派生開発の現状と問題点
その対応の重要性について

Embedded Technology 2012
スペシャルセッション資料
（第１セッション）

派生開発推進協議会
代表　清水 吉男
（㈱システムクリエイツ　代表取締役）

派生開発って？

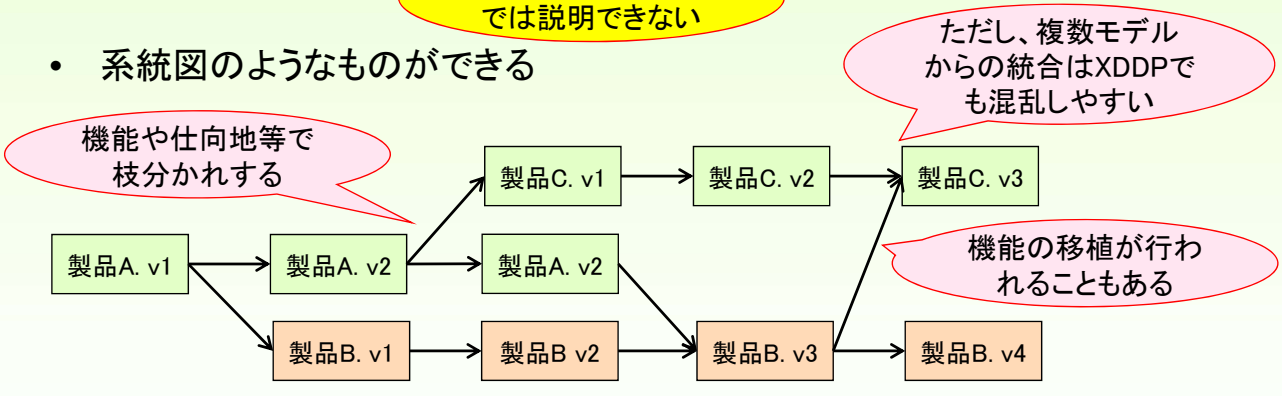
- 派生開発の特徴
  - 機能の追加
  - 仕様の変更

機能が変化していく

初期の携帯電話 → 今日のケータイ端末

この過程は「保守」では説明できない

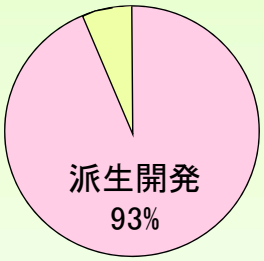
- 系統図のようなものができる



大部分は派生開発

- 多くの組織では、「派生開発」が過半数に達していると思われる

独立行政法人情報処理推進機構　ソフトウェア・エンジニアリングセンター(2010):「ソフトウェア開発データ白書」では、組込み向けで42%，エンタプライズ向けで35%と報告されている。



