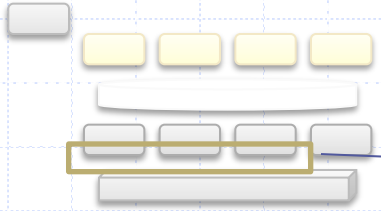
SPL技術不要



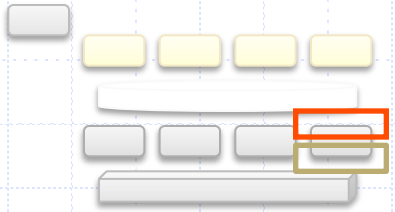
FC、規格対応による製品開発。自動的に製品構築。

※非機能用件のテストパターンは自動生成できない場合もある。

可変点固定後のアプリケーション開発



派生開発技術

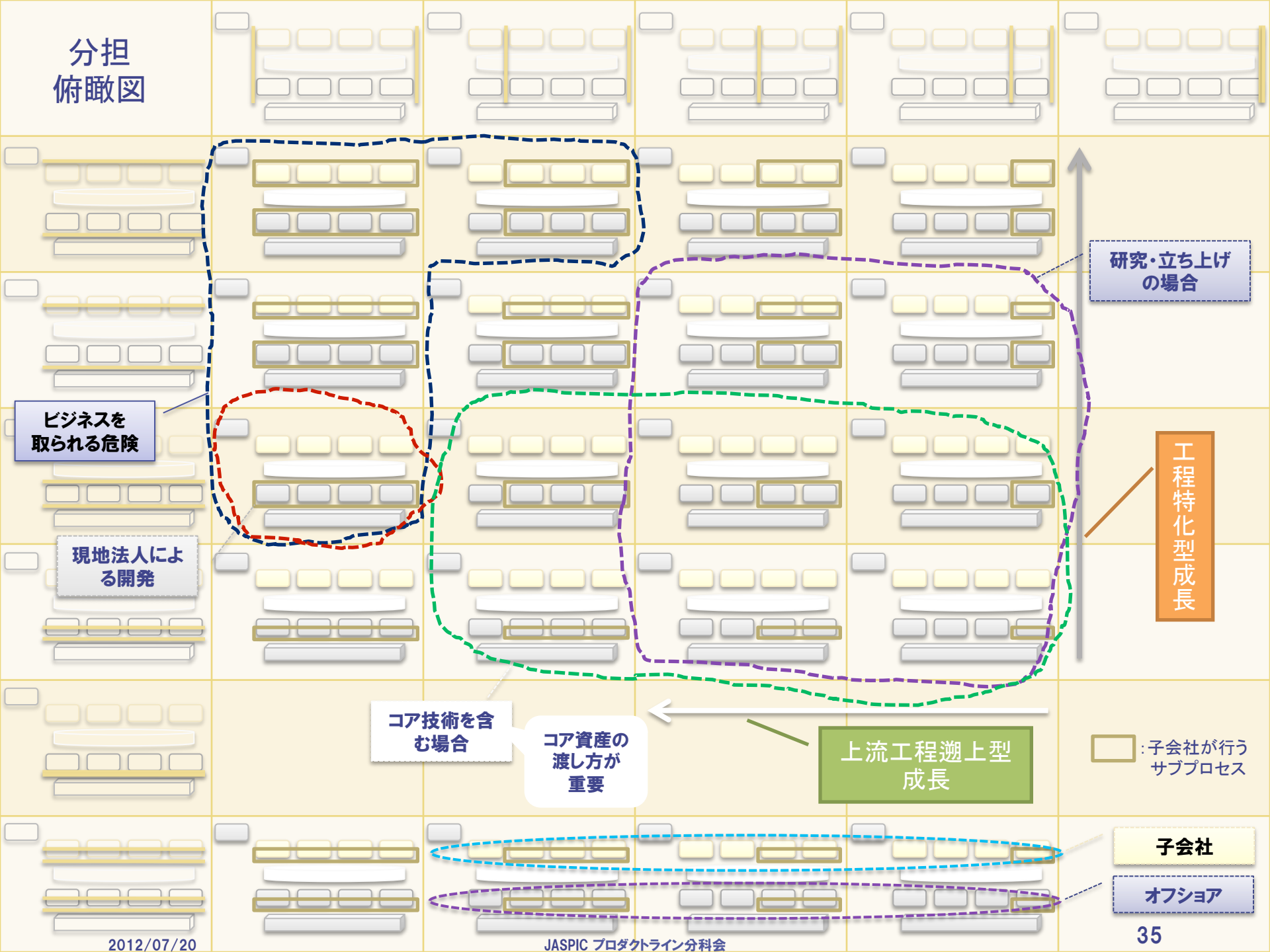


派生開発の試験設計技術だけでは弱い場合もあり。

不在変異体、ドメイン資産のアプリケーション依存性

故にSPL技術による全体試験設計が必要。

分担俯瞰図


研究・立ち上げの場合
ビジネスを取られる危険
現地法人による開発
工程特化型成長
コア技術を含む場合
コア資産の渡し方が重要
上流工程遡上型成長
:子会社が行うサブプロセス
子会社
オフショア

- SPLE の概要
- 始める前に(その1):企業間に跨る SPLE 実施の形態
- 始める前に(その2):各企業に適した SPLE 導入の形態
- 始め方:パターン、例、推奨形
- むすび

どのように移行するのか？

- SPLE導入の移行戦略（主要な4パターン）
  - 段階的移行
    - 試みを段階的に成功させ、組織と投資を拡大
  - 戦術的開始
    - 現在ニーズのあるプロセス/プラクティスからSPLEに対応させ、順次展開
  - パイロットプロジェクト
    - プロジェクトを限定して開始。組織リスクが比較的低い
  - ビッグバン
    - いちどきに移行。成功すれば最も効率が良い

* [Pohl05]に基づく。

導入戦略1―段階的導入

- 小規模に導入し、大規模に拡張
  - 一つのグループから組織全体へ　または
  - 小規模な投資から大規模な増強へ
    - 製品シリーズの一部、プロセスの一部、成果物の限られた種類への適用から始め、その範囲を拡げる
  - 例：Salion
- 長所(↑)、短所(↓)
  - ↑ 影響範囲は小規模で実施可能、現行事業は継続可能
  - ↑ 予算と期間が限定
  - ↑ 結果による計画変更・中止が容易
  - ↓ フルスケールに至るには長期間
  - ↓ 関連する現行製品の変更の影響
  - ↓ 関連する現行製品を統合する工数がかかる
  - ↓ 製品系列本来のメリットを引き出すのはビッグバンより長期

* [Pohl05]に基づく。

導入戦略2—戦術的アプローチ

- まず一部のプロセスに導入
  - 例えば関連製品群の統一的構成管理
  - 試験プロセスにおける仕様間共通性の活用
  - 例：九州日立マクセル
- 長所、短所
  - ↑ もっとも課題が意識されている領域にまず対応できる
  - ↑ 小規模なグループで開始できる
  - ↑ 移行の開始費用は低い
  - ↓ 全体的移行計画がなければフルスケールの適用は難しい
  - ↓ 問題が顕在化したプロセスを改善しても問題の原因であるプロセスが改善されなければあまり効果が上がらない
  - ↓ 製品管理を移行のプロセスに組み入れない場合、効果は限定的

