3

プロダクトライン開発への移行技術
既存シリーズ製品の再構築とコア資産管理

位野木 万里　東芝ソリューション(株)

「プロダクトライン開発は難しい」,「今の経営状況ではプロダクトラインへの投資は困難」という声をよく聞く．はじめから完璧なプロダクトラインを目指す必要はない．既存製品の成果物を活用し，少ないコストで，緩やかにプロダクトライン開発へ移行する方法もある．こうした方法により，場当たり的な再利用のスタイルを改善することが，組織の開発力強化に繋がる．また，経営状況が回復した際に，生産性や品質を向上させる大規模なプロダクトライン開発へもスムーズに取り組めることが期待できる．

プロダクトライン型の再利用の重要性

ソフトウェア開発企業は，生産性の向上，品質の安定，開発リードタイムの短縮のため，シリーズ製品の開発に取り組んできた．シリーズ製品の開発とは，製品のコンセプト，機能，技術基盤を共通化した上で，グレードや仕向地別にバリエーションを開発し，さまざまな顧客のニーズに対応する再利用の取り組みである．

あらかじめ再利用するためのソフトウェア資産を構築するには投資が必要である．厳しい経営状況下では，多大な先行投資は現実的ではない．また，技術，市場，環境の変化が激しい情報産業では，開発に必要なソフトウェア資産の予測も困難である．だからといって，シリーズ製品開発のベースになる基盤を作っておかなければ，既存の製品の場当たり的な再利用の繰り返しによって，製品間に重複した部分が多数発生する，ソースコードが複雑化する，不具合を何世代にもわたって継承するなどの問題が発生する．その結果，メンテナンスコストがかさみ再利用効果を得られない，または，技術，市場，環境が変化しても新製品開発に乗り出すことができないというリスクが高まる．

現在，求められているのは，既存のシリーズ製品の成果を用いてソフトウェア資産を構築し，投資リスクは低く抑えた上で，従来型の再利用によるシリーズ製品開発から，プロダクトライン型の開発へ移行することである．このようにすることによって，今後の製品開発において，メンテナンスコストが増大するリスクを軽減し，さらに，経営状況が回復した際には，スムーズに製品開発計画を拡大させ，製品開発のコストダウンやリードタイムの短縮が期待できる．

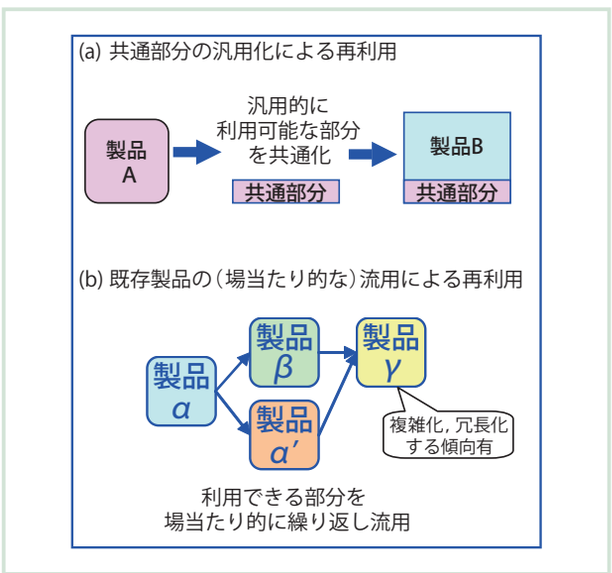

図 -1　従来型の再利用によるソフトウェア製品開発

本稿では，既存のシリーズ製品をプロダクトライン開発へ移行することに焦点をあて，そのための手順，ソフトウェア資産の抽出・管理方法について解説する．

移行プロセス

プロダクトライン開発へ移行するとは，従来型の再利用の開発から，プロダクトライン型の開発のスタイルに変革することである．従来は，開発済みのソフトウェアから汎用的な部品を抽出し，これらを再利用の対象とする(図 -1(a))，または，過去に開発したソフトウェア(シリーズ製品)から利用できる部分を流用するといった再利用がよく行われている(図 -1(b))．その結果，ソフト

ウェア開発全体に対する再利用部分はわずかで，生産性にそれほど寄与しないことや，既存の製品の場当たり的な再利用の繰り返しによって，製品間の重複の増大，ソースコードの複雑化，潜在的な不具合の継承などの問題が発生していた．

プロダクトライン開発では，シリーズ製品を対象に再利用範囲を決定し，コア資産と呼ばれるソフトウェア資産を整備し，コア資産を再利用しソフトウェア製品を開発する．コア資産は，アーキテクチャ，コンポーネント，ドキュメント，開発・テストツールなど，ソフトウェア開発の過程で作成または利用される，さまざまな成果物を含む☆1．プロダクトライン開発では，あらかじめ想定したスコープを対象にコア資産を無駄なく準備することで，生産性の高い開発を行うことがねらいである（**図 -2**）．

従来型の再利用からプロダクトライン型への移行には，各企業の試行錯誤が必要になる．Krueger と Schmid らは，プロダクトライン開発への移行方法を次のように分類している．