ある企業(部門)での年間実績

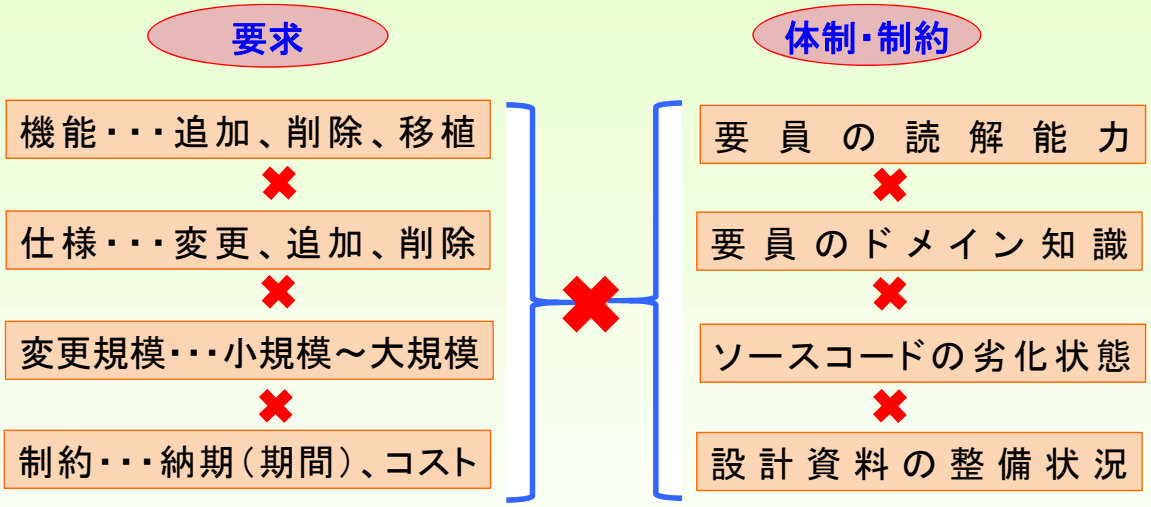
- 組織の標準プロセスも派生開発にマッチしていない

いろいろな問題が発生する

それにも関わらず、
派生開発にふさわしい開発方法は発表されてこなかった

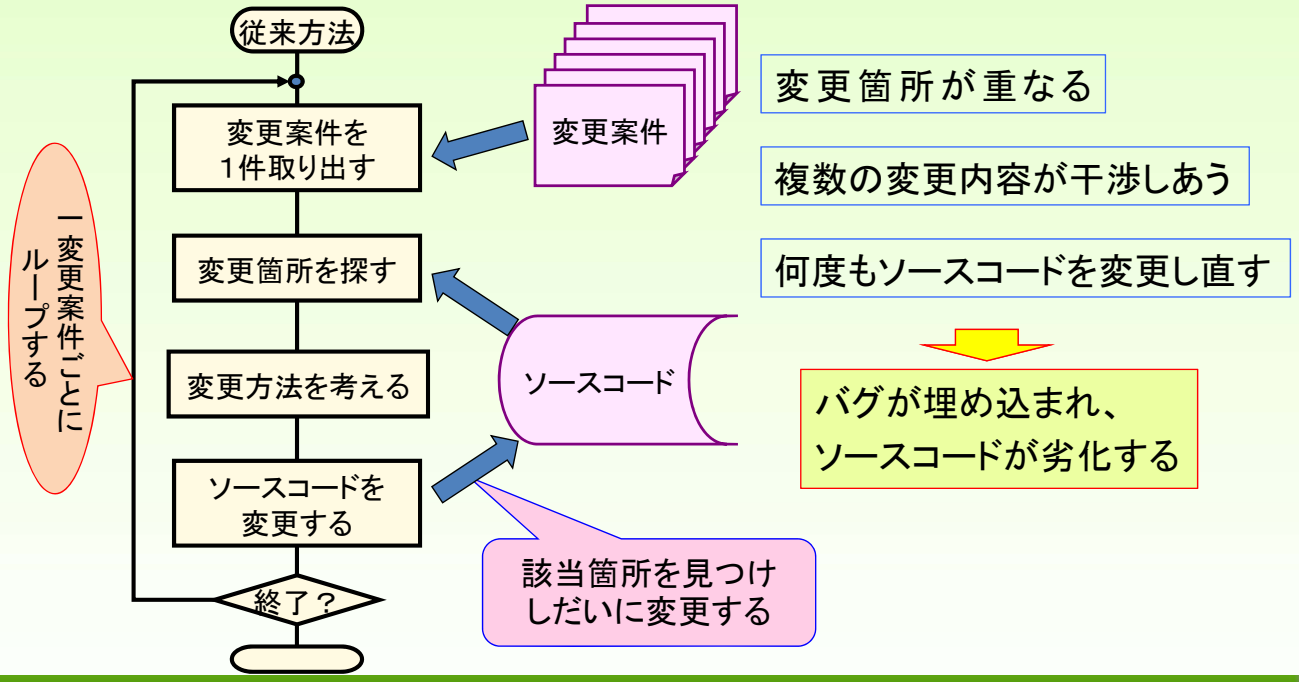
派生開発の要求は多様

- 要求が多様で、それを実現する体制も十分ではない



見つけ次第にソースコードまで変更される

- 公式文書が残されていない多くの現場でのパターン



新規開発と派生開発のバグの違いの意味

- 派生開発と新規開発とでは、発生する不具合の様子が異なる
  - レビューの視点も変えるべき

種類		バグのパターン	対応
新規開発		• 要求仕様で求められていることに対する不適合	• 要求仕様の精度向上 • 要求仕様との適合性レビュー • 仕様実現技術の習得
派生開発	機能追加		
	変更	• 依頼された変更に対する間違い	• 変更要求仕様の表現の工夫
		• ベースのソースコード（旧仕様）に対する認識の不足	• ベースを理解する技術の確保 • 関係箇所／影響箇所の発見方法の工夫