* [Pohl05]に基づく。

導入戦略3ーパイロットプロジェクト

- 新規に製品を開発する際に導入
  - 製品系列の潜在的な最初の製品の開発
  - 関連製品群の拡張(と統一)
  - 試作/先行開発として
  - 例:Lucent Technologies
- 長所、短所
  - ↑ 現行製品に関する事業は継続可能
  - ↑ 展開前に先行開発品/試作品を用いて工数、プロセス、技術変更を確認できる
  - ↑ 結果による計画変更・中止が容易
  - ↓ 必要な資金、時間は概ね段階的導入よりも多い
  - ↓ 成果物は本番使用に供せないこともある
  - ↓ 製品系列本来のメリットを引き出すのはビッグバンより長期

* [Pohl05]に基づく。

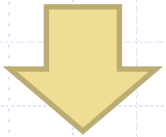
導入戦略4—ビッグバン

- 最初から本格運用
  - まずドメイン開発を完了させてプラットフォームを準備
  - 製品はプラットフォームから導出
  - 例：Cummins
- 長所、短所
  - ↑ プロセスと組織の各部分間の相互依存関係が最初から検討される
  - ↑ 投資の総量が一番小さい
  - ↑ 必要なドメイン資産の準備が他の戦略よりも早くになされる
  - ↓ 大きな先行投資が必要で、他の戦略よりも投資が短期間に集中
  - ↓ 組織が移行作業に時間を取られ、製品の生産に本来の時間を割くことができない恐れ
    - この点が重大な影響を与え、顧客が競合他社に乗り換えてしまうかも
  - ↓ 製品系列開発が適切なアプローチではないことが明らかになった場合でも、移行を戻すことは困難
    - その場合、資金的、工数的損失は相当のものになる

* [Pohl05]に基づく。

素朴な疑問？

- SPLE導入の移行戦略には、主要な4パターンがあることが分かった。



- では、我々の組織には、どの移行戦略が適しているのだろうか？

着眼点、検討内容

着眼点

- 一般には、事業環境や、組織構造から来る責任分担など、動機付けの面から、先ほどの長所・短所を考慮し、移行戦略の選択が論じられることが多い。しかし企業文化として、過去のプロセス改善への取り組みの形態やその結果を見れば、選択へのヒントが隠されているのではないか？

検討内容

- 組織のプロセス改善に対する文化の違いが、 SPLE導入の移行戦略に与える影響を考察し、組織の改善スタイルと移行戦略の相性を検討する。

予備検討：　なぜ関係が有りそうなのか？

- プロダクトラインの導入過程がプロセス改善そのものである
  - 導入／立ち上げ計画に、プロセス改善活動時の教訓を反映する必要あり
- プロダクトライン自体がプロセス改善を内包している
  - 製品自体の改善を目標に、製品系列が計画される
    - 空間的適合性：地域による違いに適合
      - 地域の購買能力に見合う、コスト低減（廉価版）
      - 特別仕様（必要、不要によりハイテク／ローテク切り分け）
    - 時間的適合性：発売時期による適合
      - EarlyAdaptor向け（先行開発、先行市場投入）
      - 技術の進歩への対応
        - 主としてHW製造技術の進歩（ SW製造技術の進歩もありか？、調整、テストとか）
          - 高性能化またはローコスト化
        - HW／SW開発技術の進歩
          - 開発コスト削減は、（出荷量に依存するが）製造コスト低減に比べて寄与が低い
          - HW／SW開発技術の進歩により実現可能な機能もある
        - 開発プロセス自体の進歩は？
          - 生産性（納期／コスト）／品質に影響するが、一般に技術進歩より寄与が低い？
          - 画期的新技術の導入とペアで、大きな寄与が見込まれる
            ⇒画期的新技術は、開発プロセスの大きな変更を要求することが多い

組織の改善スタイルの分類（３種類・４タイプ）

（Ａ）モデル参照型プロセス改善

・CMMI、ISO等のプロセスモデルを参照して、プロセスを改善するやりかた
一般にトップダウンで行われることが多いが、まれに、現場からのボトムアップで行われることもある　理想（手本）とのGapを埋める活動が基本

（Ｂ）現場課題解決型プロセス改善

BA）共通目標型（トップダウン的）

生産性向上や品質向上などの組織の共通目標から、現場が改善に取り組むもの、
共通の施策を用意して組織的に行われる場合BA１）と
施策自体は現場に任せて、個別に行われる場合BA２）がある

BB）現場自発型（ボトムアップ的）

現場が抱える課題の解決に、現場が主体的に取り組む活動
すでに問題として発生していることへの対応が主だが、まれに継続的に課題を見つけて自発的に改善に取り組んでいる場合もある

（Ｃ）技術革新連続型プロセス改善　⇒技術的イノベーション駆動型

・短い周期の技術革新が事業継続に不可欠で、常に新技術を取り入れてプロセスも同時に変えなければ生き残れない業界のプロセス改善は難易度が高い
新技術を常に自主開発できないので、多くの場合、ほかで産まれた新技術を取り入れることになるが、理解や経験が足りない状態での取り組みを強いられる

詳細検討内容

プロダクトライン導入時における組織の改善スタイルと
移行戦略との整合性を検討し、それぞれの得失を列挙する

・組織の改善スタイル（４タイプ）　×　・移行戦略（４パターン）

- モデル参照型
- 現場課題解決型（トップダウン）
- 現場課題解決型（ボトムアップ）
- 技術革新連続型

- 段階的導入
- 戦術的アプローチ
- パイロットPJ戦略
- ビックバン戦略

の４×４＝１６通りの組み合わせに関して、詳細を検討

検討結果(一覧表)

移行戦略→ ↓改善スタイル	段階的導入	戦術的 アプローチ	パイロットPJ 戦略	ビックバン 戦略
モデル参照型	×:組織や製品の部分・段階適用は経験知少	△:モデル参照で解決できない技術課題が問題	△:パイロットPJの解決力が鍵、組織支援必要	△:緻密な計画が必要、目的を見失う危険も
現場課題解決 トップダウン型	△:部署考慮の段階的計画と組織支援が鍵	○:共通課題選択が鍵、当たれば展開は順調	△:パイロットPJ選定が鍵、現場に解決力あるか	○:共通目標に一気に全体で取り組める
現場課題解決ボトムアップ型	○:問題意識と改善力ある部署から徐々に	△:初期成果を得やすいが、展開時の調整が難	○:現場課題と組織課題が一致すれば効果的	×:部署の独立性が高く、利害調整が困難
技術革新連続型	×:段階的な導入は苦手、組織や製品の部分適用は困難	△:優先課題と導入技術が一致するか？段階的な改善は苦手	○:狭い範囲の試験的導入は得意、展開時の順序に工夫要	◎:現場が慣れていればベスト、不慣れなら混乱して大失敗も

検討結果(改善スタイル別:その1)

(A)モデル参照型プロセス改善

- CMMI、ISO等のプロセスモデルを参照して、プロセスを改善するやりかた
  一般にトップダウンで行われることが多いが、まれに、現場からのボトムアップで行われることもある　理想(手本)とのGapを埋める活動が基本

×段階的導入
　組織や製品の部分・段階適用は経験知少
　⇒　理想とのGapを認識しても改善策に結びつきにくい、段階設定に手本なし

△戦術的アプローチ
　モデル参照で解決できない技術課題が問題
　⇒　モデル参照は、全体としての効果は期待できるが、特定の課題への効果は？
　　モデルの一部分と優先課題が一致すれば効果も

△パイロットPJ戦略
　パイロットPJの解決力が鍵、組織支援必要
　⇒　手本が少なく、パイロットPJの課題が解決できるか？組織全体からの支援必要

△ビックバン戦略
　⇒　組織全体で一気に取り組む方式には慣れているが、手本が少なく、改善策立案や計画策定で苦労しそう　　改善策実施という手段を目的と混同する危険も

