派生開発におけるプロセス構築

～ XDDP からアーキテクチャ再構築へ ～

Embedded Technology 2012
スペシャルセッション資料
（第３セッション）

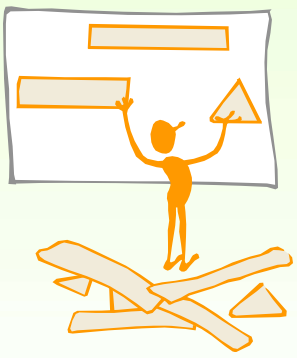
（株）デンソー技研センター
古畑　慶次
kkobata@ndtec.denso.co.jp

Agenda

1．混乱する派生開発

2．XDDPの詳細 －派生開発アプローチ－

3．アーキテクチャ再構築へ

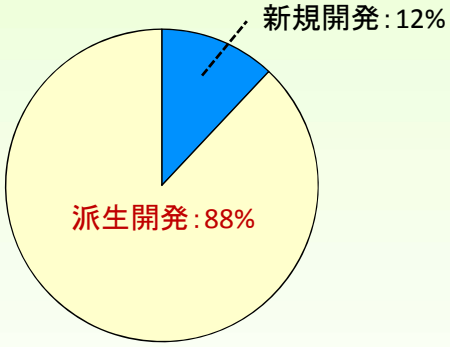


1．混乱する派生開発

1．1 ソフトウェア開発の現状

- ソフトウェア開発の大半は 「派生開発」

開発 ┬ 新規開発
　　 └ 派生開発

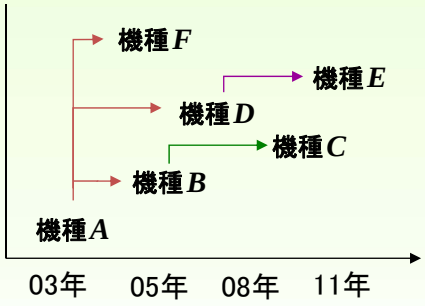


＜ 2009年度 開発行数 ＞

派生開発とは？

既存製品への機能追加や
機能の変更・削除による製品開発

例）初期の携帯電話 → 今日のケータイ端末

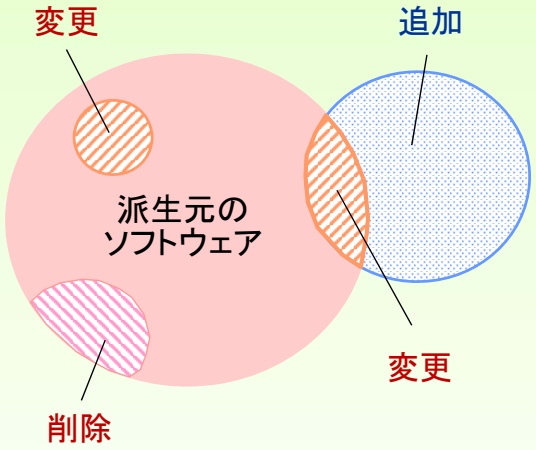


＜ 製品展開 ＞

出典： 2009年組込みソフトウェア産業実態調査（IPA）

1.2 派生開発の現実

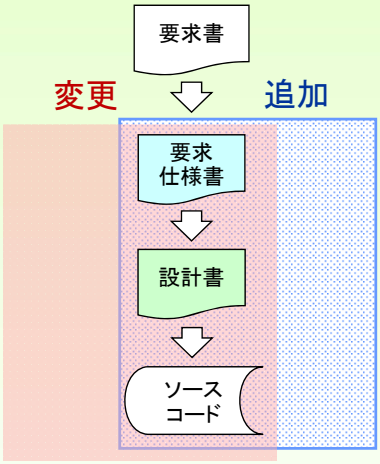
- 派生開発での開発項目



「追加」と「変更」の開発が混在

※ 削除は変更に含む

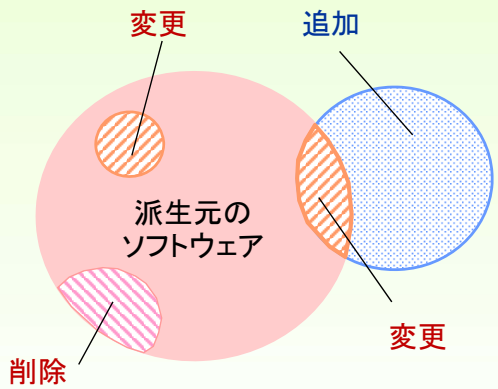
- 開発プロセス



新規開発のプロセスで対応

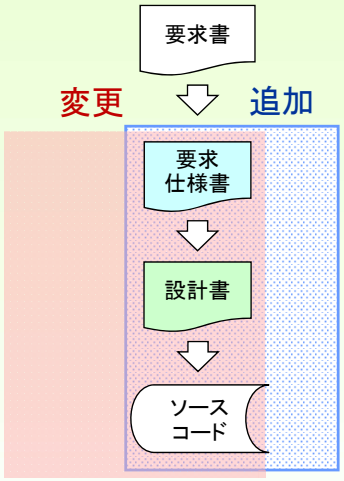
1.3 派生開発の難しさ（1）

- 既存のソフトウェアの変更
  - 関係する機能の特定　：仕様上、影響を受ける機能の抽出
  - ソースコード変更の影響　：ソースコード修正による影響範囲の特定
- 技術者の問題
  - 技術力：ソースコードの読解力
  - 経験　：ドメイン知識（機能理解）
