5

組込みシステムにおけるソフトウェアプロダクトラインの導入

吉村 健太郎 (株)日立製作所 日立研究所／大阪大学大学院情報科学研究科
菊野 亨 大阪大学大学院情報科学研究科

多機能化，多品種化を続ける組込みシステムにおいて，ソフトウェア開発の重要性は年々高くなっており，多くの開発技術が提案されている．SPL は，特に多品種製品向けに技術要素を体系化した開発方法論である．

本稿では実験事例に基づいて，組込みシステム開発における SPL の位置付けとその導入戦略について解説する．

なぜ SPL が必要なのか

近年，多品種製品向けのソフトウェア開発・再利用技術としてソフトウェアプロダクトライン（SPL）が体系化され，組込みシステムへの適用が進んでいる[1)〜3)]．なぜ，組込みシステム開発において SPL が必要なのだろうか．そして，今までの再利用技術と何が違うのだろうか．

既存ソフトウェアは市場の成長，競合との競争に伴って機能追加を繰り返し，顧客の要求に応えてきた．開発済みの資産はウォーターフォールや差分開発等，SPL 以外のフレームワークを用いて開発に成功してきている．なぜ今，開発方法論を変更しなくてはいけないのか．

また，多くの開発組織では，すでにソフトウェアの再利用が行われている．新製品開発にあたって，ゼロからソフトウェアを開発することはほとんどなく，類似の既存製品を「再利用」して差分のみを開発している．ソフトウェアの部品化も決して新しいアイディアではない．

これらは，SPL に初めて接する組込みシステム開発者の多くが抱く疑問である．

自動車機器，医療用機器等の適用事例[4)]によれば，SPL を導入する大きな理由は「市場の変化」である．製品の新規立ち上げ時に SPL を適用した例はほとんどない．SPL への移行は，市場が成熟したときに行われている．製品がある一定の成果を収めた後の市場競争の変化，すなわち多品種化，低コスト化，短納期化といった要求に対応すべく，SPL を導入している．

本稿では以下，組込みシステムにおける SPL 適用事例の背景，そして SPL 導入技術について紹介する．

市場の変化と SPL

◉製品ライフサイクルとソフトウェア開発

製品ライフサイクルにおけるプロセス技術の位置付けから，SPL 導入時期について考察する．**図 -1** 上段に，Utterback ら[5)]による製品ライフサイクルモデルを示す．製品ライフサイクルモデルは技術進化のパターンを示したものであり，技術経営論においてもよく知られたモデルである．

製品ライフサイクルの各段階において製品技術とプロセス技術とを比較すると，それぞれの重要性は大きく変化する．組込みソフトウェアにおける製品技術とは，たとえば電子マネー技術や自動車用ハイブリッドシステムのような，新機能を実現する技術である．プロセス技術とは，開発効率化や信頼性向上を目的とした技術である．SPL はプロセス技術の 1 つであり，特に成熟期において大きな効果を発揮する．

製品ライフサイクルの初期段階は「萌芽期」と呼ばれる．萌芽期では製品技術が流動的であり，多くの企業においてさまざまな技術革新が起こる．製品革新の競争が最も激しい段階である．

次の段階が「成長期」である．成長期には市場および顧客層が拡大していく．それと同時に，さまざまな技術的アプローチの製品デザインの中から，1 つの製品デザインが生き残り，その市場にとっての中心的なデザインとして認められる．これは主要デザインと呼ばれ，市場リーダによるデファクト・スタンダードや，業界団体による標準化によって決定される．主要デザインは，製品を構成する主要な要素技術と，それらを組み合わせる構造や設計思想である製品アーキテクチャを定義する．主要デザインの決定後，製品競争力を左右するイノベーショ