検討結果(改善スタイル別:その2)

(B)現場課題解決型プロセス改善

BA)共通目標型(トップダウン的)

生産性向上や品質向上などの組織の共通目標から、現場が改善に取り組むもの、
共通の施策を用意して組織的に行われる場合BA1)と
施策自体は現場に任せて、個別に行われる場合BA2)がある

△段階的導入

部署考慮の段階的計画と組織支援が鍵

⇒　部署の解決力を見極めた段階的な計画と組織として効果的な支援策が必要

○戦術的アプローチ

共通課題選択が鍵、当たれば展開は順調

⇒　組織として共通の課題を選択し、改善効果が挙げられれば、活動展開に弾み

△パイロットPJ戦略

パイロットPJ選定が鍵、現場に解決力あるか

⇒　パイロットPJの現場に解決力ある？ない場合は組織で効果的な支援が可能？

○ビックバン戦略

共通目標に一気に全体で取り組める

⇒　BA1タイプは同じやり方なので、組織として目標成果達成経験があれば有利、
BA2タイプは施策の統一経験がないので、部署間の協調・連動に不安が残る

検討結果(改善スタイル別:その3)

(B)現場課題解決型プロセス改善

BB)現場自発型(ボトムアップ的)

現場が抱える課題の解決に、現場が主体的に取り組む活動
すでに問題として発生していることへの対応が主だが、まれに継続的に課題を見つけて自発的に改善に取り組んでいる場合もある

○段階的導入

問題意識と改善力ある部署から徐々に
⇒　機能別組織であれば、イノベータ→アーリーアダプタと徐々に全体へ適用可能

△戦術的アプローチ

初期成果を得やすいが、展開時の調整が難
⇒　ボトムアップに慣れているため個々のプロセス課題改善時に効果を得やすいが組織全体へと展開する際の立案・実行において現場間の調整が難しい

○パイロットPJ戦略

現場課題と組織課題が一致すれば効果的
⇒　製品別組織であれば、新規製品開発部署をパイロットPJとして徐々に全組織へ

×ビックバン戦略

部署の独立性が高く、利害調整が困難
⇒　ボトムアップに慣れているため独立性が高く、組織全体の利害調整が困難

検討結果(改善スタイル別:その4)

(C)技術革新連続型プロセス改善　⇒技術的イノベーション駆動型

・短い周期の技術革新が事業継続に不可欠で、常に新技術を取り入れてプロセスも同時に変えなければ生き残れない業界のプロセス改善は難易度が高い
新技術を常に自主開発できないので、多くの場合、ほかで産まれた新技術を取り入れることになるが、理解や経験が足りない状態での取り組みを強いられる

×段階的導入
段階的な導入は苦手、組織や製品の部分適用は困難
⇒　事業継続しながらの導入になるので、中途半端な技術導入となり効果が薄い

△戦術的アプローチ
優先課題と導入技術が一致するか？段階的な改善は苦手
⇒　優先課題と導入技術が一致すればよいが、通常は難しいのでは？

○パイロットPJ戦略
狭い範囲の試験的導入は得意、展開時の順序に工夫要
⇒　従来製品の開発との並行作業なので、共通部品化をどこまで進められるか？

◎ビックバン戦略
現場が慣れていればベスト、不慣れなら混乱して大失敗も
⇒　一発勝負で失敗できないので、過去のリスク回避ノウハウがどれだけあるかが成否の分かれ目。そもそもそれが乗り越えられなければ生き残っていない？

まとめと考察

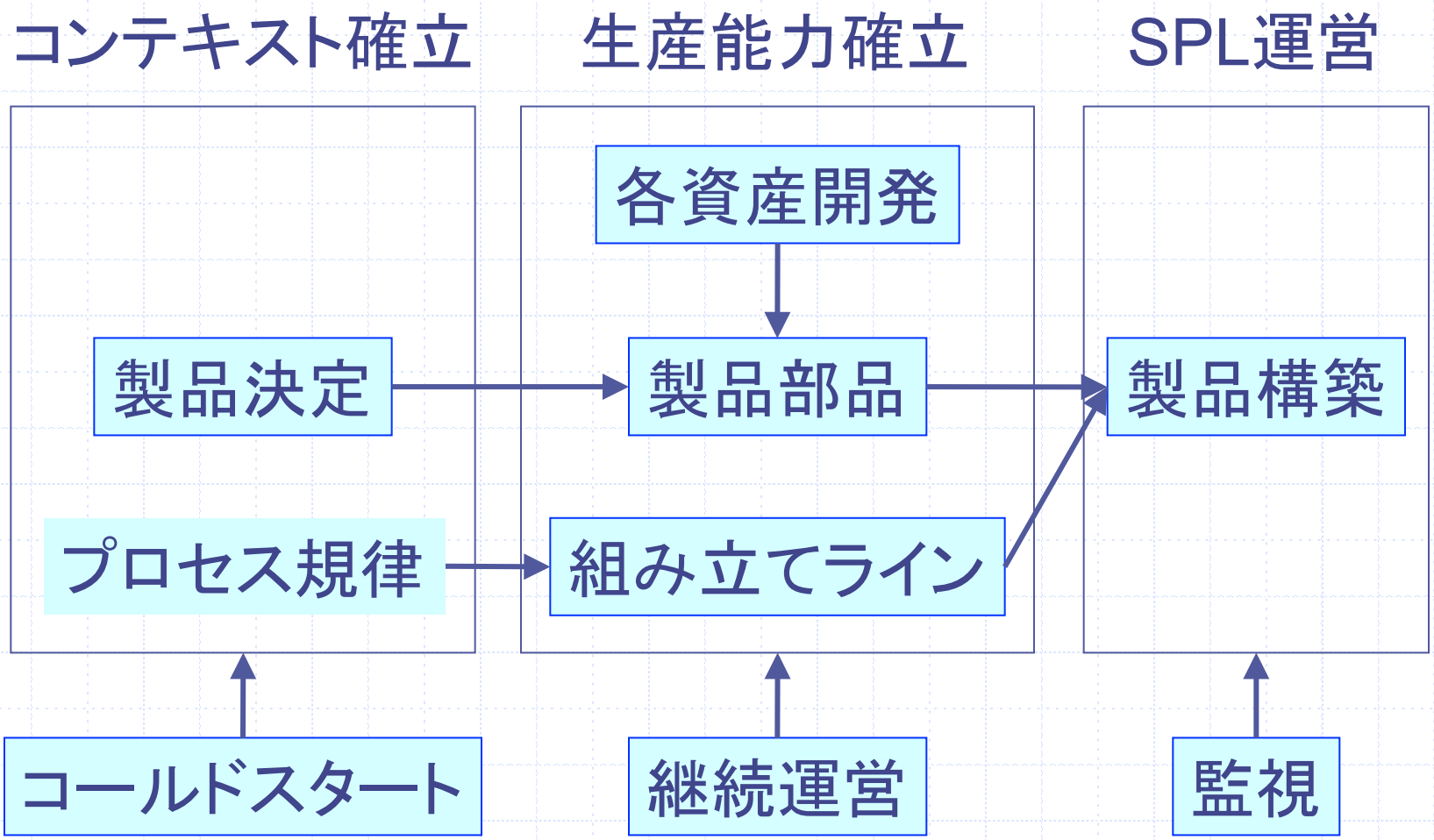
- モデル参照型改善に慣れた組織は、移行戦略によらず技術課題の解決力が問題となる、参照する例がないと戸惑って混乱する場合もありそう。
- 現場課題解決型改善を行っている組織は、比較的、プロダクトライン導入に対しても力を発揮する確率が高い。ボトムアップ型は、取り組みの最初の部分で優位性を発揮するが、組織全体への適用の段階では苦戦しそう。
  逆に、トップダウン型は、どの組織や製品から始めるかの選択に苦労しそう。
  改善の組織への展開には優位性がある。
- 技術革新連続型改善を行っている組織は、普段から改善活動がうまく回っていれば、ビッグバン戦略に対応でき、効果を最大限発揮できる。逆に、改善活動が後手に回っている組織は、どの移行戦略にも対応が難しい。
- 改善スタイルは組織風土を反映していると思われるので、その活動の過去の成果や失敗経験もプロダクトライン導入時には考慮が必要と考えられる。
- 途中からの移行戦略の転換は、効果が見えた時点でビックバン戦略へは有りか。
  ※移行戦略４パターン定義を排他的と捉えると転換はないが拡張解釈するとO.K.か。組織の改善スタイルの進化はありえるので、可能性はあると考える。