◉Krueger による 3 つの分類

Krueger は，企業によるプロダクトライン開発の採用方法を，以下の 3 つのタイプに類型化している [1]．

Proactive（先行型）

あらかじめ，製品開発ロードマップをカバーする製品間の共通部分と，各製品に固有の可変部分☆2 に相当する成果物を開発しておき，これらを用いて製品を開発する．このタイプは多大な先行投資が必要になる．市場が安定し予測可能な場合が向いている．長期間のリソース投入が可能な組織に適用しやすい．

Reactive（反応型）

1 つないし数個の製品開発の都度，再利用の対象とする共通部分と可変部分を，繰り返し開発する．先行投資は少額で対応できる．開発する製品全体をあらかじめ想定する必要はないため，予測が困難な市場にも適用可能である．ただし，再利用対象とする部分を追加開発することになるため，よく検討されたアーキテクチャが提供されていないと，リエンジニアリングコストがかさむので，注意が必要である．

Extractive（抽出型）

既存の製品をプロダクトラインの出発点とする．既存の製品を多大なリエンジニアリングなく再利用するというライトウェイトな移行方法である．上記の Proactive と Reactive の中間的なタイプである．市場を先読みして製品開発ロードマップは描くものの，既存の資産をベースラインとして初期投資は少額に抑える．予測しやすい市場の方が向いている．既存の成果物を修正せずに利用できれば，投資リスクはさらに小さくできる．対象ドメインでの開発経験と成果物が蓄積されており，プロダクトライン開発に早く移行したい組織に適合する．

◉ Schmid らによる 4 つの分類

Schmid らも企業のプロダクトライン採用のシナリオを以下の 4 つに分類している [2]．

Independent 型（独立型）

新規市場に向け，新規にプロダクトラインを構築する．既存の製品やプロダクトラインは想定していない．

Project-integrating 型（プロジェクト統合型）

複数の製品が新規市場投入に向けてすでに開発されている．同一の基盤の再利用により製品開発ができるように，製品群を統合する．

Reengineering-driven 型（リエンジニアリング駆動型）

既存の製品群が存在するが，プロダクトラインとしては活用できていない．製品開発コストが高い，新製品開発に柔軟に対応できないなどの開発の壁に直面し，リエンジニアリングを必要としている．

Leveraged 型（強化型）

新規市場への取り組みのため，既存のプロダクトラインの代わりに，新規のプロダクトラインを構築する．

求められるプロダクトライン開発像

新規市場向けでまとまった投資が可能であるなら Proactive 型のプロダクトライン開発が妥当である．すなわち，同一ドメインの製品開発を想定し，開発スコープを定義した上で，共通部分，可変部分をコア資産として先行して（つまり，Proactive に）開発しておき，そのようにして定義したコア資産を再利用して，製品開発を行うというものである（図 -2）．しかしながら，このような取り組みは予測可能な市場に限定され，現在の急速な景気低迷からも分かるように，市場の予測は困難なことが多い．また，予測可能な市場が存在しても，競合他社も参入しやすくなる．したがって，企業にとり，よほどの技術的な強みがなければ，先行投資はしにくい．

Krueger と Schmid らが示しているように，現実的には，あるドメインの特定の製品を開発し，一定の成果が確認できたところで，少しずつ展開市場を拡大し，コア資産を拡充していくことが安全と考えられる．すなわち，今後の市場をある程度想定した上で，既存製品の成

☆1 Clements, P. and Northrop, L. : Software Product Lines: Practice and Patterns, Addison-Wesley（2001）．前田卓雄訳：ソフトウェアプロダクトライン，p.652，日刊工業新聞社（2003）．

☆2「可変部分」という用語は筆者が定義した．可変部分とは，製品を構成する成果物のうち，製品ごとに異なる部分のことであり，バリエーションポイントとバリアントから構成されるものを想定している．

