4

ソフトウェアプロダクトライン開発のマネジメント：課題と技法

野中 誠　東洋大学経営学部

本稿では，ソフトウェアプロダクトライン開発におけるマネジメント上の課題として，ソフトウェア構成管理の問題，開発プロセス，投資としてのコア資産開発，および SPL 価値の組織的理解について，その課題および技法について紹介する．

SPL 開発マネジメントにおける課題

当たり前のことだが，ソフトウェアプロダクトライン（Software Product Line ; SPL）開発はソフトウェア開発である．したがって，過去 40 年にわたってソフトウェアエンジニアリングの実践および研究を通じて指摘・蓄積されてきたマネジメント上の課題は，SPL 開発においても，ほぼそのまま引き継がれる．

しかし，SPL 開発のマネジメントを考える際には，次に挙げた非 SPL 開発との特徴的な違いについて，SPL 固有の検討が必要になる．

- **ソフトウェア構成**：コア資産と製品固有部分が，明示的・戦略的に区別される．
- **開発プロセス**：コア資産の開発，コア資産を用いた製品開発，およびこれらの管理プロセスなど，非 SPL 開発にはないプロセスが必要になる．
- **コア資産投資**：製品系列全体を視野に入れたコア資産整備のための投資が必要であり，妥当な期間での回収が求められる．
- **組織的理解**：製品開発にかかわる要員だけでなく，経営層も含めた組織的な SPL 開発に対する正しい理解が不可欠である．

なお，これらの項目は，非 SPL 開発ではまったく考慮しなくてよいというものではない．SPL 開発では，これらの課題の解決がとりわけ重要であるということを指摘するのが本稿の意図である．以降，それぞれの課題について述べる．

SPL 開発におけるソフトウェア構成管理の課題

ソフトウェア構成管理は，CMMI（能力成熟度モデル統合）のレベル 2 におけるプロセス領域とされているように，ソフトウェア開発を管理可能状態にするための基盤となるプラクティスである．

SPL 開発におけるソフトウェア構成管理は，非 SPL 開発に比べて大幅に複雑化する．すなわち，コンポーネントレベルでの構成管理という一般的なものに加えて，製品系列内でのコア資産の構成管理，製品系列をまたいだコア資産の構成管理が必要になる．

例として，図 -1 に示したような，2 つの製品系列 A と B から構成される SPL 開発を考える．製品系列 A と製品系列 B は，それぞれ異なる製品群だが，自社のコアコンピタンスに相当するソフトウェア機能をコア資産として共通化した場合を想定している☆1．また，製品 A1 から製品 A2 はバージョンアップ，製品 A2-1 と製品 A2-2 はバリエーションの違いで，実務的な例では，仕向地対応(国内，北米，欧州など)の違いと思っていただければよいだろう．また，それぞれの製品を構成するソフトウェアを，①製品系列に横断して利用されるコア資産（Cross-Line Core ; CLC），②製品系列内で利用されるコア資産（Line Specific Core ; LSC），③製品固有の部分（Product Specific ; PS），の 3 種類に分けて考える．なお，図中の赤太枠で囲んだ部分は，その製品開発プロジェクトで新規に作成された，またはその製品から新たに適用されたソフトウェア部分を示している☆2．

前提として，SPL 開発である以上，コア資産として開発されたソフトウェア部分は，それ以降において当該

☆1 たとえば，GE 社のエンジンと医療機器には，いずれも遠隔診断の機能が提供されている．これらが内部的に SPL 化されているかどうかは別として，このように，製品系列をまたいだコア資産の横展開は，経営資源を有効活用し，コアコンピタンスを積極展開するのに有効である．