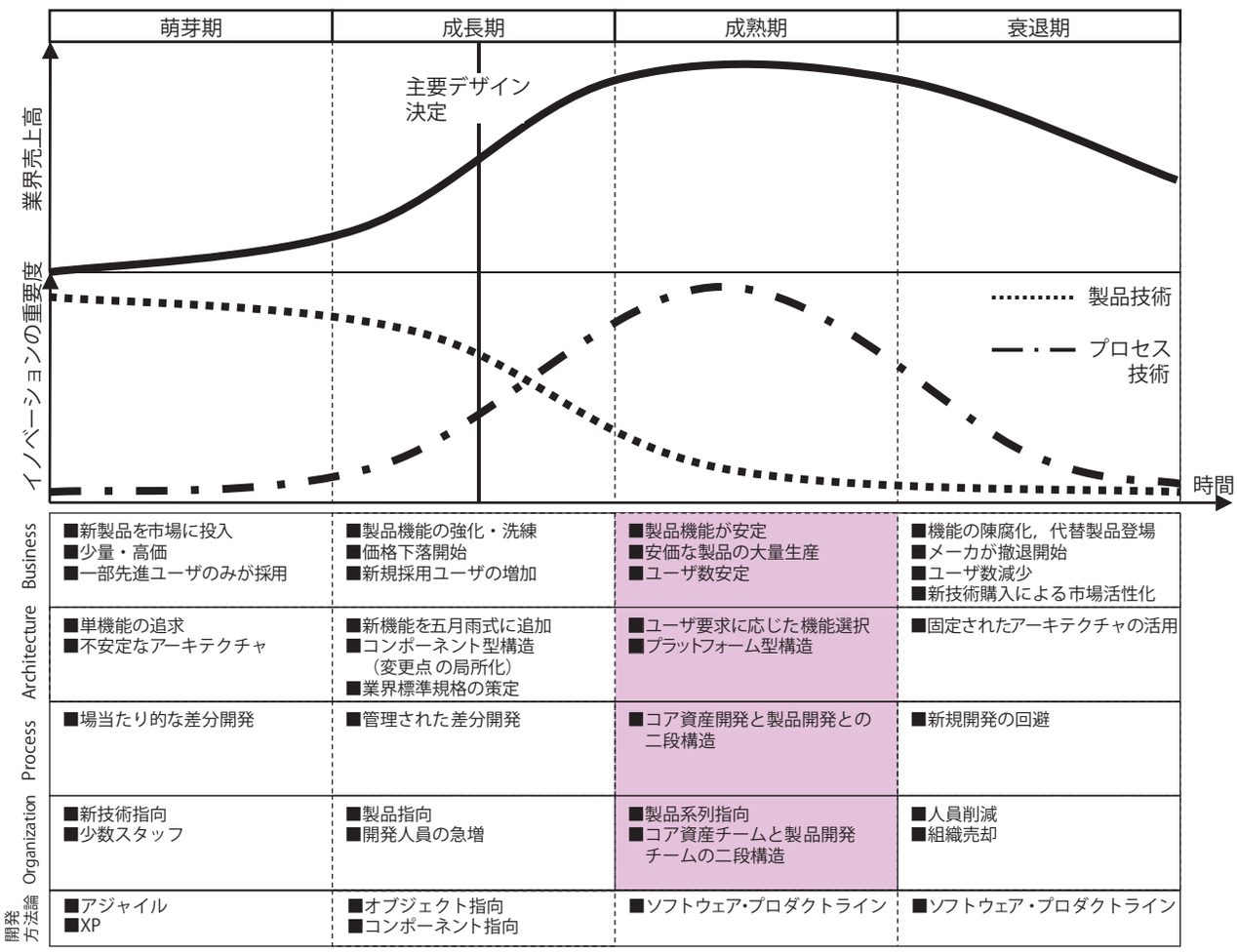

図 -1　製品ライフサイクルとソフトウェア開発

ンは，製品の技術的革新から，開発・生産プロセスの改善へと変化していく．

製品ライフサイクルの第 3 段階は「成熟期」である．技術が発展し，製品機能として顧客ニーズを満足させることができる段階になると，製品技術の持つ競争力が低下する．その結果，市場における競争は，製品技術ではなく，開発・生産プロセスにおけるスピードやコスト，品質が焦点となる．そのため，これらにとって重要な役割を持つプロセス革新が重要な位置を占める．

ライフサイクルの最終段階は，「衰退期」である．この段階では，製品技術の改善，プロセス技術の改善ともに限界を迎え，市場から撤退する企業が増加する．また，代替製品の登場によって市場そのものが減少をはじめる．

◉SPL の導入時期

製品が成熟期に入ると，市場における競争力は開発・製造におけるスピードやコスト，品質が重要となる．SPL はこれらの課題解決を目的とした開発方法論であり，製品ライフサイクルが成熟期に差し掛かる時期での導入が最も費用対効果が高いと考えられる．

萌芽期においては製品の機能が安定していない上，市場に投入する製品のバリエーションはほとんどない．そのため，共通性・可変性分析に基づく SPL 導入は困難である．さらに，そもそも萌芽期にある製品が市場に受け入れられるかどうかは不確定であり，この段階では初期コストの大きい SPL を導入する動機は小さい．