- SPLE の概要
- 始める前に(その1):企業間に跨る SPLE 実施の形態
- 始める前に(その2):各企業に適した SPLE 導入の形態
- 始め方:パターン、例、推奨形
- むすび

導入ファクトリパターン

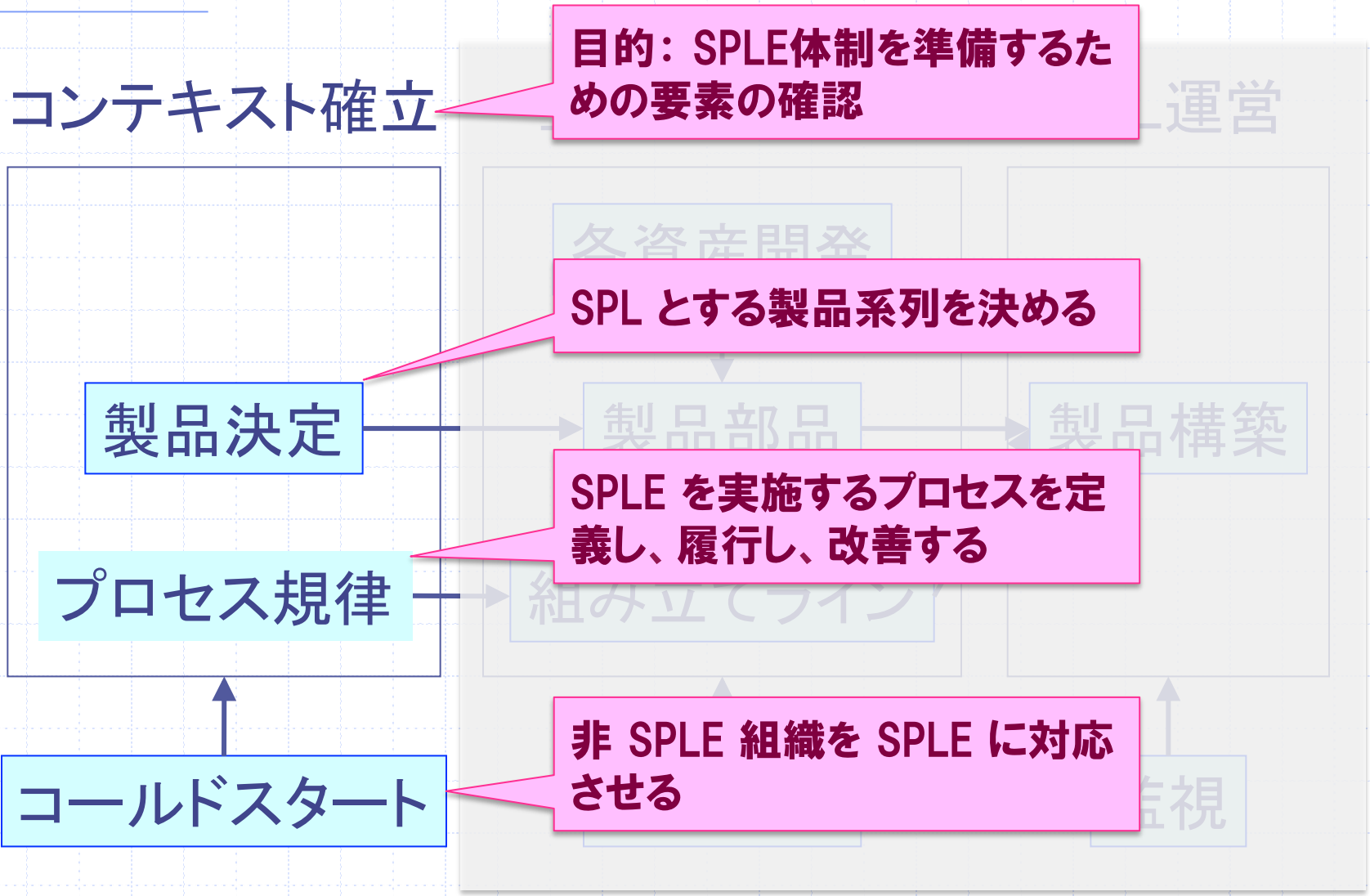
- 目的：SPLE体制を確立するための要素の確認
  - これらの要素を戦略に従って選択し、順序付けて、計画を策定する
- 大きく分けて次の要素からなる
  - コンテキスト確立
    - SPL 開発に向けた調査・分析と準備
  - 生産能力確立
    - SPL で製品を生産できる技術資産と体制を揃える
  - SPL 運営期
    - 定常的に SPL を運営して製品を世に出していく