「変更」のバグは、新規開発のプロセスでは対応できない

低い実装工程の生産性が示すもの

- 派生開発に於ける一般的な「実装プロセス」の生産性

10行未満／時間

- 簡単な関数レベルの単体テストを含む場合は「4行／時間」程度になる

4～10分に1行しかコードを書いていないことになる

この数字は何を意味しているのか？

- このとき、ソースコードが変更される以外に、その変更が適切であることを判断できる何かが残されているか？

派生開発は「部分理解」の制約を受ける

- 現状・・・
  - 設計書・・・・・・・理解の助けにならない
  - ソースコード・・・保守性無視 ＋ 劣化の進行
  - 担当者・・・・・・・ソースコードの読解技術の不足

「全体」を理解できる状況にはない

- 「部分理解」の制約の中で変更作業が強いられる

担当者の「思い込み」「勘違い」の混入を防ぐためのプロセスが不可欠

いきなりソースコードを変更する方法では、これに対応できない！

納期優先？

- 上司の声
  - 「前回は、テストで何とかバグを潰したが、納期を2ヶ月遅らせた」
  - 「品質はもちろんだが、今回は特に納期厳守で対応して欲しい」

品質をどうやって確保するの？

テストケースを増やすの？

品質を確保する技術を持たない状態で、納期やコストの削減が求められる

焦って必要なプロセスを省いてしまう

派生開発に適した開発プロセスが必要

このままではリリースできない事態も

- このままでは製品をリリースできない事態に見舞われる

派生開発が混乱した状態

→ 予定通りに次版をリリースできない → ビジネスの継続が困難になる

→ 新規開発の技術習得が進まない → 競争力の決定的要因の入手困難

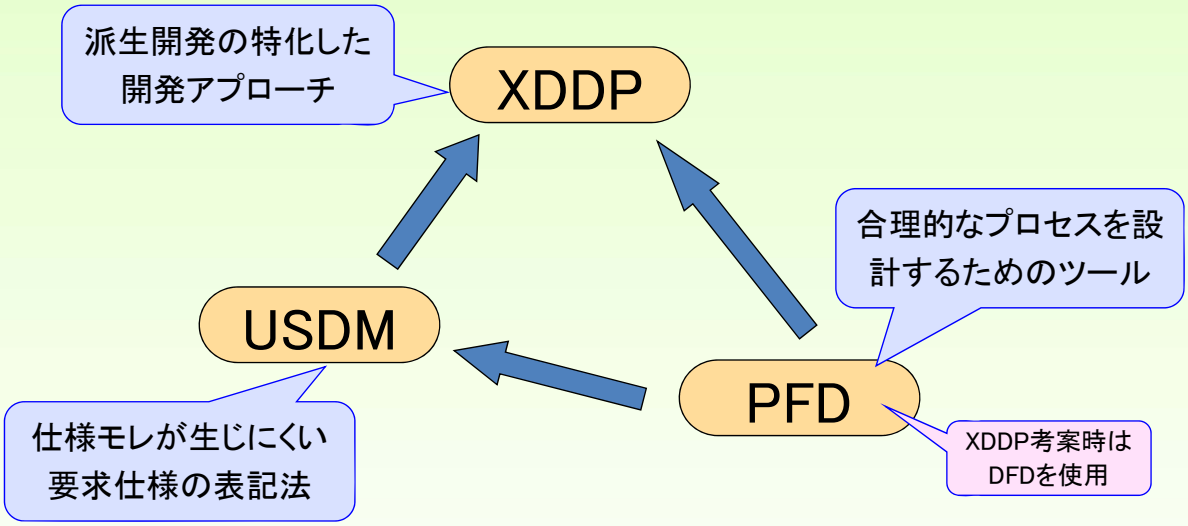
現行の「ソースコード」はいつまで競争力を維持できるのか？

派生開発の混乱状態 → 競争から脱落する

新規開発技術の喪失 → 勝つための競争力を失う

XDDPは、派生開発に特化した開発プロセス

- 「XDDP」は、「USDM」と「PFD」の支援の上で成り立っている



- 「PFD」は、派生開発のプロセス策定の他、
  要求仕様作成のプロセス策定にも活用する

強力な３つの技術

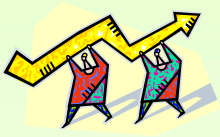
- 「USDM」「PFD」「XDDP」の相互作用で、派生開発の問題の多くを解決する