成長期では，市場が大きく拡大する．この段階では，製品機能の強化・洗練による前期採用者（Early Adaptor）の取り込みが重要となる．そのため，製品の主要機能を支援するような新機能の追加が容易に行えるような，差分開発に適した開発スタイルが本段階には適していると言える．

成熟期では市場規模および製品機能が飽和し，既存の市場において顧客満足度を高めることが製品の競争力となる．そこで，基本機種から高機能機種までの幅広い製品ラインナップ，製品開発スピードのアップ，開発コストの低減が必要となる．SPL は製品系列指向の開発技術であり，製品ライフサイクル成熟期の競争力の源泉となる．

なお，衰退期においては，市場からの撤退，新規活用分野の模索，技術革新によるライフサイクルの再起動等の対策が必要となる．

◉BAPO モデル

SPL の大きな特徴は，コンポーネント指向やアーキテクチャといった技術的な視点に加え，企業としてのビジネスの視点を積極的に取り入れた点にある．Linden ら[4] は BAPO（Business, Architecture, Process and Organization）と呼ばれる 4 つの概念に基づいて，SPL のフレームワークを説明している．この 4 つの概念は相互に影響しており，どれか 1 つが欠けても効率的な再利用は成功しない．

図 -1 下段に，製品ライフサイクルと BAPO との対応を示す．以下，成熟期における BAPO の各要素を概説する．

- **ビジネス（Business）**
  製品系列を開発する際には，まず始めに事業として利益を挙げ得る製品系列の範囲（scope）を決定する必要がある．製品系列の範囲が広すぎれば，コア資産の開発に多大なコストを要する．逆に製品系列の範囲が狭すぎれば，市場の要求に応じた製品開発が難しくなる．この範囲の決定には，当該製品分野における将来の市場動向分析と製品開発計画という，ビジネスの視点での検討が必要不可欠である．
  製品が成熟するとともに，製品機能およびユーザ要求が安定し，精度の高い製品開発計画が可能になる．また，低コスト製品の競争力が強くなるため，安価な製品を，短期間で開発することが重要となる．
- **アーキテクチャ（Architecture）**
  製品開発計画に基づいてプロダクトライン・アーキテクチャを設計することによって，市場領域をカバーするソフトウェアを効率よく開発できるようになる．プロダクトライン・アーキテクチャは，単一製品のアーキテクチャではなく，製品系列全体を対象とする．
  成熟期の製品では，ユーザの要求に対応した機能を選択できるようにする必要がある．そのため，製品系列内での共通性と可変性とを識別し，アーキテクチャに反映する．このアーキテクチャに基づいて，製品系列内で利用するソフトウェア部品を開発・実装する．製品開発では，共通性・可変性に対応する開発・検証済みのソフトウェア部品と，製品固有のソフトウェア部品とを組み合わせて開発するコア資産構造を採用し，信頼性の高い製品を効率よく開発する．
- **開発プロセス（Process）**
  SPL の開発は大規模な組織で行われることが多いため，プロセス定義は必要不可欠である．SPL の活動では大きく分けて，従来からの製品開発（Application Engineering）のプロセスに加えて，製品系列で共通に利用するコア資産開発（Domain Engineering）のプロセスを定義する．
- **組織構造（Organization）**
  さらに，以上の活動を円滑に行うための組織構造の定義も欠かせない．組織構造は個別製品に対応して構成されていることが多いが，製品系列を横断したコア資産開発の役割をどのように割り当てるかが重要なポイントとなる．従来，製品別に開発していた機能をコア資産として共通に開発・利用するため，開発工数を製品固有機能に投入することができ，多品種開発が可能となる．

以上に示すように，SPL は特に成熟期の製品分野において大きな効果を発揮する開発方法論である．ところが，製品分野が成熟期に到達するには，萌芽期，成長期を経由しているケースがほとんどである．そのため，SPL 以外の手法で開発された既存資産を，SPL へと移行することが必要となる．

実験事例

◉実験対象の概要

本稿では以下，組込みシステムの一例として，自動車制御システムでの実験事例を解説する．自動車では，エンジン，自動変速機など多くの組込みシステムが用いられている．これらのシステムに組み込まれるソフトウェアは大規模・複雑化が進んでおり，そのようなソフトウェアを限られた時間でいかに効率的に開発するかが大きな課題となっている．

図 -2 に，自動車制御システムにおけるソフトウェア構造の概要を示す☆1．ソフトウェアの構造として，アプリケーションソフトウェアと基本ソフトウェアの 2 つに分離されている．さらに，アプリケーションソフトウェアはソフトウェア部品とフレームワークとで構成されて