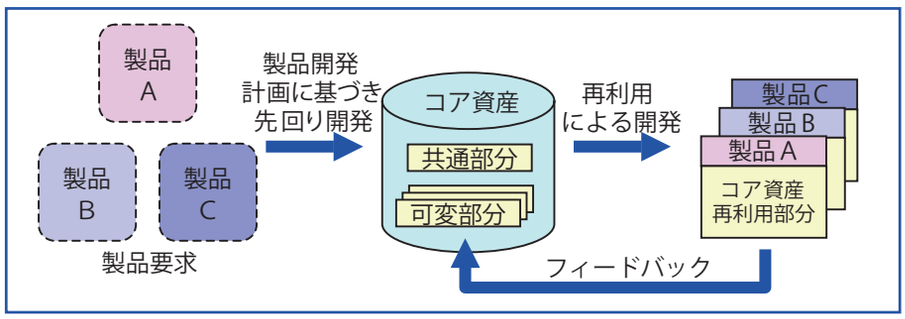

図 -2　先行投資を前提とする
プロダクトライン開発の考え方

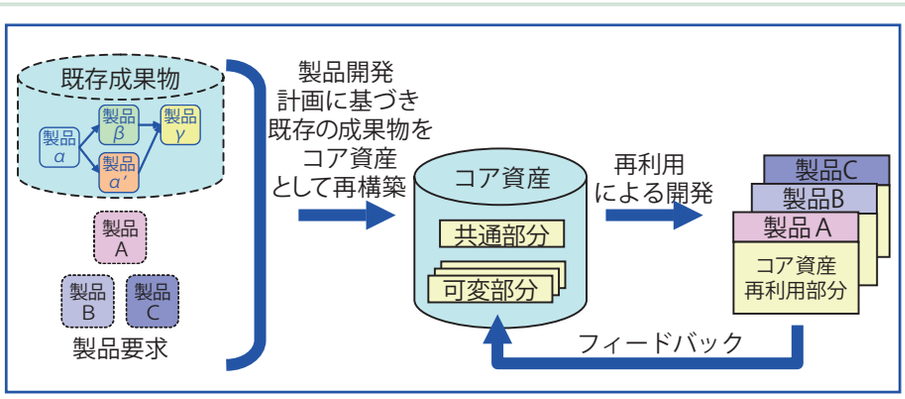

図 -3　企業で求められている
プロダクトライン開発の考え方

果物を用いて，コア資産として再構築させるという方式，Extractive で Project-integrating といったアプローチが求められているといえる（図 -3）．

既存資産からプロダクトラインを作る

既存のシリーズ製品をプロダクトライン開発へと移行するための手順，技術についてさまざまな研究がなされている．図 -3 に示した既存のシリーズ製品からプロダクトラインへ再構築させる手順について2つの例を紹介する．

◉リエンジニアリングによるプロセス

Kang らによりフォワードエンジニアリングとリバースエンジニアリングを組み合わせた手法が提案されている [3]．この手法では，次の①〜⑤のステップでプロダクトライン開発へ移行するとしている（図 -4）．①既存アプリケーションからコンポーネント，アーキテクチャに関する情報を抽出する，②既存アーキテクチャ情報からフィーチャモデルを抽出する，③今後の拡張性，市場のニーズ，技術の変化などを想定して，新規にフィーチャや可変性に関する情報（バリエーションポイントやバリアント）を追加し，フィーチャモデルをリファインする，④新しいプロダクトラインのアーキテクチャを定義する，⑤コンポーネントを設計，開発する．

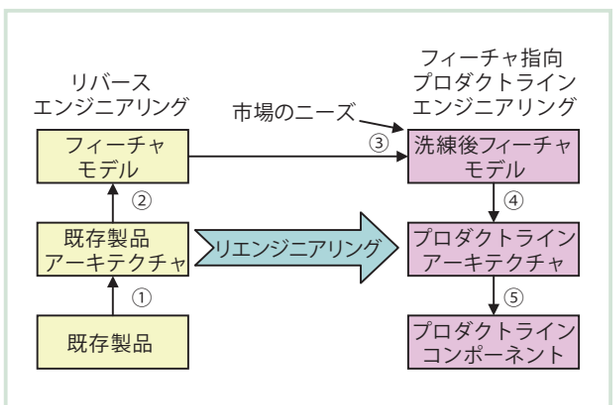

図 -4　既存資産からプロダクトラインを再構築するプロセス例（1）

図 -3 に示した求められるプロダクトライン開発像への移行には，①〜⑤を繰り返して少しずつ既存の成果物をコア資産化することや，④や⑤のステップで既存製品の成果物をどれほど活かせるかがポイントになる．

◉問題領域と解決領域を意識したプロセス

Beuche は，問題領域と解決領域を切り分けた手法を提案した [4]（図 -5）．この手法では，まず，プロダクトライン導入前の既存の製品群を入力として，製品間の関連を調査する．製品間の関係とは，何らかの共通部分がすでに存在しているかどうか，共通部分に基づく開発

が体系的に行われているかどうか，派生開発の状況と派生した製品間の関係の複雑さなどである．これらの調査結果に基づき，既存の成果物をどの程度利用するのかといったシナリオを定義する．たとえば，コンポーネントの再利用はせず，製品ドメインの概念特性を抽出する目的で既存の成果物を参照するのか，または，既存の成果物をできるだけ再利用するのかなどを定義する．続いて，問題領域と解決領域に分けて可変性分析と各モデルの定義を行う．問題領域のモデル定義では，ユーザマニュアルや開発ドキュメントなどから，バリエーションポイントとその制約を抽出し，フィーチャモデルを構築する．解決領域については，設計ドキュメント，ソースコードなどを入力とし，たとえば，C や C++ 言語であれば“#ifdef”で切り替えてある個所を手がかりにバリエーションポイントを特定する．次に，製品全体のアーキテクチャを定義し，バリエーションポイントと対応づけて既存の成果物から再利用するコンポーネントをコア資産として特定する．また問題領域で定義したフィーチャと解決領域で特定したコア資産との対応関係を定義することで，コア資産を利用した製品開発の方法を明らかにしておく．