導入ファクトリパターン



[SEI07] より編集。

導入ファクトリパターン



[SEI07] より編集。

導入ファクトリパターン

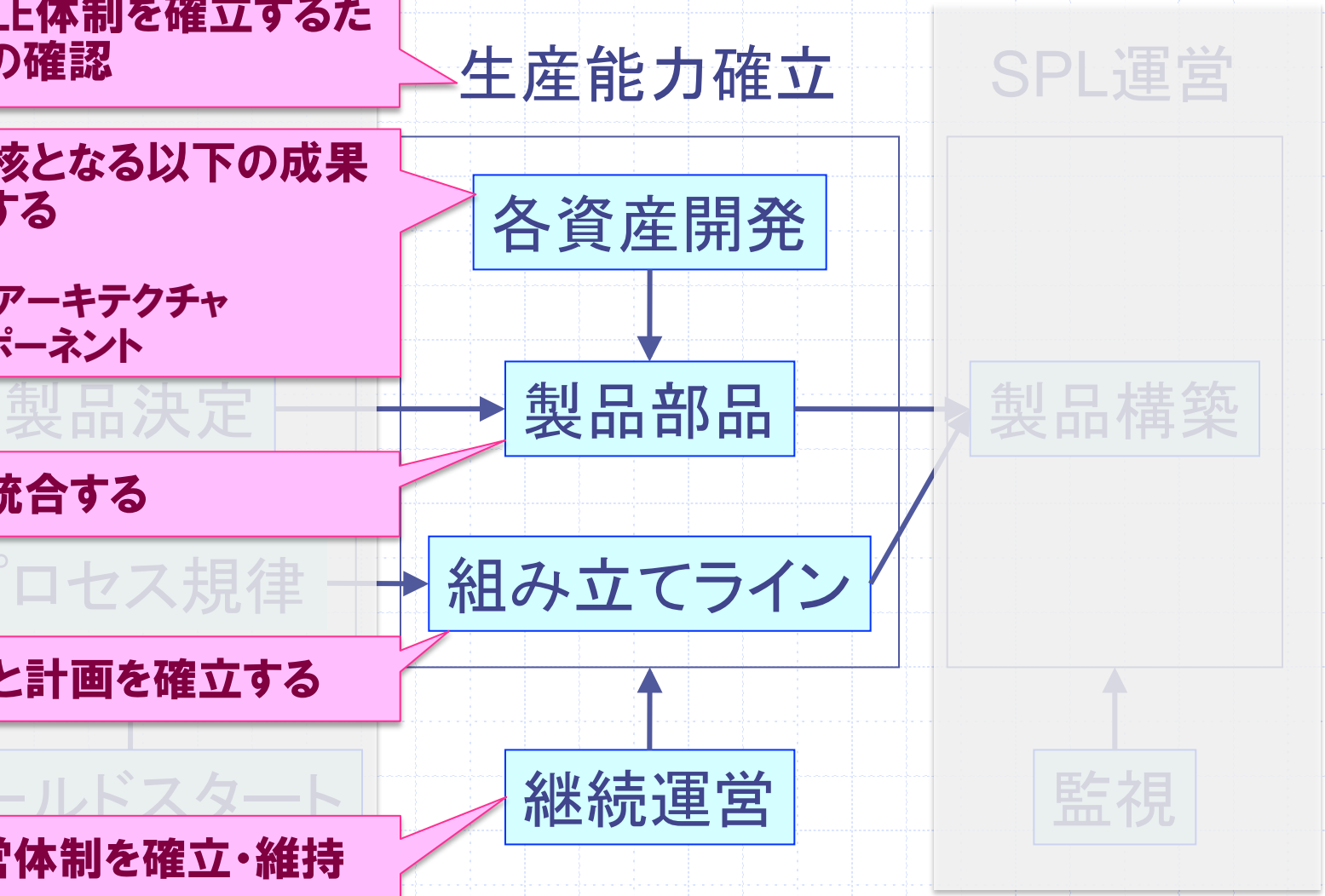
目的：SPLE体制を確立するための要素の確認

SPL の中核となる以下の成果物を開発する
- 要求
- 参照アーキテクチャ
- コンポーネント

各資産を統合する

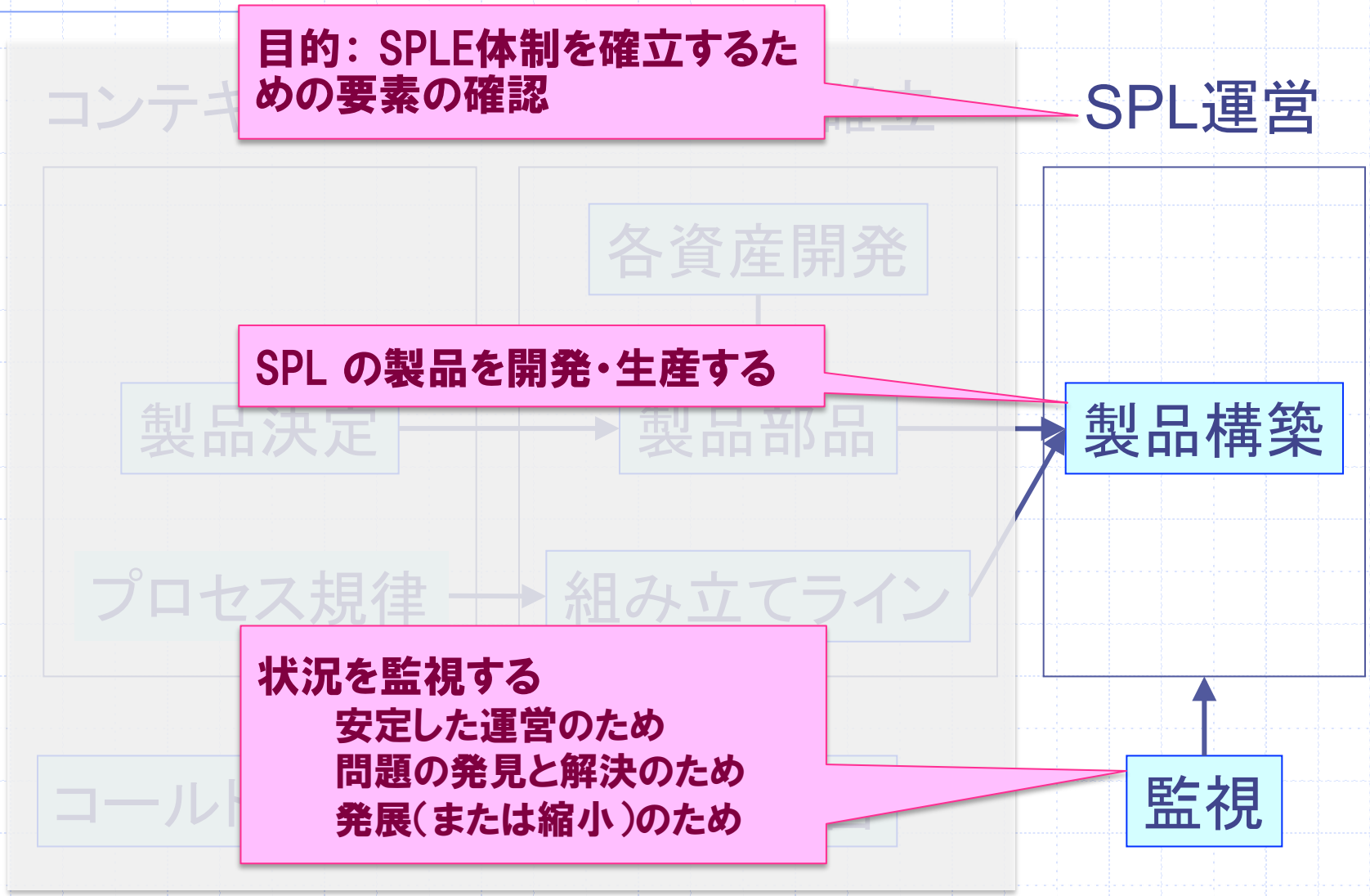
開発体制と計画を確立する

SPLE 運営体制を確立・維持する



[SEI07] より編集。

導入ファクトリパターン



[SEI07] より編集。

移行ステップ例：段階的移行 1/7

全体像：

- 調査・分析
  - コールドスタート（一部）
  - 製品決定
  - 製品部品（一部）
- 準備
  - コールドスタート（残り全部）
  - プロセス規律
  - 製品部品（一部）
- 移行
  - 継続運営（一部）
  - 各資産開発（一部）
  - 製品部品（一部）
  - 組み立てライン（一部）
- 維持・発展
  - 継続運営（残りを継続的に）
  - 各資産開発（残りを継続的に）
  - 製品部品（残りを継続的に）
  - 組み立てライン（残りを段階的に）
  - 製品構築
  - 監視

移行ステップ例：段階的移行 2/7

- **調査・分析**

SPLE が導入に足るものであることを確認する

- コールドスタート（一部）
  - 調査分析チームを立ち上げ、以下の作業を行なう
  - ドメイン開発チーム（部門）とアプリケーション開発チーム（部門）を現存の組織に当てはめる
    - ※注意
      - このとき、後の移行をにらんで、事業目標からドメイン開発、アプリケーション開発まで切れ目なく連携できるようにするために必要な組織上の役割の強化・改変は何であるかを確認しておく
  - 準備、移行、維持・発展のための大まかな計画を立案する
    - SPLE　対応のためのトレーニング計画を含む
  - これらの活動計画を基に資金計画を立て、承認を受ける