☆1 吉村健太郎，宮崎泰三，横山孝典：オブジェクト指向組み込み制御システムのモデルベース開発法，情報処理学会論文誌，Vol.46, No.6, pp.1436-1446（June 2005）．

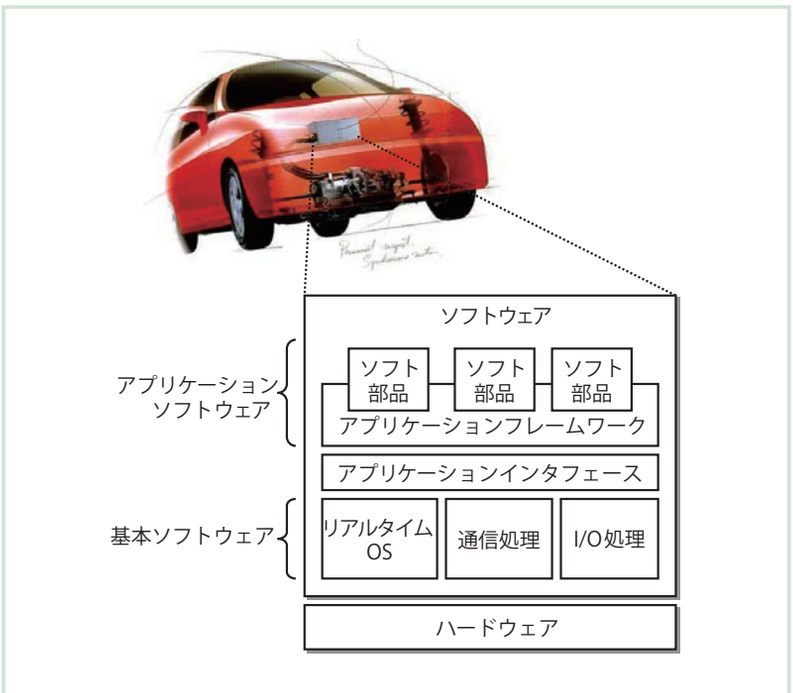

図 -2　自動車向け組込みソフトウェア

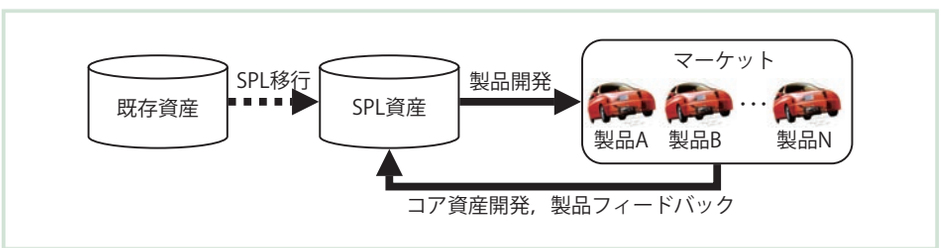

図 -3　SPL の概念図

いる．ソフトウェア部品は定型化された標準的なインタフェース規約を有している．フレームワークは，ソフトウェア部品を部品間の接続関係に基づいてリアルタイム OS 上に配置し，アプリケーション全体を統合している．

◉実験の背景

自動車制御システムでは，まず上級車種向けに先進技術を開発し，その後，中級車種，普及車種へと差分開発によって，車種展開を実施するのが一般的である．

しかし，たとえば電子式エンジン制御システムはほぼすべての車種が搭載する成熟技術となっている．ビジネスの観点から見ると，日本，米国，欧州等で異なる排気規制に対応するための多品種化や，BRICs[☆2] など新興国への市場多様化にも対応しなければならない．技術的にも，業界標準仕様 AUTOSAR（AUTomotive Open System ARchitecture）[☆3] の策定が進んでおり，主要デザインが決定されつつある．上記の観点から，従来の差分開発から SPL による多品種開発への移行が必要不可欠であると考えられる．

図 -3 に，SPL への移行および移行後の製品開発プロセスの概要を示す．自動車制御システム等のセーフティ・クリティカル・システムでは信頼性がきわめて重視されるため，実績のある既存システムに基づいて設計を最適化したいという要求が強い．そのため，既存資産に基づいた SPL 開発への移行を行う．その後に，コア資産を用いた多品種開発を実行する．SPL への移行後は，市場分析に基づくコア資産開発や，個別製品開発に伴う新規機能のフィードバックによってコア資産の進化を継続する．

ところが，既存のソフトウェア開発では，市場の要求に応えるための機能追加に主眼が置かれており，製品間の共通性・可変性については設計されていないことが多い．そのため，SPL 移行段階で可変性分析を実施する必要がある．

特に，SPL の効率的な運用には可変フィーチャ数の削減が必要不可欠である．たとえば，2 通りの選択を持

☆2 Brasil, Russia, India, China の 4 カ国.
☆3 Kinkelin, G. et al. : AUTOSAR on the Road, *Convergence 2008*, SAE Technical Paper 2008-21-0019, SAE International (2008).