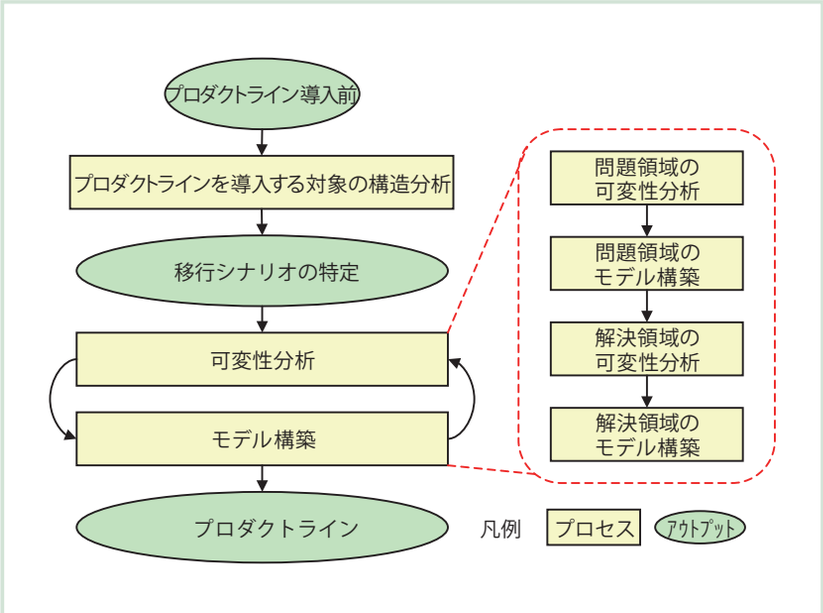

図 -5　既存資産からプロダクトラインを再構築するプロセス例(2)

◉既存資産の再構築シナリオの重要性

Kang と Beuche の 2 つの手法を総括すると，既存シリーズ製品からプロダクトラインを構築するには，(1) 現状を把握する，(2)バリエーションポイントを特定する，(3) コア資産の候補を抽出する，(4) コア資産として再構築する，というようにまとめることができる．

現状把握のために既存の資産を詳細に調査することもコストがかかる．今後の製品開発で必要となるバリアントやリスクが高い部分など，調査の範囲を絞り込んだ上で取り組むことが重要である．

現状，求められているのは，既存のものをできるだけ再利用したプロダクトラインの再構築である．再構築の作業量や複雑さは，既存製品の成果物の再利用方針やそれらの品質によって変化する．不具合もなく，新規製品にも再利用できる既存製品の成果物が豊富に蓄積されていれば，再構築のためのコストも発生せず，スムーズにプロダクトライン開発へ移行できると考えられる．しかし，既存製品の成果物中に，新規製品開発には不要の成果物が取り出しにくい状態で混在している場合には，再構築のために無駄な部分を取り除くコストが発生する．

経営者やプロジェクトオーナーにとって，既存製品の成果物の状態，組織の方針，市場予測，投資予算に基づき，既存資産をどこまで活用するのかといった移行のシナリオの選択が，重要な意思決定事項になる．

既存資産を再構築する技術

前述したプロセスにおいて利用し，既存のシリーズ製品の成果物から，コア資産を抽出し再構築させるための技術を 4 つ取り上げて紹介する．

◉ドキュメントからコア資産を抽出する

ドメインのコア資産を発見する技術として，CaVE (Commonality and Variability Extraction Approach) がある [5]．これは，ユーザドキュメントや開発ドキュメントにフォーカスし，問題領域のバリエーションポイントとバリアントを発見する方法である．図 -4 の Kang らのプロセスでは，②の作業を進める際に適用する手法である．

CaVE のステップを図 -6 に示す．このステップでは，プロダクトラインエンジニアがユーザドキュメントや後述する可変性抽出パターンから，必要と思われるものを抽出する．抽出された結果を用いて，プロダクトラインエンジニアがドメインの共通部分と製品ごとに異なる可変部分(バリエーションポイントとバリアント)を抽出する．最後に対象領域の専門家と一緒にこれらをチェックし，プロダクトラインの要求(または，その一部)を導出する．ここで，可変性抽出パターンとは，可変性を抽出するためのノウハウである．たとえば，次のようなパターンがある．

● Heading Feature パターン：
表題や見出しは通常フィーチャを表す．フィーチャとはユーザにとって重要な特性なので，ユーザドキュメントの目立つ部分に見出せる．表題や見出しを入力にして，フィーチャを抽出する．

● Parameter-Value パターン：
異なるドキュメントに，同じ文やフレーズがあり，それらに続く数値が異なっている場合，これらは異なるパラメータを持ったバリアントになる．

本手法の工夫は，対象領域の専門家とプロダクトラインエンジニアが相互連携して作業を行う点である．プロダクトラインエンジニアは，対象領域の専門知識がなくても作業が行える．また，対象領域の専門家はプロダクトライン開発への移行をプロダクトラインエンジニアに委託することで，本来業務へ集中することができる．