- ソースコード
  - 劣化した派生元のソースコード
    - 保守性を無視した開発



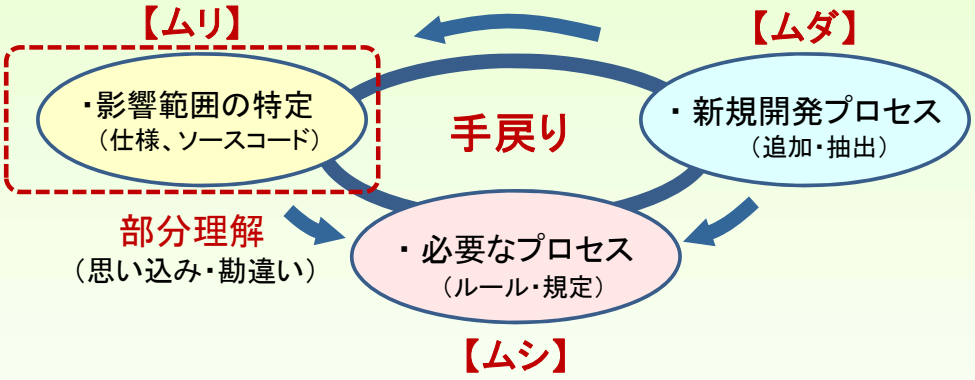
1.4 派生開発の難しさ（2）

- プロセス
  - 新規開発のプロセスは、要求の異なる「変更」と「追加」の両方に対応できない
    - 変更 ： 既存のソフトウェアの変更
    - 追加 ： 新しい機能の開発
  - 差分情報が整理されていない　[変更]
    - 変更点の追加と抽出を繰り返す
  - 見つけ次第ソースコードを直す［変更］
- 環境
  - 設計資料（ドキュメント）の不備
    - 理解の助けにならない



1.5 混乱する派生開発

- 混乱のメカニズム ： ムリ・ムダ・ムシ → 手戻り
  - 品質を確保する技術、ドメイン知識を持たない状態で納期やコストの削減が求められている　→　技術者の疲弊



開発プロセス【ムダ】と影響範囲の特定【ムリ】が問題

1.6 派生開発：問題解決へ向けて

- 派生開発の混乱要因
  - 「開発プロセス」と「影響範囲の特定」
- 「現状」と「改善ポイント」

	現 状	改善ポイント
開発プロセス	・**新規開発**のプロセスを適用 （新規開発崩し）	・**派生開発（変更・追加）**に 対応したプロセス
影響範囲の特定	・**部分理解**での作業 （思い込み、勘違い）	・思い込み、勘違いを 成果物とレビューでカバー

XDDPは派生開発の問題に対する有効なソリューション

2. XDDP の詳細 –派生開発アプローチ–

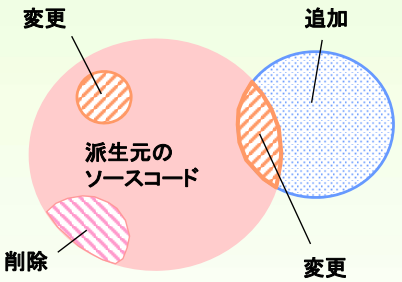
2.1 XDDPとは何か？
2.2 従来手法との比較
2.3 XDDPのプロセス –変更と追加–
2.4 追加のプロセス
2.5 変更のプロセス
2.6 ３点セット –変更プロセス–
2.7 変更要求仕様書 –USDM–
2.8 TM –トレーサビリティ・マトリックス–
2.9 XDDPによる問題解決
2.10 XDDPの効果