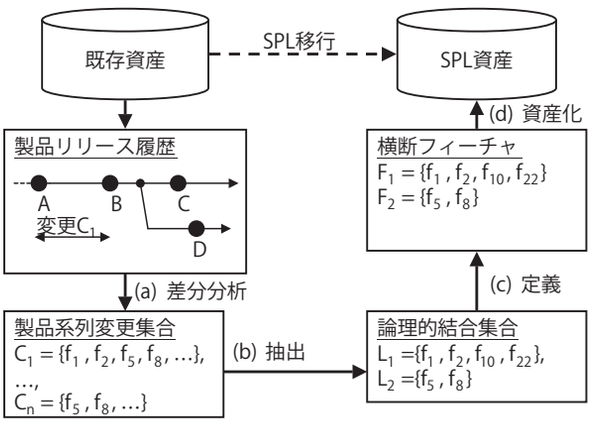

図 -4　既存システムへの SPL 導入

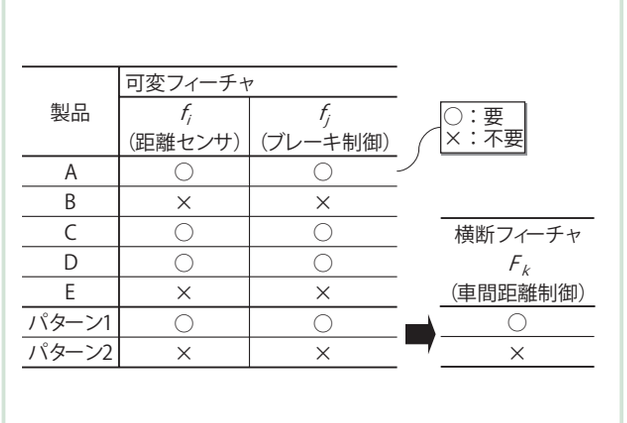

図 -5　横断フィーチャ

つ可変フィーチャを 20 個持つシステムですら，組合せ数は 100 万通りを超えてしまう．自動車制御システムでは，可変フィーチャ数が製品全体で数千になる例も報告されており，製品開発時に選択しなければならない可変フィーチャの組合せ数が膨大になる．このため，派生製品の開発において製品仕様を満足する可変フィーチャを設定するための工数が増大するという課題がある．SPL 型開発を効率的に運用するためには，可変フィーチャ数の削減が重要である．

既存製品における SPL 導入支援を目的として，我々は SPL 導入前の既存製品リリース履歴を用いた分析法を提案し，自動車制御システムを題材とした実験および評価を行っている [6]．次章では，既存資産に基づく可変性分析の概要と実験事例を紹介する．

製品リリース履歴に基づく可変性分析

◉既存システムへの SPL 導入

図 -4 に，我々が提案する既存製品の SPL 移行手法の概要を示す．既存製品から SPL への移行にあたり，まずはじめに (a) 製品リリース履歴から製品間の変更情報を分析する．次に，(b) ソフトウェア部品の論理的結合集合を抽出する．論理的結合集合については後述する．(c) 決定した論理的結合集合に基づいて横断フィーチャを定義し，(d) 横断フィーチャを含むフィーチャモデルをコア資産として用いる．

本手法では，開発履歴の分岐を考慮した上で，製品リリースタイミング間での変更個所を抽出して製品リリース履歴を前処理する．これにより，バリエーション開発が行われている製品分野における分析が可能になる．

◉横断フィーチャ

横断フィーチャとは，複数の詳細な可変フィーチャを横断して影響をおよぼす，抽象度の高いフィーチャである．**図 -5** に車両制御システムの例を示す．製品 A から E は，車種や仕向地が異なる製品としてリリースされている．可変フィーチャとして「距離センサ f_i」，「ブレーキ制御 f_j」が定義されている．ここで，f_i, f_j は独立なフィーチャとして個別に設定可能である．しかし，製品 A から E までの開発履歴を確認すると，f_i, f_j の両方が要，または両方が不要として設定されていることが分かる．横断フィーチャとして「車間距離制御 F_k」を定義することによって，共通に影響を受けていた複数の可変フィーチャ f_i, f_j をまとめられる．そのため，製品開発時の選択肢を削減することが可能となり，派生製品の開発を効率化できるという利点がある．