PFD	• 多様な要求に対して合理的な開発プロセスの設計を支援する • プロセスで品質を確保するための強力なツール
USDM	• 「追加機能要求仕様書」と「変更要求仕様書」の表記法を提供 • 「要求」と「仕様」の階層表現によって仕様が漏れにくい状態を確保 • 要求に「理由」を付加することで認識のズレを矯正
XDDP	• 派生開発の要求にマッチした開発アプローチを提供 • 「機能追加」と「変更」を異なるプロセスで対応 • 「変更要求仕様書」ですべての変更を扱う • 「変更３点セット」の成果物によって「部分理解」による思い込み等に気付く仕掛け • 特に、変更を「before / after」で表現することで影響箇所に気付きやすい

ソフトウェア開発の生産性は大幅に改善できる

- ソフトウエア開発では生産性を追求してこなかったことで、改善の余地は手つかずで残っている

原因・・・H／Wの製造プロセスに囚われている？



- 「XDDP」によって、派生開発の生産性は飛躍的に向上できる

生産性を3倍
バグの発生率を1／10
→ も難しくない

過去の「文化」や「慣習」と決別する
改めてプロセス改善に取り組む
→ 絶好の機会

XDDPの効果

この数字は、現状のプロセスが余りにもずさんであることを示している

- 大手A社から開発途中で3名の不足の申し出
  - 申し出を拒否して別の外注先から「1名」調達（派生開発初経験）
  - 3週間を準備に充てる
  - 2.5ヶ月で終了、変更行数7000行を50時間で実装、責任バグ「0件」
  - この時点で、A社の分担は未完了
- 平均30%の工数減（対当初確保していた工数）
  - 直接のコンサル事例・・・約140例

方法を適切に習得した上での実施

- ざっくり計算

5000人 X 12ヶ月 X 20% = 12000人月

ソフト開発への年間従事者

20%減による年間削減効果

50% → 1人月のコスト X 100万円 = 60億円

50% → SPLEやアーキテクチャなどの勝つための研究に投資

ソフトウェア開発のリショアリング（国内回帰）

- ソフトウェアのオフショア開発が行き詰まる可能性
  - – オフショア先の労働コストの上昇や円安への転換など

それでも、ソフトウェアの開発（派生開発）を海外に発注しつづけるのか？

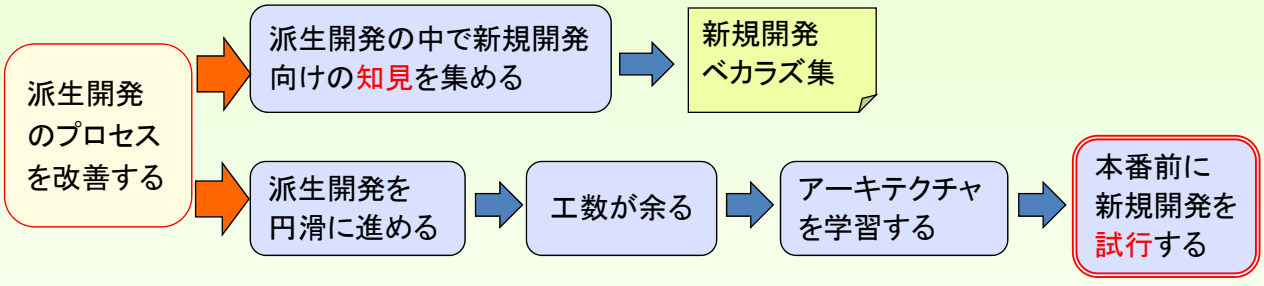
- 国内回帰の際の障壁
  - 必要なソフト技術者をすぐに確保できるか？ （米国との差）
    - 自社の技術者は？
    - 国内の外注先は使えるのか？