2.1 XDDP とは何か？

- XDDP ： ***eXtreme Derivative Development Process***
  - 派生開発の要求に合った開発プロセス
  - 清水吉男氏（システムクリエイツ）が提案
- 合理的な開発プロセス
  - 追加 と 変更 にマッチしたプロセス
    - ２つの独立したプロセス
  - 部分理解を成果物とレビューでカバー
    - 差分情報に基づいた開発
    - 「思い込み」と「勘違い」を排除
    - ムダの徹底排除 ： ***Just in Time （TPS）***

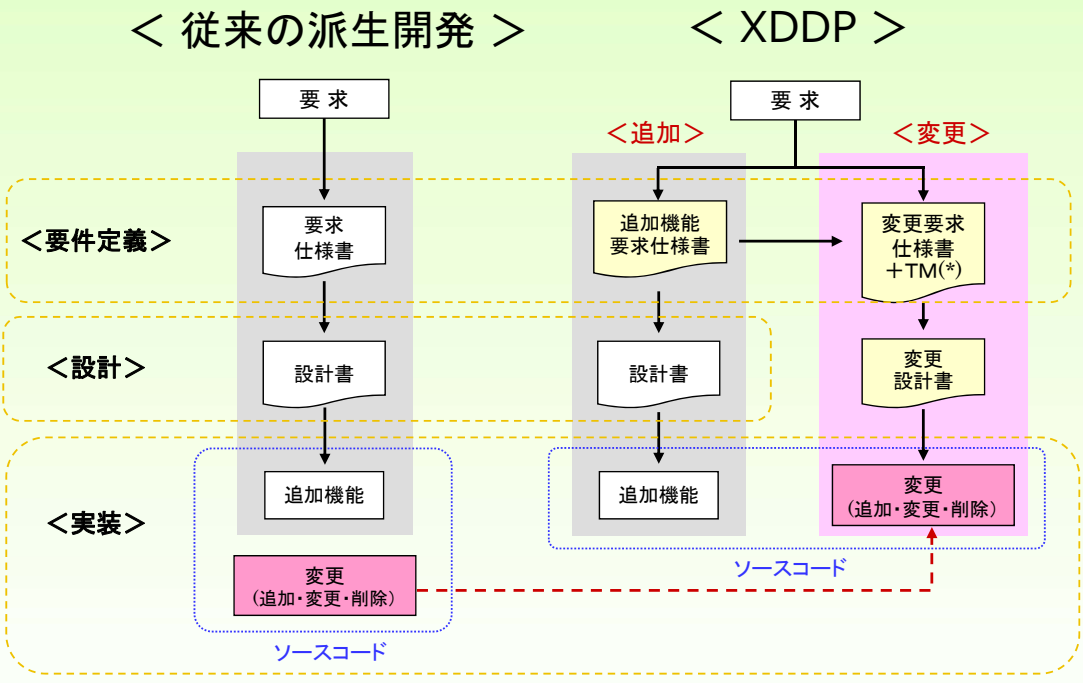
「派生開発」を成功させる
プロセス改善の技術と極意

技術評論社（2007）
清水吉男氏著



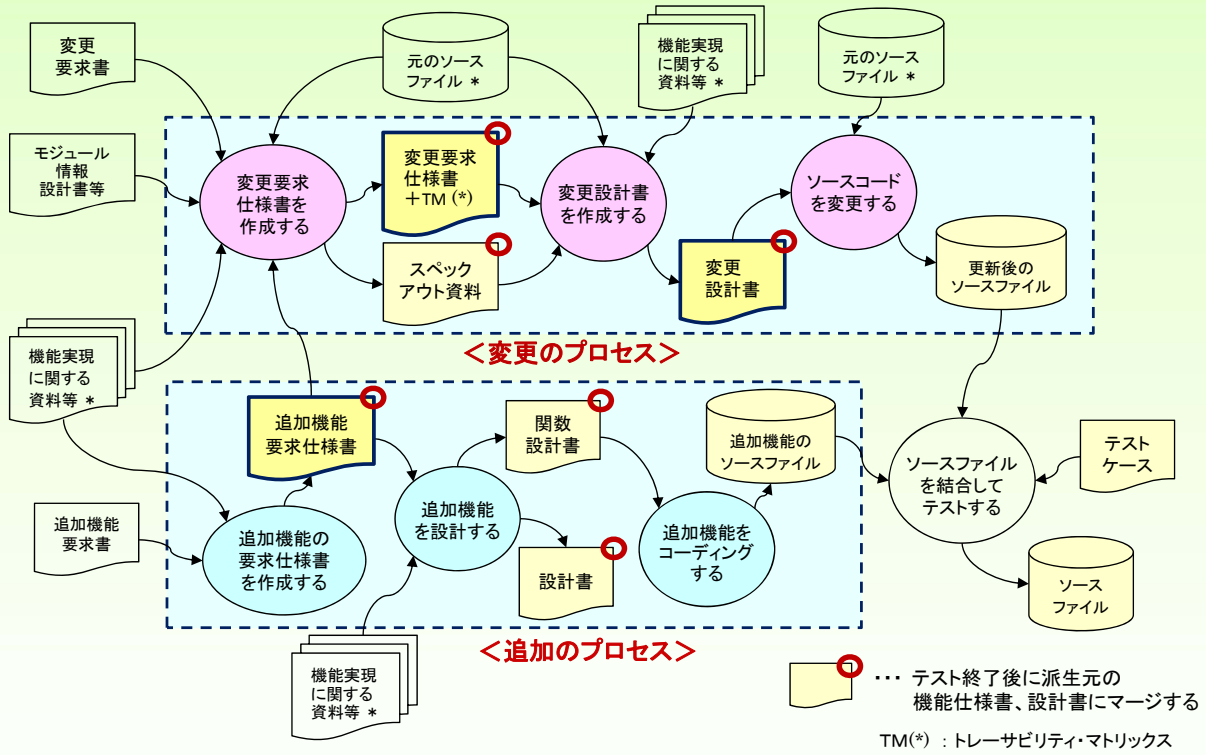
機能追加を受け
入れるための **“変更”**

2.2 従来手法との比較



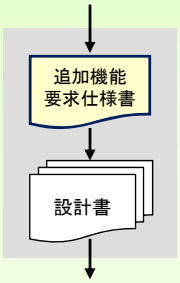