◉製品リリース履歴

製品リリース履歴の概要を**図 -6** に示す．提案手法では，製品リリース履歴に基づいて，ベース製品から新製品への差分情報を抽出し，製品を構成するソフトウェア部品の変更集合として表現する．各ソフトウェア部品と 1 対 1 に対応する可変フィーチャに基づいて，横断フィーチャを推定する．

製品リリース履歴 i における変更集合を C_i とする．

$$C_i = \{f_j, f_k, \cdots, f_l\} \quad (1)$$

図 -6 の例では，製品 A から製品 B にかけての変更集合 C_1 は下記のようになる．

$$C_1 = \{f_1, f_2, f_3, f_5\}$$

その上で，製品リリース履歴 $C_1, C_2, C_3, \ldots$において同

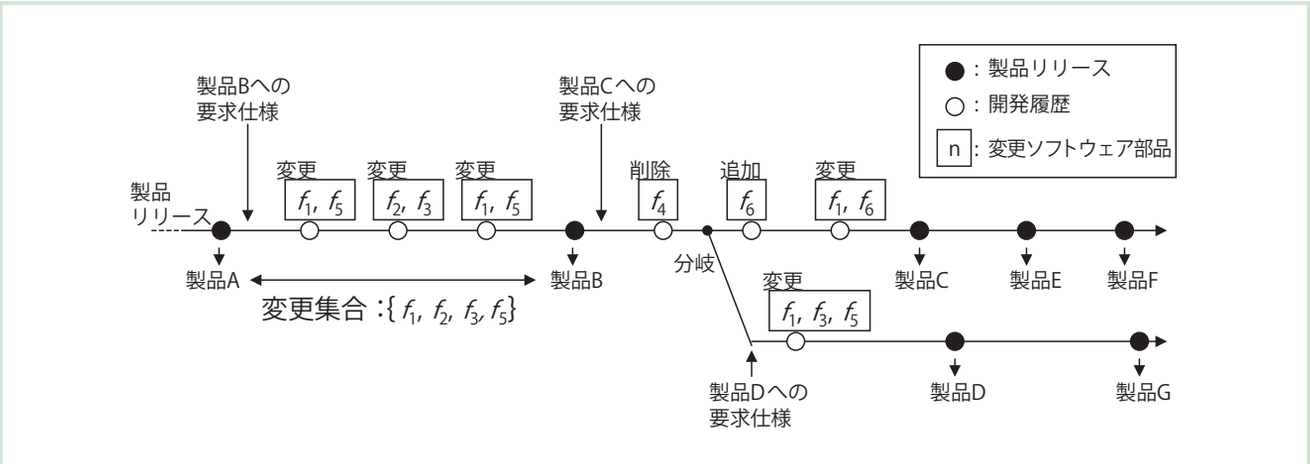

図 -6　製品リリース履歴

分析対象	エンジン制御（一部）
製品数	37
ソフトウェア部品数	63

表 -1　実験対象

横断フィーチャ	可変フィーチャ
F_1	f_{21}, f_{31}
F_2	f_{25}, f_{26}, f_{27}
F_3	f_{28}, f_{29}, f_{30}
F_4（燃料性状）	f_{32}（点火時期補正）, f_{39}（燃料補正）
F_5	f_{50}, f_{51}
F_6	f_{52}, f_{53}

表 -2　横断フィーチャ定義結果

時に変更される頻度が高いソフトウェア部品の集合を，論理的結合集合として抽出する．さらに，論理的結合集合にあるソフトウェア部品に対応する可変フィーチャに基づいて，横断フィーチャの推定を行う．

◉実験

自動車制御システムの製品リリース履歴に対して本手法を用いて横断フィーチャを分析した．

実験対象の概要を**表 -1** に示す．分析対象はエンジン制御ソフトウェアであり，ソフトウェア全体の規模は 50 万 LOC 以上である．実験では，特にエンジン気筒数等の機械的構成や，仕向地による排気規制等の非機能要件の影響を受けやすい，燃焼制御サブシステムの一部を選定して実験対象とした．対象となるサブシステムは 63 個のソフトウェア部品で構成されている．各ソフトウェア部品はたとえば補正係数等の，粒度の細かいフィーチャ f_1 から f_{63} に対応している．製品変更履歴には 37 製品が含まれ，搭載車種，気筒数等のエンジン形式，出荷地域など多くの製品仕様が少しずつ異なっている．