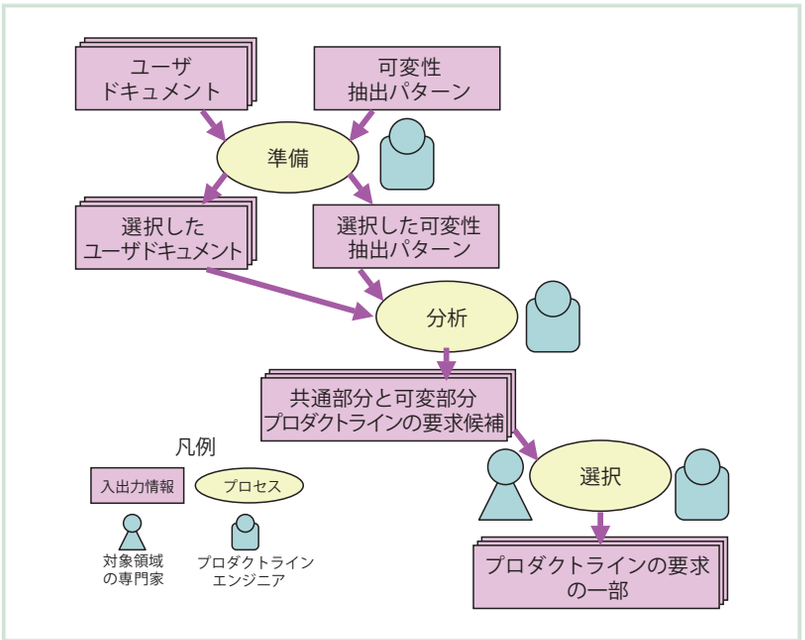

図 -6　CaVE アプローチによるコア資産の抽出プロセス

◉ソースコードからコア資産の候補を抽出する

ソースコードからバリエーションポイントとバリアントの候補を抽出する手法もある．吉村らは，クローン技術を用いて複数の製品間のソースコードを比較することで，コア資産の候補を自動的に抽出する手法を考案した[6]．この手法では，各製品のソースコード中に定義された関数に着目する．関数の単位で製品間の類似度を分析し，分析結果から，共通部分，バリエーションポイント，バリアントを特定する．異なる製品間に存在する等しいコード列のことを製品間コードクローンと呼び，**表 -1** の 4 つのタイプに分類している．製品間コードクローンの分類結果から，対象ドメインの共通性・可変性を評価し，既存製品のソースコードの再利用部分を定める支援ができる．

プロダクトライン開発では，対象ドメインの共通性と可変性の分析が必須であるが，現在市場にある製品の特性を抜けなく把握することは困難である．本手法は，既存製品が大規模・複雑，かつ，さまざまな製品バリエーションが市場に複数投入されている場合に，現状を整理することや，トップダウンで作成したフィーチャモデルにおいてバリエーションポイントやバリアントの抜けを防止することに役立つ．

タイプ	説 明	共通／可変
タイプ 1	関数のインタフェースが同一で，ソースコードの中身が一致する．	共通
タイプ 2	関数のインタフェースが同一だが，ソースコードの一部が異なる．	共通と可変混在
タイプ 3	関数のインタフェースが同一だが，ソースコードの大部分が異なる．	可変
タイプ 4	関数のインタフェースは異なるが，ソースコードに重複がある．関数のコピー後，関数名を変えるなどの作業により発生．	個別に判断する必要有

表 -1　製品間コードクローンの分類

◉アーキテクチャを再構築する

Kang らは文献 3）において，既存の製品情報からアーキテクチャ情報を再構築させるプロセスも定義した（**図 -7**）．そのプロセスは次のように構成している．(1) 既存製品のソースコードからリバースエンジニアリングツール等を用いてオブジェクト図を得る．(2) 主要サービスを形成するオブジェクト群を特定する．たとえば，センサやコントローラなどを特定する．これらをオペレーショナルユニットと呼ぶことにする．(3) オペレーショナルユニットを概念的なコンポーネントと見立て，概念アーキテクチャを定義する．(4) オブジェクト図とオペレーショナルユニットに基づいて，振舞いを分析し，プロセスやスレッドを生成するアクティブオブジェクトを特定する．(5) アクティブオブジェクト間の相互作用を調査・分析し，プロセスアーキテクチャを定義する．

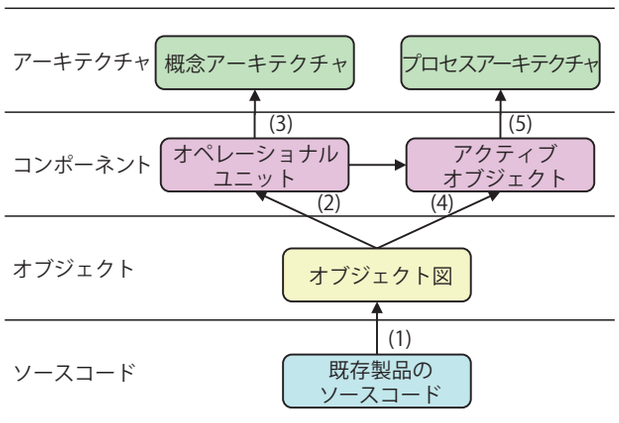

図 -7　既存ソフトウェア製品からアーキテクチャを再構築するプロセス

上記のステップを繰り返すことで，構造に関する概念アーキテクチャと，振舞いに関するプロセスアーキテクチャを再構築する．

既存製品の成果物には，設計ドキュメントに不備がある，現行のソースコードに基づいた内容に更新されていないなど，課題がある場合が多い．現状ソースコードのありのままの構造や振舞いをすべて仕様化するのではコストがかかる．本手法は，ソースコードをベースに，核になる構成要素を中心に，概念構造とプロセスを特定し，現状のアーキテクチャを可視化する．これによって，現状の理解が深まり，新アーキテクチャを検討するという，移行のステップに進みやすくする．