TM(*) ：トレーサビリティ・マトリックス

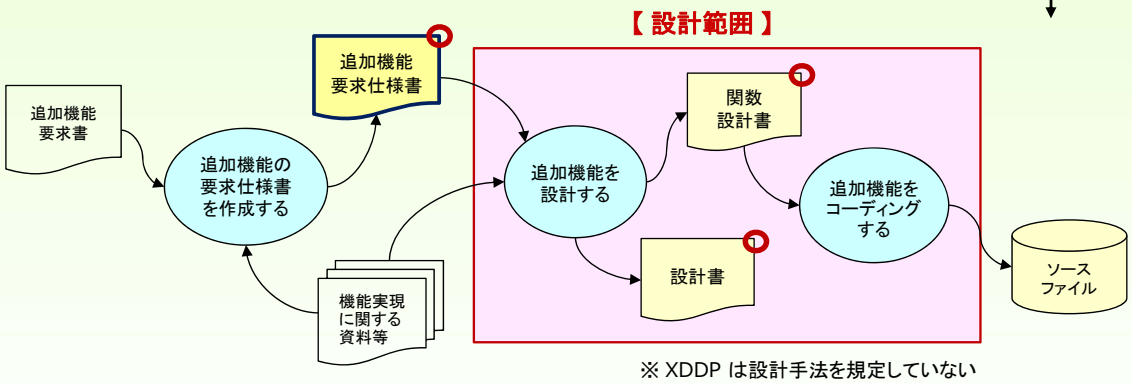
2.3 XDDP のプロセス –変更と追加–



2.4 追加のプロセス

- 追加プロセスの特徴
  - 一般の新規開発のプロセスに準拠したプロセス
  - 追加機能要求仕様書は、新規開発の要求仕様書と基本的に同じ

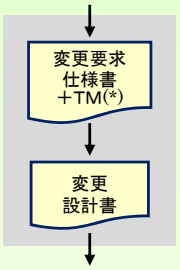


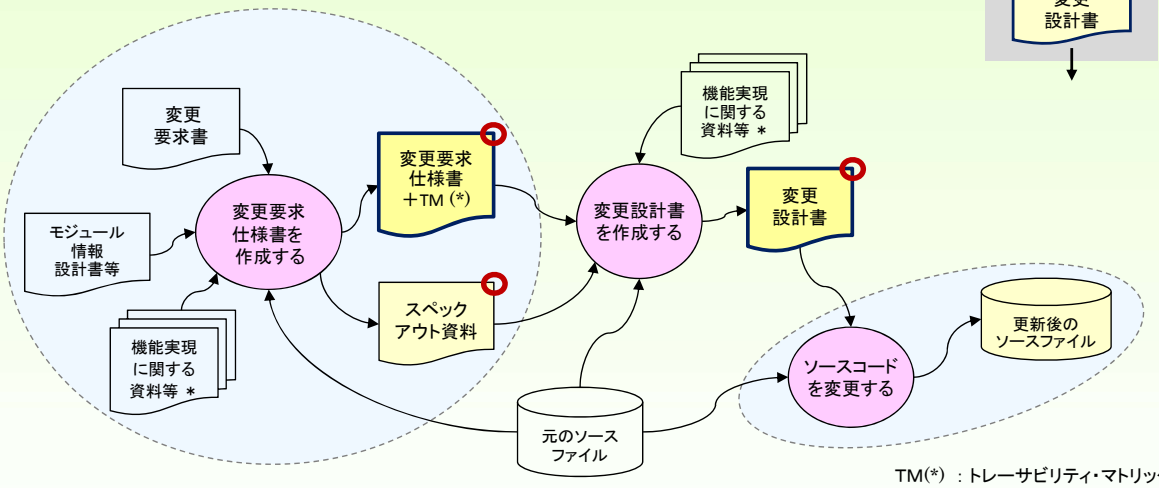


※ XDDP は設計手法を規定していない

2.5 変更のプロセス

- 変更プロセスの特徴
  - 派生元のソースコードの変更点に着目したプロセス
    - 変更箇所、変更方法を 「変更要求仕様書」「TM」「変更設計書」（3点セット）で記述する





TM(*) ：トレーサビリティ・マトリックス

2.6 3点セット –変更プロセス–

- 変更に着目した成果物