※注意

- 活動計画と資金計画では当調査・分析の部分のみ詳細に立案し、準備以降の計画は大まかなものに留める
  - 先の方まで詳細に計画すると、それによって制約を受けるリスクが高まるため
  - 飽くまで調査・分析の活動であることを忘れないようにする

移行ステップ例：段階的移行 3/7

- 調査・分析（続き）
  - 製品決定
    - 市場分析、技術予測を基に、SPL の範囲（何を含み何を含まないか）を定義する
    - ビジネスケースを作成し、従来の方法よりも事業面で益があることを示す

    ※注意
    - SPL 対象範囲はなるべく経験の深い方面に絞る
      - そこには過去から蓄積された資産がある
      - 過去から事業をしてきた領域は一般に安定しており、安全な投資先である
  - 製品部品（一部）
    - 製品決定の結果に沿って、既存ソフトウェア資産を整理し、初期のコア資産として利用できるものを識別する

    ※注意
    - コードだけではなく、要求定義、アーキテクチャ、コンポーネント設計等、ソフトウェアに関連する全ての資産を対象とする
    - 設計と実装を分析する際、特に非機能要求の共通性に留意する
      - 機能が同一であっても、前提としての要求（例えばメモリ消費量、初期化にかかる時間、性能等）が異なると、同一の資産として扱うことはできない
  - 反復
    - 識別した既存資産が、製品決定の結果を裏打ちできるかどうかを判断する
      - 作成したビジネスケースが有効であることを、主に再利用によるコスト、工数、開発期間の節約度合いから検証する
    - 判断に基づいて製品決定または製品部品の作業を調整する
    - この反復を満足のいく結果が得られるまで繰り返す

    ※注意
  - 既存資産の分析を製品決定に先立って行なってはならない
    - 明確な基準（範囲）がなければ資産の分析はいい加減なものになる

- 上の反復で満足のいく結果が得られなかった場合、SPLE 導入を断念する

移行ステップ例：段階的移行 4/7

- **準備**

SPLE 導入の様々な準備を行なう

- 引き継ぎ
  - 準備チームを立ち上げ、作業を調査・分析チームから引き継ぎ、準備の詳細計画を立案する
- コールドスタート（残り全部）
  - 調査・分析の際と同じように移行、維持・発展のための大まかな計画を立案する
    - SPLE 対応のためのトレーニング計画はさらに重要になる
  - それらの活動計画を基に資金計画を立て、承認を受ける
- プロセス規律
  - SPLE 実施プロセスを定義する
    - 当座の目標としては FEF の全軸レベル3を据える
    - プロセスの詳細は［Pohl05］が参考になる
  - 定義したプロセスが履行されるようにする
    - 通常のプロセス改善と同じ

- 製品部品（残り全部）
  - 各コア資産の開発（または購入、発掘、外注のいずれか）を、定義したプロセスに従って行なう（ドメイン開発）
    - この中の「発掘」が既存資産の整備に当たる
    - 典型的にはまず参照アーキテクチャを定義し、その構造と作法（ルール）に従うインタフェースをコンポーネントに与える
    - その他にも要求定義当、他の資産と参照アーキテクチャとの関係を明確にする
    - 最重要事は SPL の全製品の共通性を識別し、それらに相当する資産を識別することである

※注意

- **当座の目標は共通性が利用できるアーキテクチャレベル3 であり、厳密でないインタフェース定義を持ち込んで無理に可変性を定義してしまおうとしないこと**
  - 曖昧なインタフェース定義はのちのすり合わせの工数およびそれによる手戻りにより SPLE の益を減じることが報告されている

- 準備の作業中、状況の変化により調査・分析の結果が影響を受けるようであれば、結果を再評価し、SPLE 導入が適切でないと判断されたら導入を中止する

移行ステップ例：段階的移行 5/7

- 移行

SPLE 体制を整える

- 引き継ぎ
  - 移行チームを立ち上げ、作業を準備チームから引き継ぎ、移行の詳細計画を立案する
- 継続運営（一部）
  - 継続的な SPLE 運営の枠組みを作る
    - 資金供給、組織編制、トレーニング、調達の仕組み作り・役割割り当て

※注意

- この段階で、事業目標に沿って事業戦略を策定する活動と移行の企画および開発活動がスムーズに繋がる仕組みを確立する
  - 次の維持・発展の段階では SPLE の定常的な運営を行なうため、その段階で漏れや手戻り等が発生すると致命的であるため

- 各資産開発（一部）
  - 定義されたプロセスに基づいてドメイン要求・設計・実現を用意する
    - 既存資産でまかなえないものは新規開発／調達する
- 製品部品（一部）
  - 初期製品群に充分なだけコア資産を統合し試験する
- 組み立てライン（一部）
  - 統合されたコア資産を、製品計画およびそれを支える生産計画に則り、アプリケーション開発に供給できるようにする

- 移行の作業中、状況の変化により調査・分析の結果が影響を受けるようであれば、結果を再評価し、SPLE 導入が適切でないと判断されたら導入を中止する

移行ステップ例：段階的移行 6/7

維持・発展

SPLE を継続する

- 継続運営（残りを継続的に）
  - 移行チームから作業を引き継ぎ、定常的なコア資産開発の運営に吸収する
  - SPLE 運営の枠組みを維持する
    - 資金供給、組織編制、トレーニング、調達の続行

※注意

- この段階では、事業目標に沿って事業戦略を策定する活動と移行の企画および開発活動がスムーズに連携されていなければならない

- 各資産開発（残りを継続的に）
  - 製品計画に即した生産計画に則り、コア資産に必要な要求、設計、実現を提供していく
- 製品部品（残りを継続的に）
  - 前項と同じく、計画に則ってコア資産の統合を続けていく
- 組み立てライン（残りを継続的に）
  - 同じく、アプリケーション開発にコア資産を継続的に提供していく

※注意

- 上記三つの活動は、種類としては移行時のものと変わらないが、継続運営による計画や枠組みの変更および状況（市場動向や新技術の採用等）と、それらを反映した事業戦略に、常に沿うように留意する

移行ステップ例：段階的移行 7/7

- 維持・発展（続き）
  - 監視
    - データ収集・分析を恒常的に行なう
    - データ分析の結果を受け、リスクを管理し、必要に応じて計画やプロセスの改訂を行なう
      - 維持・発展の活動はこれらのリスク管理および計画に基づく

    ※注意

      - 維持・発展の途中でも、状況の変化により、SPLE 継続が事業にとって適切でないと判断されたら、SPLE 運営を打ち切る
  - 製品構築
    - 計画とコア資産に基づいてアプリケーション開発を行なう
      - 計画の続く限りこの活動は継続する

自分が始めるとき ― 組織面

- トップダウン：上から「SPLEをやれ」と言われた
  - 予算とある程度の組織内調整が期待できる
  - 具体的な目標は不明確かも知れない

  →やりたくない部分もあるかも知れないが説得不要で動ける部分もある
- ボトムアップ：自分は管理者でないがSPLEを適用したい
  - 予算獲得や組織内調整を自分で行なう必要がある
  - 上層部ではなく自分で判断した課題から着手できる

  →面倒だがやりがいはある
- ミドルアウト：中間管理層・リーダクラスから上層部と技術者を巻き込む
  - 状況はボトムアップの場合に似る

  →中間層には課題認識があることも多く、「何とかしなくては」という思いを共有しやすい

自分が始めるとき ― 技術面

- 多くの場合、人は多大なリスクを負おうとしない
- であれば、最初は小さい範囲での適用を提案する
  - パイロットプロジェクト、段階的導入、**戦術的アプローチ**
    - 戦術的アプローチは SPLE の完全適用に至らないリスクがあるが、逆にSPLEを謳わなくても開始できる可能性がある
- あるいは自分ひとりで始めてしまう