◉コンポーネントを再構築する

既存システムから，再利用可能なコア資産を抽出し，再構築させることも重要である．これにはリファクタリングの技術が有用である．リファクタリングとは，プログラムの振舞いを変えることなくソースコードを変更することである☆3．プロダクトライン開発におけるリファクタリングでは，今後の拡張や可変性を考慮し，重複部分の一般化や特定部分の独立化を行う．**図 -8** にリファクタリングの例を示す．これは，Kang らによるクレジットカードの認証管理システムを題材にしたリファクタリングの事例である☆4．従来は，カード利用者に対する付随サービスとして 2 つの選択肢 (a)，(b) があり，それぞれを実現するコンポーネントを用意していたが，これらは処理の流れが重複しており，今後削除する場合もある．したがって，(s) のように一般化した「割引サービスチェック」コンポーネントを作成しておき，(a)，(b) は，それぞれ (t)，(u) のように差分部分をコンポーネント化する構造にリファクタリングした．このようにすると，新規の製品開発において，全新製品に共通的な仕様は (s) に追加・変更を行い，各製品に特有の仕様は，(t)，(u) の差分部分にのみ変更を加えることで，修正・変更の範囲を限定することができる．

既存製品の成果物をどれほど活用できるかが，プロダクトラインへの投資コストを低減させるポイントである．既存製品があまりにも複雑な作りで，コア資産として利用するには多大なリファクタリングコストを必要とする場合には，既存製品の成果物を捨てる方が良い場合もある．一方，既存製品の成果物の本体に手を入れず，ブラックボックスの部品とそして，そのまま活用することが可能な場合もある．既存製品の成果物の利用範囲と再利用の方法の見極めが重要であり，つねに投資対効果，リスクのバランスを見ながら，コア資産の再構築に取り組む必要がある．

コア資産を管理する技術

再構築したコア資産は，組織内で有効活用できるように適切に管理することが必要である．そのためには，ツール導入も検討すべきである．

◉コア資産の例

コア資産の要素を整理すると**表 -2** のようになる☆5．通常の製品開発で作成する成果物と比較した際の特徴は 2 点ある．1 つ目は，製品開発全体のロードマップがあ

☆3 Fowler, M., Beck, K., Brant, J., Opdyke, W. and Roberts, D. : Refactoring: Improving the Design of Existing Code, Addison-Wesley (1999).

☆4 Kang, K. C., Lee, J. J., Kim, B., Kim, M., Seo, C. and Yu, S. : Re-engineering a Credit Card Authorization System for Maintainability and Reusability of Components - a Case Study, Proc. of 9th International Conference on Software Reuse (ICSR2006), pp.156-169 (2006).

☆5 Klaus, P., Böckle, G. and van der Linden, F. J.: Software Product Line Engineering Foundations, Principles and Techniques, Springer (2005).