  - ソースコードの変更前に3つの成果物で全ての変更内容をそれぞれの視点で抽出しレビューする
  - 担当者の「思い込み」と「勘違い」をレビューでカバー

成果物	カバー範囲	内容	レビュー(**)	
変更要求仕様書	What Why	何をなぜ変更するか どのような振る舞いをするか	○	○
TM (*)	Where	変更仕様がどこにあるか	○	
変更設計書	How	変更仕様をどのように修正するか	○	○

TM (*) ：トレーサビリティ・マトリックス
レビュー(**) ： 変更要求仕様書と *TM* は一緒にレビューをしてもよい

2.7 変更要求仕様書 –USDM–

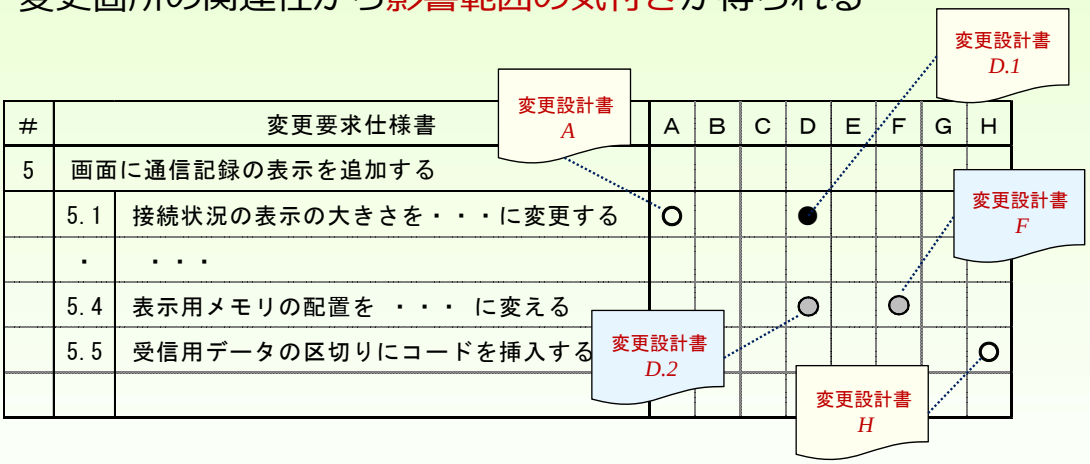
- 要求と仕様を階層構造で表現
  - 要求
    - 仕様
    - 仕様
- 要求の理由を記述
  - 適切な変更を引き出す
- 表現の工夫 （変更の表現）
  - Before ／ After
    - 「 ～ を ○○ に変更する 」
    - 「 ～ を 削除する 」
    - 「 ～ を △△ に追加する 」