☆2 製品開発プロジェクト内でコア資産を開発するのか，あるいは製品開発プロジェクトと並行して初期段階からコア資産を開発するのかは，SPL 導入戦略に依存する．

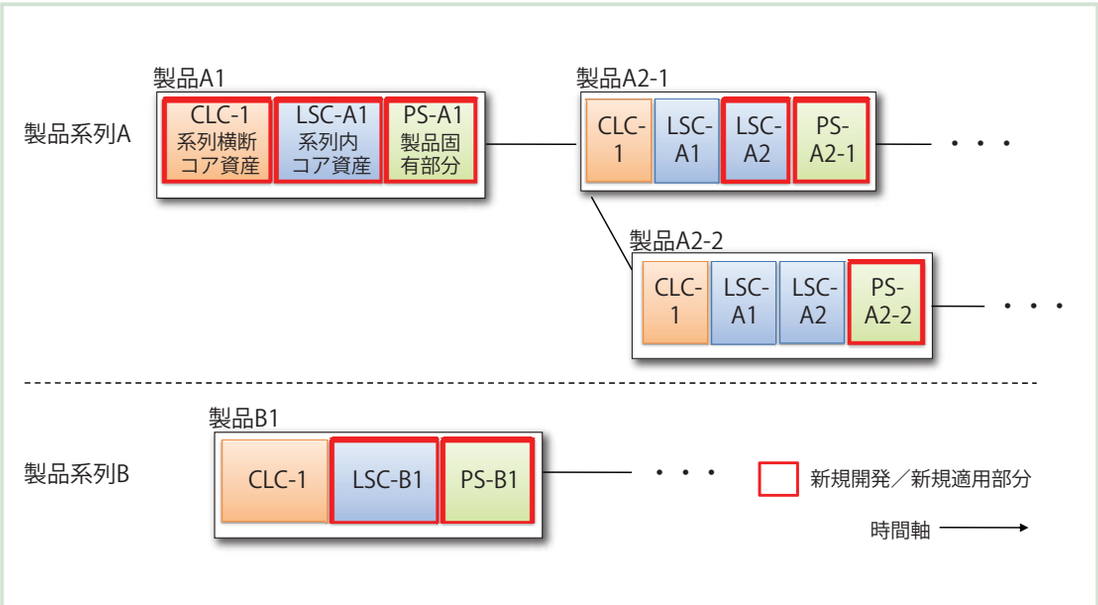

図 -1　2 つの製品系列 A と B におけるソフトウェア構成の例

機能を必要とする製品では必ず再利用されるという状況を想定する（これは SPL 開発の利点を活かした典型的な再利用の状況である）．このとき，次のシナリオのそれぞれにおいて，どの範囲まで構成管理の対象となるかを考えてみよう．

(1) 製品 A2-2 の開発または運用時に，製品固有部分 PS-A2-2 内の潜在バグが発見された．
(2) 製品 A2-2 の開発時に，系列内コア資産 LSC-A1 内の潜在バグが発見された．
(3) 製品 A2-2 の開発時に，系列横断コア資産 CLC-1 内の潜在バグが発見された．
(4) 製品 B1 の運用時に，系列横断コア資産 CLC-1 内の潜在バグが発見された．いま，製品 A2-1 の開発中である．

(1) の場合，これは製品 A2-2 に閉じた問題であり，製品 A2-2 におけるコンポーネントレベルでの構成管理の範囲で対処が必要となる．これは SPL 開発に固有の話ではない．一方，(2) の場合，製品 A2-2 と同じ分岐である製品 A2-1，および先行製品 A1 において，発見された潜在バグへの対処が必要になる．また，(3) の場合は，図 -1 に現れているすべての製品においてバグの対処が必要になる．これら 3 つのシナリオは，いずれもバージョンを遡った対処の例であるが，シナリオ (4) の場合は，先行製品の運用時に発見されたバグを，すでに開始している後続バージョンの開発プロジェクトで対処しなければならないといった状況になる．これは，後続バージョンの開発プロジェクトでの手戻りを誘発し，コスト増やリリース遅延の引き金となるリスクがある☆3．

このような品質情報の製品間および製品系列間での伝播は，SPL 開発における利点の 1 つでもある．しかし，この単純な例を見ただけでも明らかなように，SPL 開発に即したソフトウェア構成管理が適切に行えていないと，品質情報の伝播はメリットではなくデメリットとなり，プロジェクトを混乱に陥れるリスク要因となる可能性が高い．しかし，これを適切に管理しない限り，SPL 開発のメリットは限定的にしか享受できないであろう．