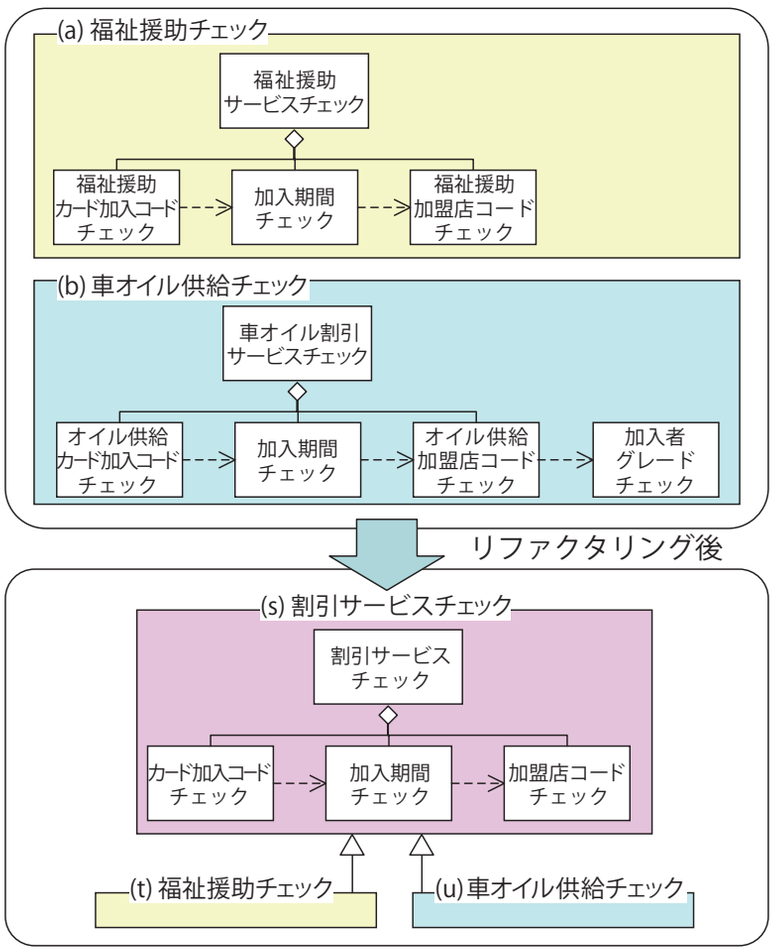

図 -8　コンポーネントのリファクタリング例

項番	ドメイン成果物	説明
1	製品開発ロードマップ (Product Roadmap)	開発対象とする製品の主要フィーチャと製品の市場への投入計画.
2	可変性モデル (Variability Model)	プロダクトラインの可変性に関するモデル．バリエーションポイント，バリアント，依存性や制約，他の成果物との対応関係などを示したもの.
3	ドメイン要求モデル (Domain Requirements)	すべての製品に共通する要求と，個別の製品に固有の要求の仕様.
4	ドメインアーキテクチャモデル (Domain Architecture)	すべての製品の核となる静的／動的な観点からのアーキテクチャと，設計，実現方式，製品の構成方法に関する共通ルールの集合.
5	ドメイン実装モデル (Domain Realization Artefacts)	ソフトウェアコンポーネントとインタフェースの設計と実装結果．実装結果は，ソースコード，コンフィギュレーションファイル，メイクファイルを含む.
6	ドメインテストモデル (Domain Test Artefacts)	ドメインテスト計画，テストケース，テストケースシナリオ．製品開発時のテストで再利用するために，バリエーションポイントの定義も含む.

表 -2　コア資産の構成要素例

り，それに従った製品開発に必要と考えられる成果物を備えている点である．もう 1 つは，全製品に共通の部分と特定製品に固有の部分を可変性モデルとして明らかにし，その可変性モデルと，ドメイン要求モデル，ドメインアーキテクチャ，ドメイン実装モデル，ドメインテストモデルの間での関係付けを明確化する点である．必要に応じ，コア資産には，製品向けのコンポーネント作成やテストの実行を支援するためのツールや開発標準などの成果物も含める．

前述した手法で再構築したコア資産は，表 -2 の要素

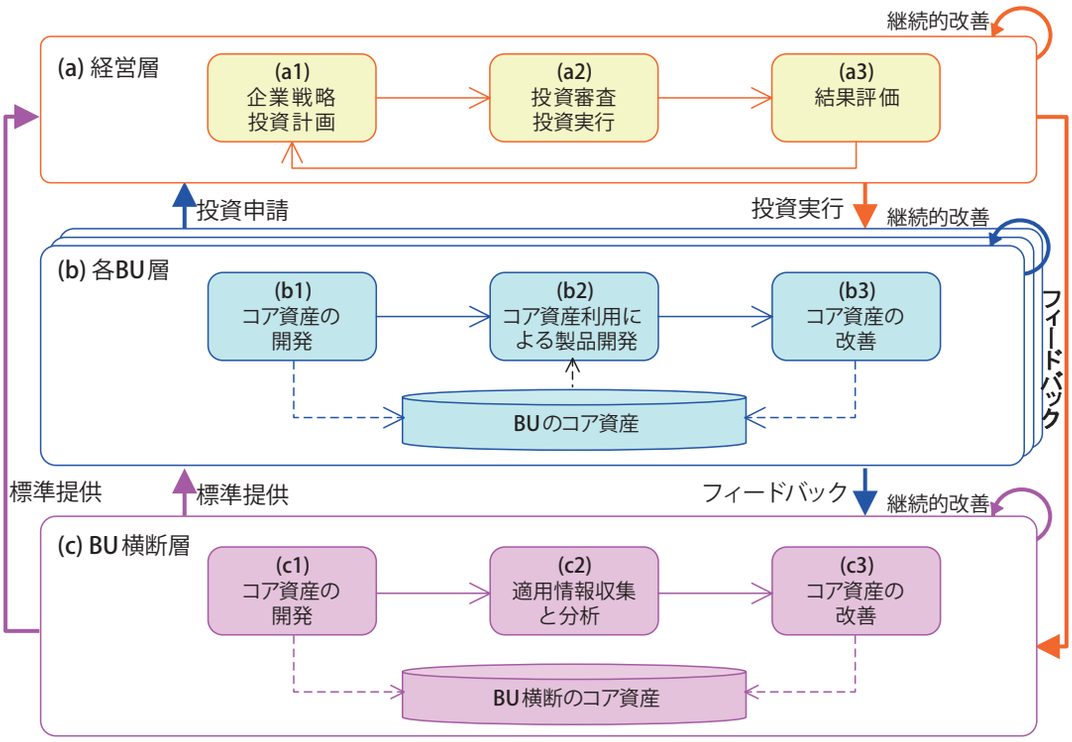

図 -9　経営層，BU 層，BU 横断層によるコア資産管理

のいずれかに対応づけられるものである．既存製品の成果物からプロダクトライン開発へ移行する初期の段階では，表 -2 の要素がすべて揃っていないことも考えられる．いつの段階で，どこまでコア資産を揃えるのか，組織の戦略に基づいて定めることが必要である．