何から始めるか？ ― いくつかの例

- 同じアプリケーションでありながら OS のバージョンごとに細かい処理が異なるので、それらの部分を `#ifdef` で区切ってビルド時にそれぞれの OS 向けのバイナリが得られるようにした
- どう開発したかは知らないが類似のアプリケーションをいくつも試験する仕事を支援することになり、試験ケースが極力共通になるようにした
- 仕様変更が頻繁に起き、その度に作り直して試験をしなくてはならないので、試験のよく変わる部分とそう変わらない部分を識別して、前者を抽象化した
  - 試験と要求の間には強い関連があるので、さらに要求の一部抽象化も行ない、仕様変更の度に広範囲の記述を変更しなくて良いようにした
- 上層部から、実装をお願いするパートナー企業を機種ごとに替えても大丈夫なようにしてくれと言われたので、パートナーに受け持ってもらうことを想定した部分と自分達が手がける部分の間にインタフェースを設定し、互いの間の依存性を低減した

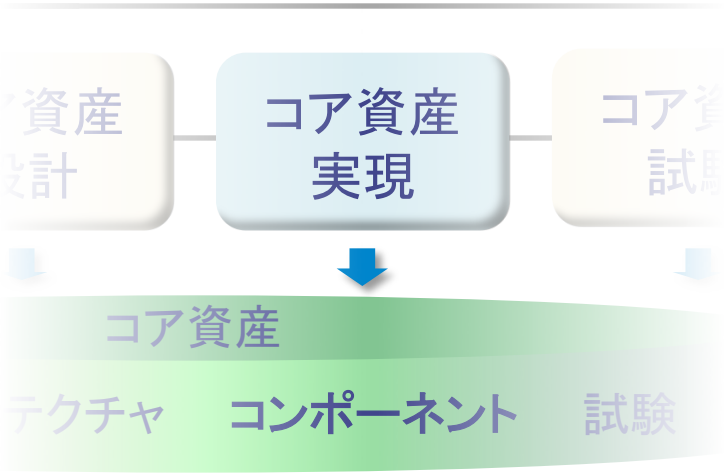
例では何を再利用/共通化したのか？

- #ifdef による区別
  → ソース(の一部)の PL 化
- 試験ケースの共通化
  → 試験ケースの PL 化
- 試験ケースと要求の抽象化
  → 試験ケースと要求の PL 化
- インタフェースの設定によるソフトウェアの切り分け
  → 設計(アーキテクチャ)の PL 化

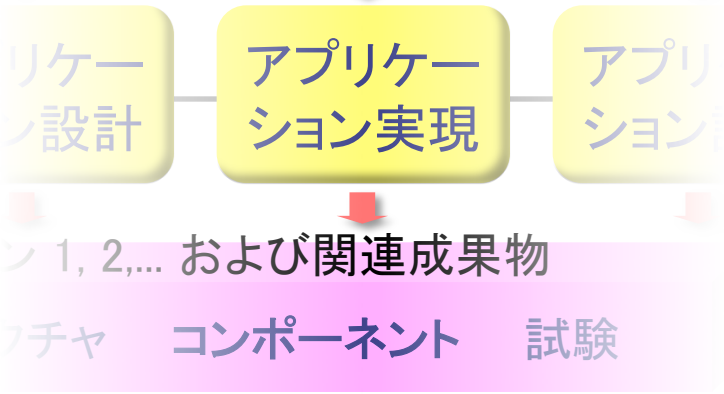
トレンド — Partial SPLE

ドメイン開発

- 一部の資産を PL 化
- その後、範囲を広げる
  - 範囲：同種の成果物の他の部分／他の成果物の種類／他のサブプロセス／他のプロジェクト／…



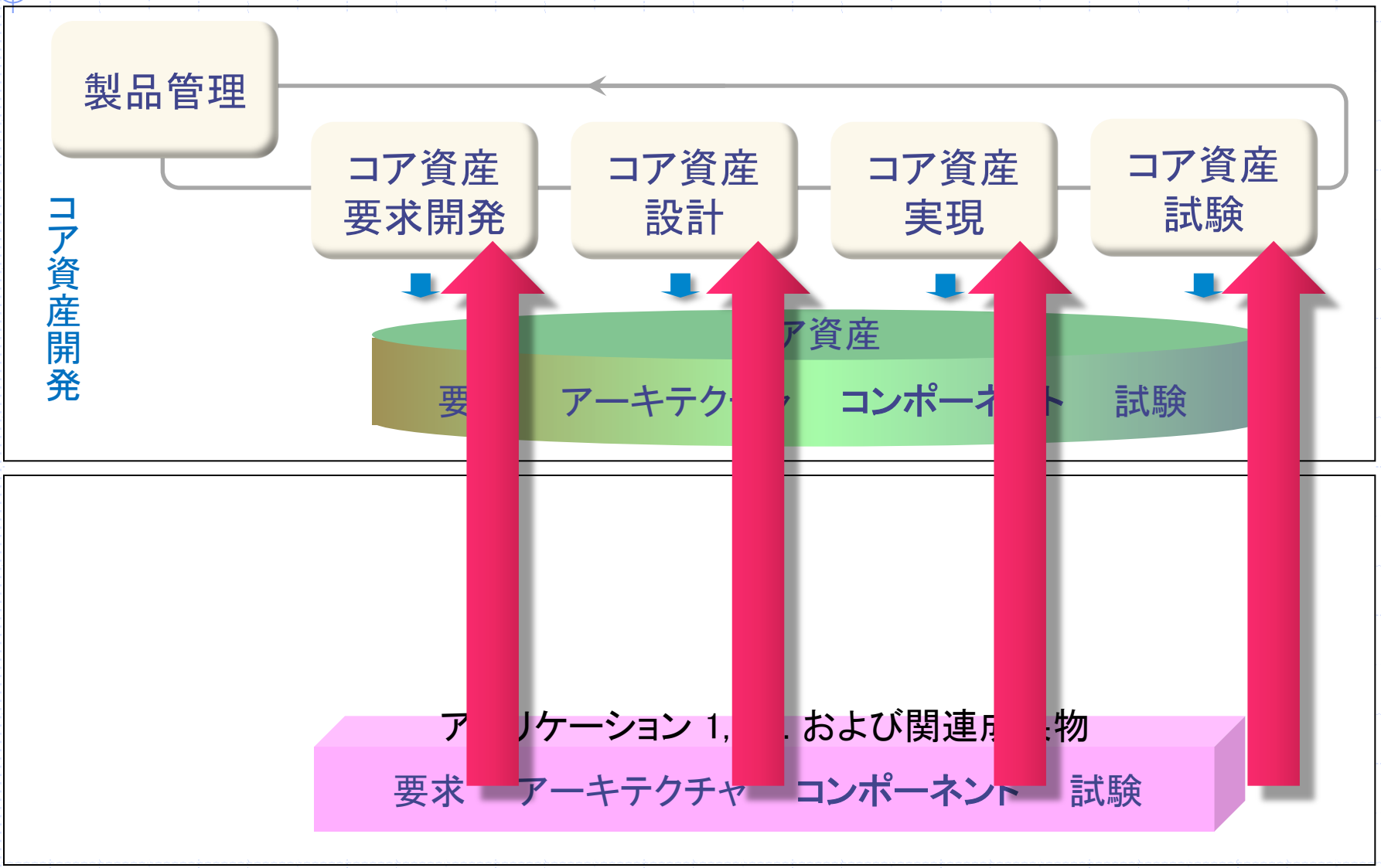
アプリケーション開発



現実的アプローチ ― 抽出式 SPLE

- 既存の開発成果物から共通化できる/すべきものを抽出しコア資産第1版とする
  - 必要に応じて可変性を持たせて
- 以後はそれを進化させていく
  - 可変性の追加
  - 新規コア資産の追加
  - コア資産の保守