要求	Req.1		
	理由		
	説明		
	<Group 1>		
	要求	Req.1-1	
		理由	
		説明	
	仕様	<グループA>	
		Req.1-1-1	
		Req.1-1-2	
		<グループB>	
		Req.1-1-3	
		Req.1-1-4	
	<Group 2>		
	要求	Req.1-2	
		理由	
		説明	
	仕様	<グループC>	
		Req.1-2-1	
		Req.1-2-2	
		<グループD>	
		Req.1-2-3	
		Req.1-2-4	

USDM : Universal Specification Describing Manner

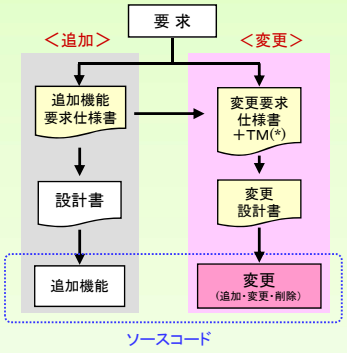
2.8 TM –トレーサビリティ・マトリックス–

- ***TM*** 上で変更仕様と変更設計書を対応させる
  - 変更仕様に該当する箇所を ***TM*** 上に表す
  - 変更箇所の関連性から影響範囲の気付きが得られる



2.9 XDDP による問題解決

- 混乱要因の解消
  - 開発プロセス
    - 変更／追加 の独立したプロセス
  - 影響範囲の特定
    - ① 変更要求仕様書
    - ② TM（トレーサビリティ・マトリクス）
    - ③ 変更設計書



① 変更要求仕様書

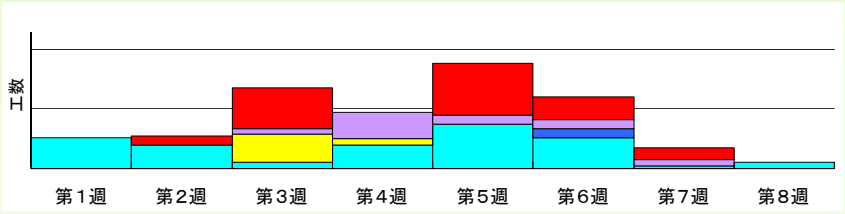
② TM（トレーサビリティ・マトリクス）

#	変更要求・仕様	A	B	C	D	E	F	G	H
5	画面に通信記録の表示を追加する		●						
	5.1 接続状況の表示の大きさを・・に変更する							●	
	・・・					●			
	5.4 表示用メモリの配置を・・に変える								
	5.5 受信時データの区切りにコードを挿入する								