ソフトがなければ、
ハードはタダの箱！

- 「XDDP」でQCDを改善することで国内回帰への準備を
  - 国内開発の生産性を30%上げれば、オフショア開発の必要はない
  - 派生開発の領域を国内で対応することで、国内回帰を先取りする

日本には「遷宮」の文化がある

- 多くの神社が「遷宮」の仕組みを持っている
  - ❖日常の修理（保守／派生開発）は適時実施（宮大工の確保）
  - ❖一定期間で建て替えることで「新築の技術」を維持・革新する



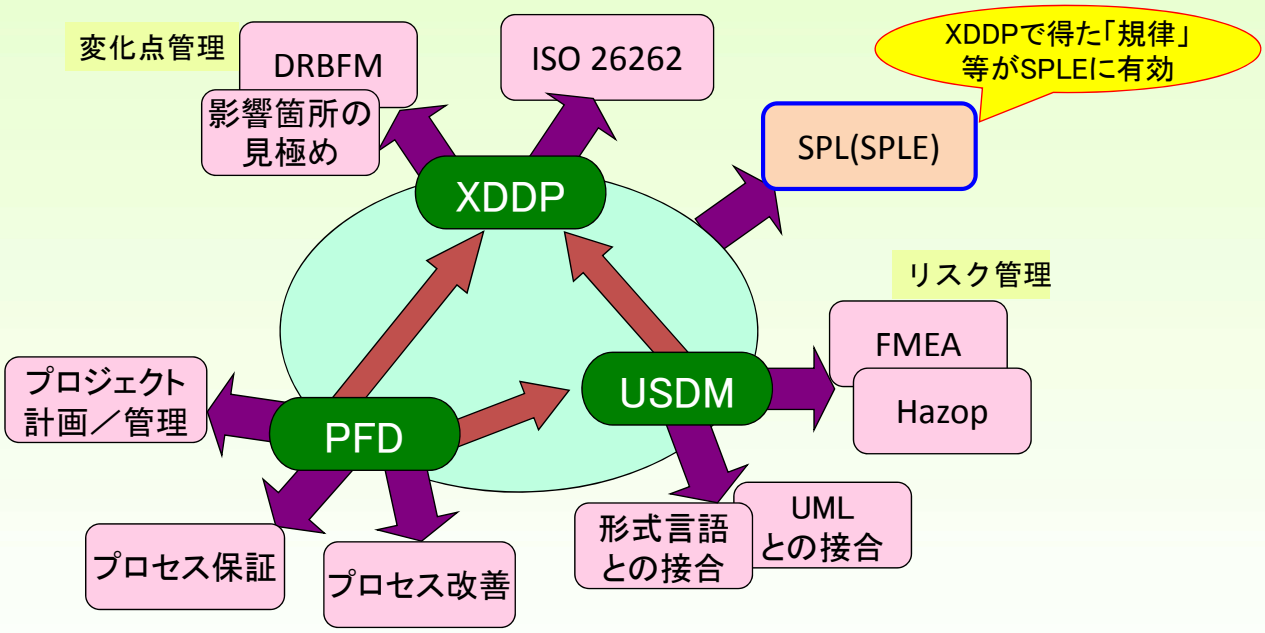
派生開発の技術

新規開発の技術

一体で手に入れておかないと
競争に勝ち続けられない

XDDPトライアングルの展開

- 「XDDP」の3つの技術は、いろいろな取り組みに展開する



それでは、このあと個別のセッションをお聞きください