さらに，コア資産に含まれるコンポーネントレベルでの構成管理も必要である．たとえば，従来は不可変としていたコア資産の一部分を，新製品以降では可変部分として扱う必要が生じたとする．このとき，コア資産に含まれるコンポーネントに対して，異なるバージョンのコンポーネントが生み出される．また，別の例として，先ほどのシナリオ (2) において品質情報が系列内製品に伝播した際，製品 A1 では当該バグの影響がないことが確認されたとする．この場合，問題がないのに製品 A1 を変更する必要はないという判断がなされるのが，実務における現実的な対応であろう．コア資産のコンポーネントに異なるバージョンが生じた状況で，さらにシナリオ (2) に相当する新たなバグが生じた場合，製品 A1 での対応は慎重にしなければならない．このように，コア資産のコンポーネントレベルでの構成管理も適切に行うことが求められる．

以上の通り，構成管理の適切な実施は，SPL を実践する上での肝となる．このような課題を十分に認識した上で，構成管理のポリシーおよび変更管理ルールを定め，適切に構成管理を実施することが求められる．

☆3　このような状況におけるバグ修正コストを考慮した SPL コスト評価研究として文献 4）がある．

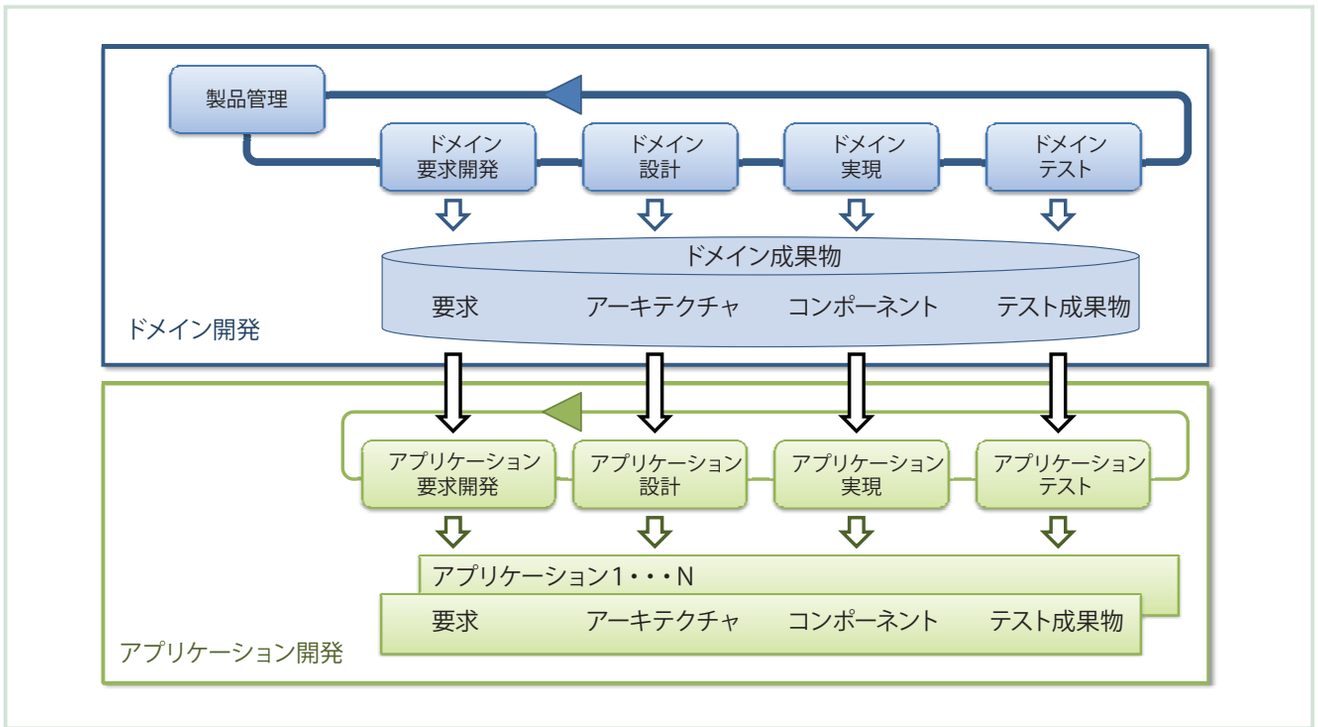

図 -2　SPL 開発プロセスのフレームワーク（文献 5），一部の用語を変更）

SPL に特有の開発プロセスの整備

続いて，SPL における開発プロセスの問題について述べる．ソフトウェア開発における汎用的なライフサイクルプロセスは，一般に SLCP (Software Life Cycle Process) と呼ばれる国際規格 ISO/IEC 12207（JIS X 0160）として提示されている．この規格では，ソフトウェアを開発するメインのプロセスだけでなく，開発を支援するプロセスや，組織的な対応を要するプロセスまで網羅的に示されており，国際的にも評価が高く，実務でも広く参照されているプロセスである．また，SLCP を参考にして，組込みソフトウェア向けの開発プロセスを示した『組込みソフトウェア向け開発プロセスガイド』が IPA/SEC により提示されている．同ガイドは，SLCP に比べて具体的かつ実践的なプロセス定義を与えている．

しかし，これらのプロセス定義には SPL 開発に特有の開発プロセスは言及されておらず，SPL の実践にあたってはその整備が必要である．SPL 開発では，特に，ドメイン開発とアプリケーション開発の視点から開発プロセスを整備する必要がある（**図 -2**）．すなわち，製品系列内または製品系列を横断したドメインにおけるコア資産を開発するプロセスと，コア資産に対して製品固有部分を追加して製品を開発するアプリケーション開発に分けられる．これらのプロセスを切り分けて考え，それぞれどのようなアクティビティを実施すべきか定義しなければならない．特に，ドメイン開発では製品系列における製品群の管理というプロセスが必要である．図 -2 の各プロセスでのアクティビティは文献 5）が詳しいので，詳細はそちらを参照されたい．