◉コア資産を管理する組織構造

プロダクトラインの開発と維持管理においては，BAPO（ビジネス，アーキテクチャ，開発プロセス，組織構造）の観点からのガバナンスが重要である．コア資産は，対象ドメインを担当するビジネスユニット（以下 BU と略す）単位で作成するものであるが，複数のプロダクトラインを抱える企業では，さらにこうした BU 横断のコア資産というものも必要になる．BU 横断のコア資産は，各社の標準やそのための開発ツール群が相当する．

図 -9 に経営層，BU 層，BU 横断層によるコア資産の管理の関係を示す[☆6]．経営層は，プロダクトライン開発への移行に関して，投資判断を行う．各 BU は，経営層の投資実行判断に基づき，コア資産の開発と，それらを利用した製品開発，製品開発結果からのコア資産へのフィードバックを継続的に行う．上述したように，組織全体で共有する BU 横断のコア資産は，BU 横断層によって管理することが望ましい．BU 横断層が組織標準やツールを管理することによって，各 BU でのコア資産の均質化を促進し，BU 間で重複したコア資産を開発する無駄を防止することが期待できる．

◉コア資産管理ツールの導入

コア資産の管理では，成果物のバージョンに加えて，フィーチャに基づいてさまざまな抽象度の成果物の関連を管理できることが重要である．また，コア資産の利用では，可変性を考慮してコア資産の組合せによる製品ソフトウェアの自動生成の支援が求められている．前者に関しては，構成管理ツールや要件管理ツールを拡張したツールが提案されている．後者については，対象ドメインに固有の言語である DSL（Domain Specific Language）によるモデルベース開発環境を定義できるツールが提案されている．しかし，プロダクトライン開発と製品開発のプロセス全体をカバーするコア資産の管理と利用のためのツールの実現には至っておらず，これは今後の研究課題である．

ツールだけを導入すればプロダクトラインを維持管理

☆6 位野木万里，深澤良彰：プロダクトラインのスコープ定義におけるカバー率と対応度によるコアアセット可視化手法，情報処理学会論文誌，Vol.49, No.2, pp.968-987 (Feb. 2008).

できるわけではない．経営層による戦略，各 BU の状況を踏まえ，BU 横断層によって，コア資産の維持管理のポリシーを確立した上で，ツールの導入を検討することが重要である．

移行に向けて今なすべきこと

移行のための手順，既存の成果物からコア資産を再構築する技術やツールの研究・開発が進み，実用的なレベルに達してきたものもある．しかし，いくらすばらしいコア資産やツールの準備が完了しても，開発した製品が売れなければ意味がない．プロダクトライン開発への移行は，組織の戦略に合致させ，利益の出せる取り組みでなければならない．

一方，現在，投資が十分にできないからといって，まったくプロダクトライン開発に取り組まないのは問題である．既存の成果物を把握し，軽微な改善を施すだけでも，プロダクトライン開発への移行の第一歩となる．企業は，現状の既存資産を改めてよく見直し，地道な改善によってプロダクトライン開発に移行する準備をすることだけでも行うべきである．そのようにすることが，経済状況が回復し投資が可能になった際に，大規模なプロダクトライン開発にスムーズに取り組み，飛躍的な成長をもたらす源泉になると期待できる．

参考文献

1) Krueger, C. : Eliminating the Adoption Barrier, IEEE Software, Vol.19, No.4, pp.29-31 (2002).
2) Schmid, K. and Verlage, M. : The Economic Impact of Product Line Adoption and Evolution, IEEE Software, Vol.19, No.4, pp.50-57 (2002).
3) Kang, K. C., Kim, M., Lee, J. and Kim, B. : Feature-oriented Re-engineering of Legacy Systems into Product Line Assets - a Case Study, Proc. of 9th International Software Product Line Conference (SPLC Europe), pp.45-56 (2005).
4) Beuche, D. : Transforming Legacy Systems into Software Product Lines, Tutorial, SPLC 2007 (2007).（同様のチュートリアル資料は http://www.fuji-setsu.co.jp/files/pure_variants-spl-seminar.pdf より入手可能）
5) John, I., Doerr, J. and Schmid, K. : User Documentation Based Product Line Modeling, Fraunhofer IESE, IESE-Report No. 004.04/E (2003).
6) 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 (Aug. 2007).

（平成 21 年 2 月 6 日受付）

位野木 万里（正会員）
inoki.mari@toshiba-sol.co.jp
1989 年早稲田大学理工学部数学科卒業．1991 年同大学院修士課程修了．同年（株）東芝入社．2008 年早稲田大学大学院博士課程修了．博士（工学）．現在，東芝ソリューション（株）IT 技術研究所に所属．プロダクトライン開発手法ほか，ソフトウェア生産技術の研究・開発，開発部門への適用・普及業務に従事．ACM，IEEE 各会員．