製品リリース履歴に基づいて，横断フィーチャを抽出した結果を**表 -2** に示す．まず，論理的結合集合に含まれるソフトウェア部品の可変フィーチャを調査し，可変フィーチャ間に共通する横断フィーチャを推定した．たとえば，点火時期補正 f_{32} と燃料補正 f_{39} という 2 つのソフトウェア部品に共通するフィーチャとして，燃料性状（レギュラー or ハイオク）という横断フィーチャ F_4 を定義した．さらに，論理的結合集合が生じている製品リリース履歴において，定義した横断フィーチャ（燃料性状）が実際に変更されていたかどうかを調べ，推定の妥当性を確認した．抽出した論理的結合集合に対して上記の仕様分析を実施し，燃料性状やバルブタイミング制御の有無等，6 つの横断フィーチャ $F_1, F_2, F_3, F_4, F_5, F_6$ を定義した．

SPL の実用化に向けて

本章では，SPL 実用化における代表的な研究課題を，特に産業界の視点から紹介する．

◉スコーピング

どの製品群に対してどのようなコア資産を構築するべきか，というスコーピングの問題は，SPL 導入において依然として大きな課題である．

たとえば，ドイツの Robert Bosch 社は「ガソリンエ

ンジン制御システム」という1つの製品分野に対して複数のコア資産を導入した一方で，オランダのPhilips Medical社は複数の医療用製品を横断して「画像処理」という機能を提供するコア資産を導入している[4)]．

スコーピングを適切に実施するためには，フィーチャの共通性・可変性分析に加えて，将来の製品群への要求を予測するための市場分析が必要不可欠である．しかしながら，市場分析とSPLのスコーピングとを関連付けた手法はほとんど提案されておらず，今後の研究が必要である．

また，SPLへの導入を決断するためには，投資対効果を予測するための経済的予測モデルが必要であり，たとえば野中ら[☆4]は開発工程における不確実性を考慮した予測モデルを提案している．今後は製品分野の特性，たとえば市場成長率，機能増加率，競合の存在等を考慮した，カスタマイズが可能で拡張性の高い予測モデルが必要である．

◉モデルベース開発

近年，組込みシステムではDSL（Domain Specific Language）によるモデルベース開発の実用化が進んでいる[☆5]．この流れを受けて，可変性のモデル化を支援するツールも登場している．しかしながら，これらのツール群における可変性の表記方法やファイル形式の統一は行われておらず，プロセス全体を一貫したツールチェインの構築は困難である．ツールチェイン構築のための標準化は，実用上重要な課題である．

◉品質保証

この分野での重要な課題は，品質保証工程を効率化するための統合テストケースの再利用である．たとえば，岸ら[☆6]は形式検証を用いたソフトウェア部品の高信頼化，再利用支援手法を提案している．

複数製品へのコア資産再利用を支援するためには，ソフトウェア部品に対する仕様の形式化および形式検証がさらに重要になる．すなわち，可変性と統合テストケースとの対応付けによる再利用可能統合テストケースの開発および，個別製品向けのテストケース自動生成，自動実行環境の構築が求められる．

◉構成管理

共通したコア資産を用いた複数製品の同時並行開発には，適切な構成管理が必要である．また，可変性メカニズムの実装によっては，ファイルのバージョンだけでなくバリエーションも管理する必要がある．しかしながら，CVS，Subversion等，現在広く用いられている構成管理ツールにはバリエーション管理の機能がない．

また，実現した製品群とコア資産との対応関係の管理に対しても，現在提案されている構成管理の機能には改善の余地が大きい．コア資産開発および製品開発の両プロセスにおいて可変性と開発成果物を管理するために，切れ目ない構成管理を確立することは，今後の主要研究課題である．

◉組織構造

SPLにおける組織の構造化および意思決定プロセスについては，十分な知見が蓄積されていない．特に，コア資産開発組織と製品開発組織とのインタフェースについては，多くの議論が残されている．

さらに，実際の運用では製品開発に比重がおかれることが多く，コア資産開発の専門組織を持たないことがある[☆7]．最適な組織構造を決定するためのパラメータおよび運用方法について，組織論の観点からの研究が必要である．

◉コア資産の進化

コア資産の肥大化，陳腐化を避けるためには，実際の製品開発におけるコア資産の利用状況に応じて，コア資産を洗練させていくことも必要である．たとえばLoeschら[☆8]は，可変フィーチャの利用状況を概念束により分析し，フィーチャ数を削減する手法を提案している．