また，先に述べた SPL 開発におけるソフトウェア構成管理のプロセスを明確に定義する必要がある．特に，コア資産の変更やバグ修正がなされた際の対応方法を明確に定義し，これを組織全体で運用し，コア資産の整合性および一貫性を維持する仕掛けが必要である．

コア資産開発の投資効果の評価

SPL 開発において，コア資産開発は将来への投資である．投資する以上は，投入したコストを適切な期間で回収し，期待された効果が確実に得られるようにしなければならない．そのためには，まず，SPL への移行が十分にペイすることを，十分な精度で事前評価することが求められる．その際に役立つのが先行事例であり，事例の経験から構築された汎用的なコストモデルである．

ここでは，SPL 開発のコストモデルとして国際的な SPL コミュニティで標準的に参照されているものを 2 つ紹介する．また，これらのモデルに基づいて，段階的に SPL 化する場合のコスト評価を筆者が行った結果を紹介する．

◉ SIMPLE コストモデルフレームワーク

Clements らは，SIMPLE (Structured Intuitive Model for

Product Line Economics) という SPL 開発のコストモデルのフレームワークを示している[3]．SPL 開発のコスト評価に関する論文の多くは，このフレームワークに基づいて議論されている．

SIMPLE フレームワークでは，SPL 開発におけるコスト要素として次の 4 つを挙げている．

- **組織の適応**：SPL 採用に伴って発生する組織的な対応コスト．組織編成，プロセス改善，研修などのコストが含まれる．
- **プラットフォームの構築**：コア資産の開発に必要なコスト．共通性および可変性の分析，コア資産の開発，コア資産の試験にかかるコストなどが含まれる．
- **製品固有部分の構築**：プラットフォームから独立した製品固有部分の開発に必要なコスト．
- **共通部分の再利用**：コア資産を再利用する際にかかる追加コスト．再利用するコア資産の特定，コア資産の適応，統合試験の実施などのコストが含まれる．

SIMPLE コストモデルは，あくまでコストモデルのフレームワークを提示したモデルであり，それぞれのコスト要素を具体的に算出する手順を提示しているわけではない．このフレームワークに基づいて具体的な SPL 開発のコストを算出するには，何らかの異なる手法を用いるか，またはビジネスケースを作成してこのフレームワークに当てはめて考える必要がある．

SIMPLE フレームワークを用いて，現実的な SPL 開発の状況を想定してコスト予測を行った文献が示されている[2]．この文献が示したシナリオにおいて，SPL 開発の 2 製品目の時点で，非 SPL 開発の累積コストを下回るという試算が示されている．SPL 開発が 2 製品目または 3 製品目あたりでペイするというのは，SPL 開発の適用事例でもしばしば報告されている目安であり，シナリオを自社の状況に照らし合わせて検討すると参考になるだろう．