現実的アプローチ ― 抽出式 SPLE

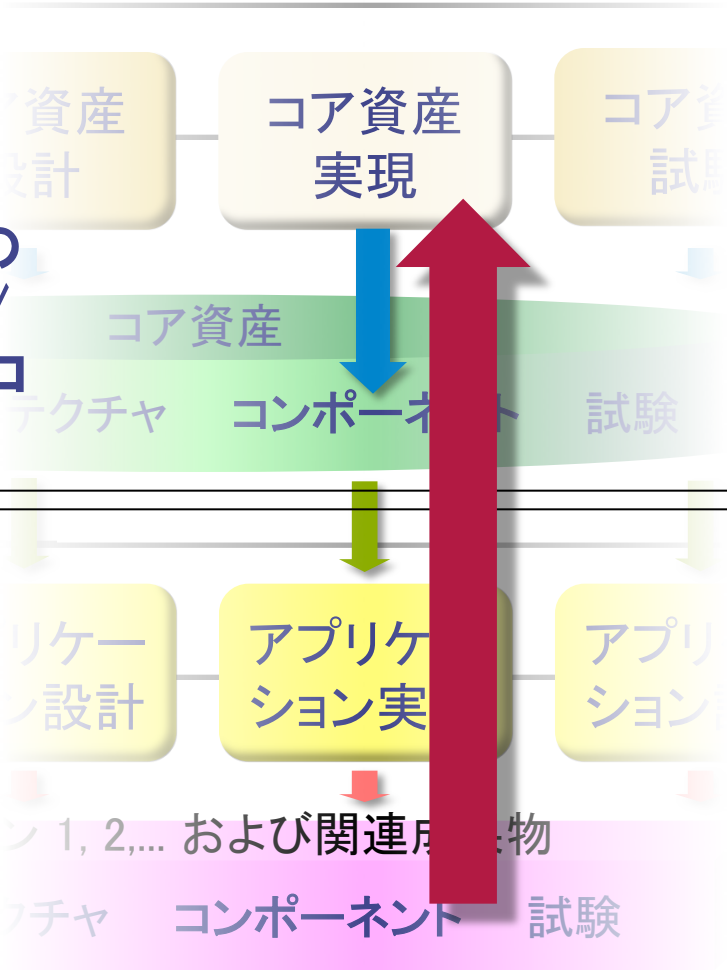


[Pohl05]を基に編集。

トレンディな現実的アプローチ ― Partial+抽出式 SPLE

コア資産開発

- 一部の資産を抽出式で PL 化
- その後、範囲を広げる
  - 範囲：同種の成果物の他の部分／他の成果物の種類／他のサブプロセス／他のプロジェクト／…



アプリケーション開発

詳細設計/ソースコードからコア資産にする例 →

[Pohl05]を基に編集。

で、その後は？

- 「効率」は事業に結びついて初めて意味を持つ
  - 闇雲な多種対応には意味がない
- 類似のアプリケーションを開発するプロジェクト/チームが複数あるのであれば、その間でも共通化がなされた方が効率が良い
- プロジェクト/チーム内での活動とプロジェクト/チーム間での活動には違う種類のマネジメントが必要になる
- Partial SPLE に着手する際には、どの範囲を目標にするかについて想定しておく
  - プロジェクト/チーム内、部門内、事業部内、全社、企業間

で、その後は？（補足）

- 利害関係者の思惑・動きに依って仕様が決まる・変わる
- 要求の可変性分析、さらには利害関係者分析が望まれる
  - プロジェクト/チーム内での実施でも、そのプロジェクト/チームの持つ要求や至近の利害関係者が影響するという点では同じ
  - ヒントは「どの範囲をカプセル化できるか」
  - つまりどこにインタフェースがあるか/持つべきか、そしてそのインタフェースのむこうとこっちで何をどれくらい独立に変えられるか
    - 例：ヘッダファイル、人、プロセス、契約、…
  - 事業面では社内の企画/提案活動を行なう人達が一番影響力を持つ

一度企画/提案チームと I/F をとってみては…？

- SPLE の概要
- 始める前に(その1):企業間に跨る SPLE 実施の形態
- 始める前に(その2):各企業に適した SPLE 導入の形態
- 始め方:パターン、例、推奨形
- むすび

むすび

- ソフトウェアプロダクトライン開発の鍵は高い共通性
- しかし、共通性は
  - 何を(共通とする要求・設計・コード・・・)
  - いつ(システム開発時／前／後)
  - 誰が、どこで(分担)
  - どうやって(技法)
  - そして、何故(最終的には事業に結び付けて)

  定義し達成するのかに関しては、様々な形がある
- 文献の説明や他所の成功例をそのまま適用するのではなく、自分達の情況を勘案した上で検討すべきである

参考文献

[Abeta08]
安部田 章、組込みシステム産学官連携技術交流会in熊本－プロダクトライン開発ワークショップ講演資料、2008

[Clements01]
Paul Clements, Linda Northrop; “Software Product Lines; Practices and Patterns”; Addison-Wesley, 2001 ［邦訳：前田 卓雄 訳『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事業戦略と実践』日刊工業新聞社］

[Ishida07]
Yuzo Ishida; "Software Product Lines Approach in Enterprise System Development", In: Proceedings of the 11th Software Product Line Conference, Kyoto, Japan, September 10-14, IEEE Computer Societyt, 2007, pp. 44-53.

[Iwasaki11]
岩崎 孝司、「ソフトウェア・プロダクト・ライン開発手法の実践的導入事例(1)」、CQ出版社 Tech Village、2009/09/16、http://www.kumikomi.net/archives/2011/09/ep04spl1.php?page=3

[JASPIC08]
日本SPIコンソーシアム（JASPIC）プロダクトライン分科会2008年度成果物（JASPIC会員にのみ公開）

[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］

[Rotibi10]
Bola Rotibi, Ed., “Podcast summary: Managing quality in Product Line Engineering,” August 2010 (http://www.biglever.com/extras/PLE_Automotive_report.pdf)

[SEI07]
Linda M. Northrop, Paul C. Clements, "Launching and Institutionalizing," in "A Framework for Software Product Line Practice, Version 5.0," Software Engineering Institute, Pittsburgh, U.S.A., July 2007 (http://www.sei.cmu.edu/productlines/frame_report/launch.inst.PL.htm)

[Takizawa09]
滝沢 治, 赤石 富士雄, 早瀬 健夫, 「ソフトウェアプラットフォーム構築技術」, 東芝レビュー , Vol. 64, No. 4, pp36-39, 2009 (http://www.toshiba.co.jp/tech/review/2009/04/64_04pdf/b03.pdf)

[vanderLinden07]
Frank van der Linden, Klaus Schmid, Eelco Rommes (eds.), "Software Product Lines in Action – The Best Industrial Practice in Product Line Engineering", Springer Verlag, Berlin, 2007

[Yoshimura07]
吉村 健太郎, ダルマリンガム ガネサン, ディルク ムーティック: “プロダクトライン導入に向けたレガシーソフトウェアの共通性・可変性分析法,” 情報処理学会論文誌, vol. 48, no. 8, pp. 2482-2491, 2007.