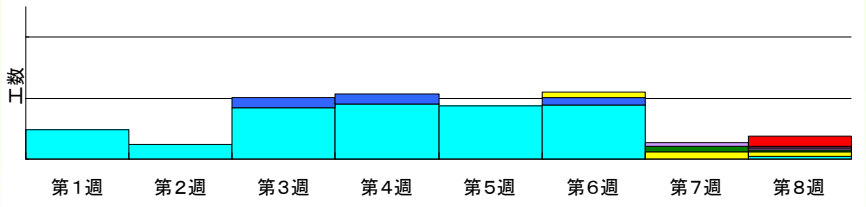
③ 変更設計書

2.10 XDDP の効果

- XDDP 導入前（Before）



- XDDP 導入後（After）



（注）変更規模で正規化済み

出典：加藤由之他：XDDPによるソフト派生開発のQCD向上活動，ソフトウェア品質シンポジウム 2008

3．アーキテクチャ再構築へ

3．1 派生開発では生き残れない

3．2 XDDP から新規開発へ

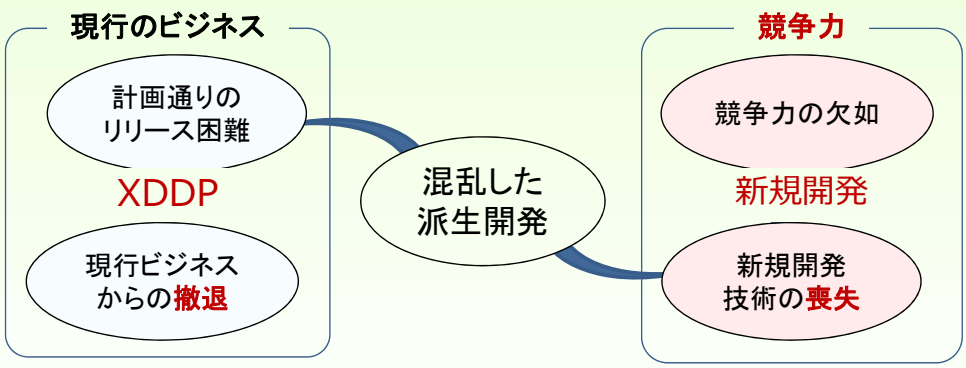
3．3 アーキテクチャ再構築（１）

3．4 アーキテクチャ再構築（２）

3．5 新規開発へのシナリオ

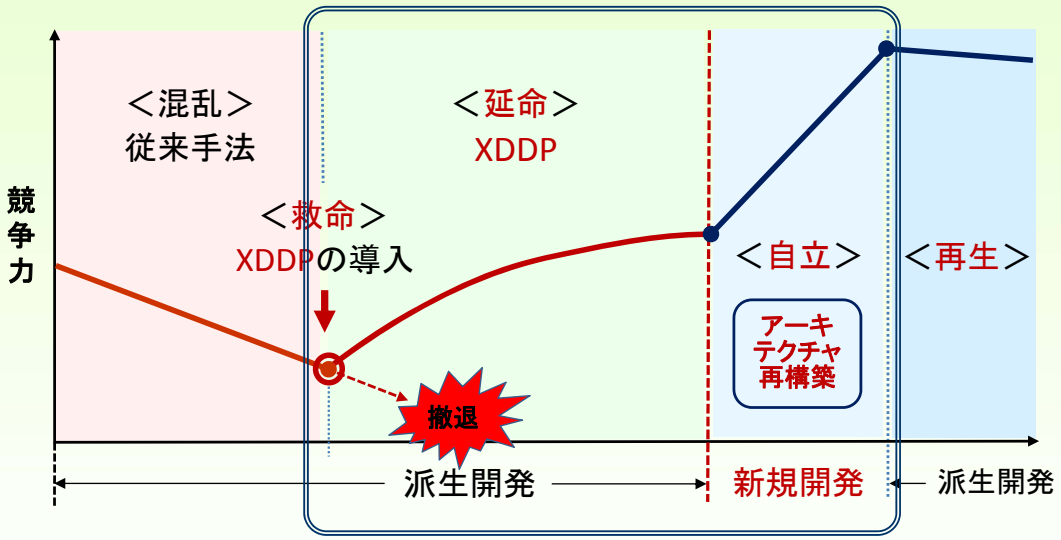
3．1 派生開発では生き残れない

- 競争力の獲得が必須
  - XDDP で現行のビジネスは維持できる
    - 計画通りのリリース、品質確保、生産性向上 ・・・
  - 「生き残る」ための準備はできているか？
    - 現行のソースコードでは限界