◉COPLIMO コストモデル

ソフトウェアエンジニアリング分野における代表的なコスト見積りモデルと言えば，Boehm らによる COCOMO II である．COPLIMO (Constructive Product Line Investment Model) コストモデル[1]とは，COCOMO II を SPL 開発に適応させたコストモデルである．

COPLIMO は，SIMPLE フレームワークのうち「組織の適応」以外の 3 要素について，ソースコード行数やプロジェクト特性などに関するパラメータを用いて開発コストを見積もる．その基本的な構造は，COCOMO II とほぼ同じである．COPLIMO における SPL 開発に固有のパラメータは，次の 3 項目に集約される．

- **ソースコード行の構成**：製品固有のコード，修正せずに再利用されたコード，修正した上で再利用されたコードの 3 分類について，製品全体に占めるそれぞれの比率．
- **コア資産化コスト**：あるコンポーネントをコア資産として将来の製品で再利用できるようにするためにかかる相対的追加コスト．
- **コア資産の再利用コスト**：コア資産を再利用する際にかかる相対的追加コスト．

COPLIMO コストモデルを用いる際には，ソースコード行数の見積り値や，上記のパラメータの値などをある程度の確度で見通せていることが望ましい．また，モデルのロジックが必ずしも単純ではないため，経営層に SPL 導入効果を説明するための大雑把な試算というよりは，技術スタッフに対して論理的にコスト試算を示すのに向いている．

COPLIMO コストモデルを用いて，SPL 開発の効果を試算した文献が示されている[1]．この文献の試算でも，2 製品目の時点で SPL 開発の累積コストが非 SPL 開発と等しくなり，それ以降の製品開発において SPL 導入がペイすると示している．

◉段階的 SPL 導入戦略でのコスト評価

これまで紹介してきた SPL 導入効果の試算や，その他の文献で示されているものは，いわゆる proactive な開発戦略と呼ばれるような，最初にコア資産を一通り揃えて，それ以降は製品固有部分を開発するという状況を想定したものがほとんどである．これを，新規開発を繰り返して複数製品を提供した場合と比較している．

しかし，実務の現実を見ると，まったくの新規開発はごくわずかであり，多くのプロジェクトは既存製品の流用開発である．このような状況を鑑みると，proactive な開発戦略と新規開発の比較というのは必ずしもフェアではなく，より現実に即した比較が求められる．また，proactive な開発戦略により，コア資産を最初から一通り整備することは，市場変化への対応力をかえって鈍らせることになる恐れがあり，確固とした製品ロードマップを描けない限り，リスクがある．現実的な製品開発の場面を考えると，むしろ，段階的な SPL 導入のコストと，流用開発のコストを比較評価するのが望ましい．

このような問題意識のもとに，筆者は，COPLIMO コストモデルをベースにして，SPL を段階的導入した場合と，従来型の流用開発のコスト比較を試みた[6]．その中で，製品に含まれるソースコード行を，次の 4 種類に分類して考えた．

- コア資産の再利用コード行 (core reuse)
- 非コア資産の再利用コード行 (non-core reuse)
- 製品固有のコード行 (unique)

No.	コード行（KLOC）				
	new core	core reuse	non-core reuse	unique	total
1	20	0	60	20	100
2	30	20	30	20	100
3	0	50	10	20	80

表 -1　段階的 SPL 導入のシナリオ

No.	コード行（KLOC）				
	new core	core reuse	non-core reuse	unique	total
1			80	20	100
2			80	20	100
3			60	20	80