また，どんなに注意深く分析されたコア資産であっても，顧客の要望に応えたり，競合に対抗するために，製品開発において新規の機能開発や変更が行われる．そのため，製品開発において実現された新機能をどのようにコア資産として共通化するかというプロセスも重要な研究課題である．

SPLの普及展開に向けて

組込みシステム向けのソフトウェア開発方法論として，SPLの位置付けを解説し，実験事例を紹介した．組込

☆4 Nonaka, M., Zhu, L., Ali Babar, M. and Staples, M. : Impacts of Architecture and Quality Investment in Software Product Line Development, *11th Software Product Line Conference（SPLC 2007）*, pp.63-73（2007）.
☆5 Matsumoto, Y.: Experience of a Software Factory from Domain Preparation to Product Line Adoption, *Keynote of 11th Software Product Line Conference（SPLC 2007）*（2007）.
☆6 Kishi, T. and Noda, N. : Formal Verification and Software Product Lines, *Communications of the ACM*, Vol.49, No.12, pp.73-77（2006）.
☆7 森　勇仁，榎並崇史：ソフトウェア大規模再利用のための共通フレームワーク構築，*Ricoh Technical Report*, No.34, pp.98-104（2008）.
☆8 Loesch, F. and Ploedereder, E. : Optimization of Variability in Software Product Lines, *11th International Software Product Line Conference（SPLC 2007）*, pp.151-162（2007）.

みシステムは我が国が競争力を持つ産業の多くを支える重要な技術となっているが，近年の機能増加・市場拡大に伴い，ソフトウェアの大規模化・多品種化が大きな課題となっている．このような状況下で，SPL は製品系列を横断したソフトウェアの再利用技術として活用が期待される．

SPL の高度化には，学術界による研究はもちろん，産業界による事例報告および課題提起が必要不可欠である．提案されている手法を現場のソフトウェア開発で試行・評価して，その効果や問題点をコミュニティにフィードバックすることにより，研究開発の有用性を高めることができる．

代表的な SPL コミュニティの 1 つとして，ソフトウェア・プロダクトライン国際会議（Software Product Line Conference）が毎年開催されている．第 13 回目となる今年は，2009 年 8 月 24 日〜28 日に米国・サンフランシスコで開催される．日本の産業界からも多くの方が参加され，活発な議論が行われることを期待する．

参考文献

1) Bayer, J. et al. : A Methodology to Develop Software Product Line, *Proceedings of the Fifth ACM SIGSOFT Symposium on Software Reusability (SSR'99)* (1999).
2) Clements, P. and Northrop, L. M. : *Software Product Lines : Practices and Patterns*, Addison-Wesley (2001). 前田卓雄（訳）：ソフトウェアプロダクトライン，日刊工業新聞社 (2003).
3) Pohl, K., Böckle, G. and van der Linden, F. : *Software Product Line Engineering : Foundations, Principles and Techniques*, Springer (2005). 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).
4) van der Linden, F., Schmid, K. and Rommes, E. : *Software Product Lines in Action : The Best Industrial Practice in Product Line Engineering*, Springer (2007).
5) Utterback, J. M.: *Mastering the Dynamics of Innovation*, Harvard Business School Press (1994). 大津正和，小川　進（訳）：イノベーション・ダイナミクス，有斐閣 (1998).
6) 吉村健太郎，成沢文雄，橋本幸司，菊野　亨：製品リリース履歴における論理的結合関係に基づいた横断フィーチャ分析法，組込みシステムシンポジウム 2008 論文集，pp.51-59 (2008).

（平成 21 年 2 月 9 日受付）

吉村 健太郎（正会員）
kentaro.yoshimura.jr@hitachi.com
1999 年早稲田大学理工学部機械工学科卒業．2001 年同大学院理工学研究科機械工学専攻修士課程修了．同年（株）日立製作所日立研究所入社．組込みソフトウェア開発方法論に関する研究に従事．IEEE，日本機械学会各会員．

菊野　亨（正会員）
kikuno@ist.osaka-u.ac.jp
昭和 50 年大阪大学大学院博士課程修了．工学博士．同年広島大学工学部講師．同大助教授を経て，昭和 62 年大阪大学基礎工学部情報工学科助教授．平成 2 年同大教授．現在，大阪大学大学院情報科学研究科教授．平成 20 年大阪大学留学生センター・センター長．主にフォールトトレラントシステム，ソフトウェア開発プロセスの定量的評価に関する研究に従事．電子情報通信学会，本会各フェロー．ACM，IEEE 各会員．日本信頼性学会会長．