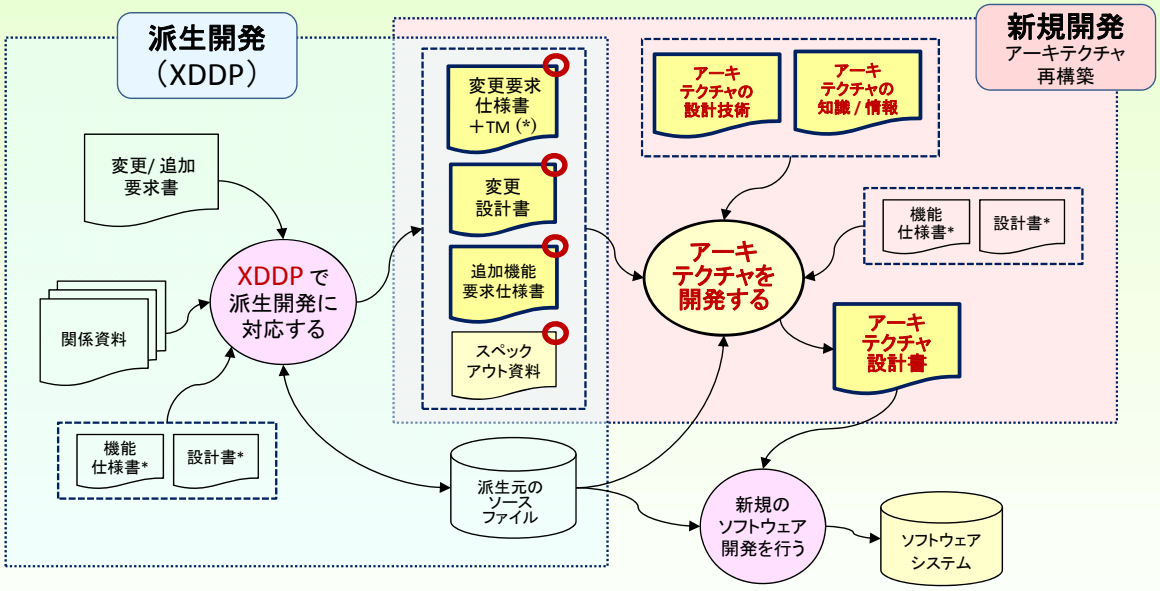
3.2 XDDP から 新規開発へ

- 競争力を手に入れる
  - XDDP（救命・延命）から 新規開発（自立・再生）へ



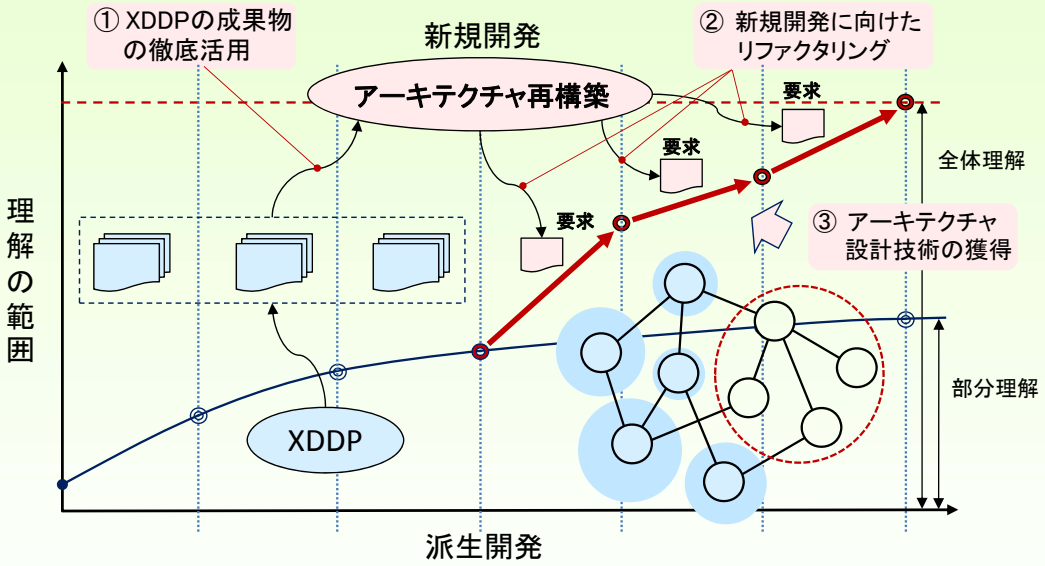
3.3 アーキテクチャ再構築（1）

- XDDP の成果物を新規開発に活用する
  - アーキテクチャの設計技術、知識、情報も重要



3.4 アーキテクチャ再構築（2）

- 派生開発の中で段階的な新規開発を進める
  - 通常の派生開発から新規開発を指向した開発に切り替える



3.5 新規開発へのシナリオ

① XDDP の成果物の活用
- 変更要求仕様書 / 追加機能要求仕様書
- 変更設計書、スペックアウト資料
- TM：再利用資産 (共通性) の検討

② 新規開発に向けたリファクタリング
- リファクタリング → 変更要求

③ アーキテクチャ設計技術の獲得
- XDDP で工数を確保
  → アーキテクチャを検討
- 派生開発 → 新規開発ベカラズ集