表 -2　流用開発のシナリオ

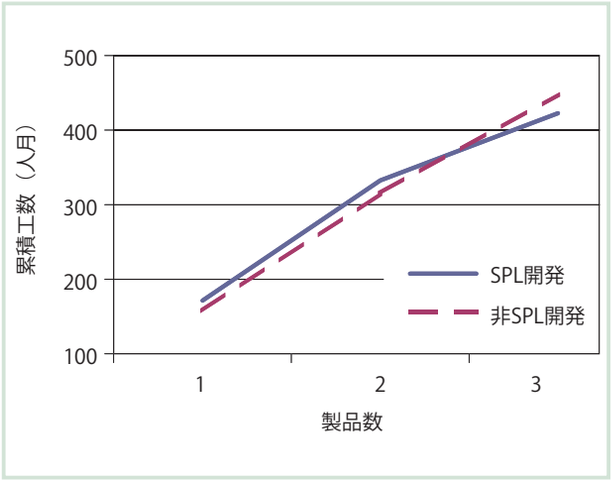

図 -3　段階的導入時の SPL コスト評価の例

• 新規作成コア資産のコード行 (new core)

この分類を用いて，**表 -1** に示した段階的 SPL 導入シナリオと，**表 -2** の流用開発シナリオを想定した．これらのシナリオでは，総コード行数が 100KLOC（10 万行）ないし 80KLOC の製品を 3 つ開発することを想定している．表 -1 では，たとえば第 1 製品の開発で新規に作成したコア資産 20KLOC を，第 2，第 3 製品で再利用するという状況を示している．一方，表 -2 では，第 1 製品を開発するのに既存コードから 80KLOC を流用していることを示している．単純な例ではあるが，比較的，現実の状況に近いシナリオであると考えている．

COPLIMO コストモデルを適用するにあたっては，先に述べたとおり，「コア資産化コスト」と「コア資産の再利用コスト」をパラメータとして与える必要がある．ここでは，コア資産化コストを新規開発コストの 156%，コア資産の再利用コストを新規開発コストの 13% と見積もった．また，COPLIMO コストモデルにはないが，non-core コード行を流用する際の再利用コストを新規開発コストの 42% として見積もった☆4．

この前提に基づいて，COPLIMO モデルを用いてコスト比較を行った結果が**図 -3** である．多くの事例（ただし，段階的 SPL 導入と流用開発を比較したものではないが）などで言われているように，やはり，第 3 製品のところでペイするという結果が得られた．一方で，第 1 製品および第 2 製品の時点では段階的 SPL 導入と流用開発のコスト差はわずかであることから，段階的 SPL 導入はコスト面でのリスクを抑制するのに貢献する可能性があるといえよう．

もちろん，シナリオ次第で結果はどうにでもなるため，このシミュレーション結果の妥当性は検証できていないし，そもそも厳密な検証は不可能である．しかし，SPL 導入効果を議論する上では，その出発点としては十分に役立つ試算であると考えている．

SPL 導入のコスト評価研究では，モンテカルロ法を用いたり，リアルオプション理論を用いたりするなど，いくつかの技法の適用がなされている．しかし，多くの場合，このようなシナリオベースでの議論，または 1 つないし 2 つ程度の事例に当てはめた議論が中心である．多変量解析に耐えられるだけのデータ数が蓄積されることはおそらく期待できないので，本稿で示したように，現実の状況を踏まえた試算結果を蓄積し，これを事例と比較していくことが，コア資産の投資効果予測において必要であろう．

SPL 導入効果の組織的理解

これまで述べたように，SPL 開発への移行には，SPL の実践に耐え得るソフトウェア構成管理の導入，ドメインにおける可変性や共通性を分析するプロセスの導入，ドメイン開発プロセスとアプリケーション開発プロセスの整備，コア資産開発投資の効果算定など，いくつかの課題がある．これらはすべて SPL 化のための投資である．投資が必要ということは，これはプロジェクトや製品担当者のレベルで意思決定できる範囲を超えており，組織階層のより上位レベルで，経営的観点に基づいた意

☆4　これらの数値の算出は，COCOMO II および COPLIMO に示されたガイドラインに従った．

思決定が必要になる．

ここでは，SPL 導入による効果を組織的認識とするための課題をいくつか挙げる．

◉SPL 導入がもたらす価値の明確化

まず，SPL の導入によってどのような価値が期待されるのか，その種類と，実現レベルを明確にすることから始めなければならない．また，これを経営層だけでなく技術者に対して，分かりやすく伝える必要がある．SPL 導入により期待される価値の例として，次が挙げられる．これらは，字面としてはソフトウェアエンジニアリングで以前から指摘されていることと大差ないが，SPL では，その期待される効果レベルが大きく異なる．

- 開発コストの削減
- 製品品質の向上
- 製品提供時期の短縮
- 保守コストの低減
- コア資産比率増による予測精度の向上
- 顧客満足度の向上
- 利益率の向上
- プロセスの再現可能性の向上

◉SPL 導入効果の根拠の提示

前項に挙げた項目を語る際に，つねに難しいのが，その妥当性をいかに提示できるかという課題である．定性的なものも含めてすべてをコストと利益に換算するという方法もあるが，そのアンチテーゼとしてバランススコアカードが提唱されたように，多面的なデータを用いて導入効果を実証するという地道な努力が求められるであろう．

たとえば，活動原価計算の取り入れによる作業工数およびコストの定量化，売上高や利益率などの財務指標，市場でのシェア占有率，市場に投入した製品数，顧客満足度調査など，多様な観点から SPL 化の効果を説明する努力が求められる．

おわりに

冒頭にも述べたように，SPL 開発におけるマネジメントの課題は，以前から指摘されているソフトウェア開発の課題と重なりが多い．したがって，まずは，ソフトウェア開発一般におけるマネジメント上の課題を解決することが必要である．たとえば，バグの根本原因を探って根源的レベルでの再発防止策を打つ，技術レビューを重視して品質を早い段階から作り込む，定量的な品質コントロールの仕組みをプロジェクトに導入する，組織標準のライフサイクルプロセスを定義して組織内に展開する，ソフトウェア開発における人間的側面（NIH[☆5] 症候群による再利用の妨げなど）を考慮したプロジェクト管理を実施するなど，さまざまな取り組みが必要である．その上で，本稿で述べた課題の解決に向けた組織的な取り組みが必要である．

SPL 開発におけるマネジメントの課題解決は，一筋縄ではいかない．ソフトウェア再利用に関する議論がソフトウェアエンジニアリング分野でこれまでにたびたび行われてきたが，経営的・組織的な視点の欠けた技術論だけでは，長期的に成功した例はほとんどない．SPL は，技術と組織の双方から，再利用の課題を克服しようとする総合的なアプローチである．組織体制を整えた上で，競争優位性を確保できる可能性がある製品分野に SPL を導入し，これを成功させられれば，市場における競争優位性をより確実なものにできる可能性が高まるであろう．

謝辞　本稿のソフトウェア構成管理および段階的 SPL 導入のコスト評価を考えるにあたっては，(株) 日立情報制御ソリューションズの大島啓二氏，桜庭恒一郎氏，舟越和己氏，渡辺滋氏らとの議論によるところが大きい．また，本特集のエディタより貴重なコメントを頂戴した．ここに付して謝意を表す．

参考文献

1) Boehm, B., *et al.* : A Software Product Line Life Cycle Cost Estimation Model, *Proc. ISESE'04*, pp.156-164 (2004).
2) Böckle, G., *et al.* : Calculating ROI for Software Product Lines, *IEEE Software*, Vol.21, No.3, pp.32-38 (2004).
3) Clements, P. C., et al. : The Structured Intuitive Model for Product Line Economics (SIMPLE), *Technical Report CMU/SEI-2005-TR-003*, Software Engineering Institute, Carnegie Mellon University (2005).
4) Nonaka, M., *et al.* : Impacts of Architecture and Quality Investment in Software Product Line Development, *Proc. SPLC 2007*, pp.63-73 (2007).
5) Paul, K., Böckle, G. and van der Linden, F. : *Software Product Line Engineering*, Springer-Velag (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，エスアイビー・アクセス (2009)).
6) 野中　誠：ソフトウェアプロダクトライン開発と流用開発のコスト比較，ウィンターワークショップ 2009・イン・宮崎論文集，情報処理学会ソフトウェア工学研究会，pp.85-86 (2009).

(平成 21 年 3 月 5 日受付)

野中 誠(正会員)
nonaka-m@toyonet.toyo.ac.jp
1972 年生．早稲田大学理工学部工業経営学科卒業，同大学院理工学研究科経営システム工学専門分野修了，博士後期課程単位取得退学．早稲田大学助手を経て，2003 年東洋大学経営学部専任講師，2006 年現職．

☆5 Not Invented Here：自分たちが開発したものではないと再利用したがらない技術者の傾向を表す言葉．