92

大規模・長期的な製品シリーズの開発に有効なソフトウェア開発手法「モデル指向 SPL 開発」の紹介 SPL：Software Product Line[1]

1. 適用手法の全体概要

ソフトウェアプロダクトライン開発は製品系列ソフトウェアの開発に大きな効果をもたらす開発手法であるが、その割には広く普及しているとは言えない。本稿では、ソフトウェアプロダクトライン開発をソフトウェア開発の現場に広く普及させることを目的に、独自に考案したモデル指向開発手法を技術的な基盤として、ソフトウェアプロダクトライン開発を具体的に展開するための開発方法論について紹介する。

以降では、ソフトウェアプロダクトライン開発の概念から始まり、従来の開発方式との違い、開発プロセス、期待できる効果までの一般論を説明し、次に開発方法論の特徴を述べる。その後で、開発方法論が必要になった背景であるソフトウェア開発の現場における問題点と課題を提示し、それらの解決策として開発方法論の具体的な適用内容について解説していく。

最後に開発方法論の評価と確認できた効果、および今後の取組みと課題について述べる。

1.1. ソフトウェアプロダクトライン開発

ソフトウェアプロダクトライン（略して SPL：Software Product Line）開発とは、ハードウェアエンジニアリングの世界で当たり前のように進められているプロダクトライン開発をソフトウェアエンジニアリングの世界にも適応させた開発方式である。プロダクトライン開発は共通部品化という製造分野における戦略的概念を具現化したものであるが、未だ成熟しているとは言えないソフトウェアエンジニアリングでは未定義の技術分野であった。しかし、このような開発方式の導入は時間的な問題であり、ソフトウェアエンジニアリングの世界にも部品化の概念やプロダクトライン開発の考え方が開発方式として提案されるようになってきた。

図 92-1 にソフトウェアプロダクトライン開発の概念と乗用車開発との類似点比較について示す。

[1] 事例提供：株式会社日立産業制御ソリューションズ 渡辺 滋 氏

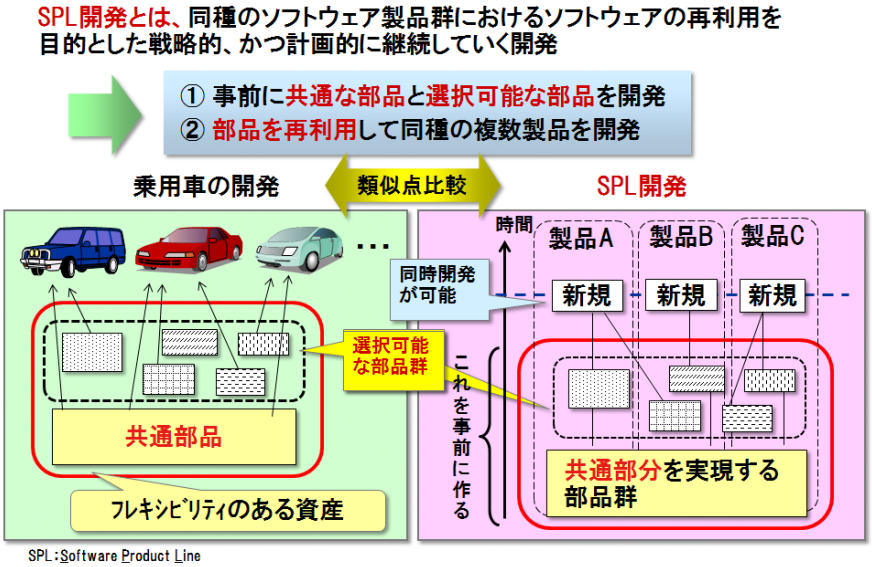


図 92-1 SPL 開発の概念

(1) 従来の製品ソフトウェア開発との違い

SPL 開発が提案される以前は、類似のソフトウェアシステムでも、すべて単一システムとして扱い、過去に開発した類似のソフトウェアシステムをベースに行う派生開発の形態であった。

この開発方式では、過去に開発されたソフトウェアシステムが新規に開発するソフトウェアシステムにどの程度類似しているかが大きな鍵となり、ベースとなる（流用可能な）部分が大きければ大きいほど有効性は高くなる。しかし、方式としては戦略的なものは無く、偶発的な要素が大きい。また、従来の開発方式は単一システムの個別開発であるため、時間的にはシリアルな開発となり、競争の激しい製品市場における戦略的な多品種同時生産のような開発方式は実現できない。SPL 開発は戦略的な多品種同時生産を実現させ、激しい製品競争に対応できる開発方式である。

図 92-2 は、従来のソフトウェア開発と SPL 開発の方式的な違い、および SPL 開発の方式的な特徴について説明している。ここで流用とは、その対象が後続のソフトウェア開発において変更無しに利用できるソフトウェアの範囲であり、独立した部分とは限らない。それに対して再利用可能とは、後続のソフトウェア開発において独立したソフトウェア部品として利用できるということである。上記の意味付けは、必ずしも一般的なものではないが、本稿における説明の範囲内での意味付けとして用いる。

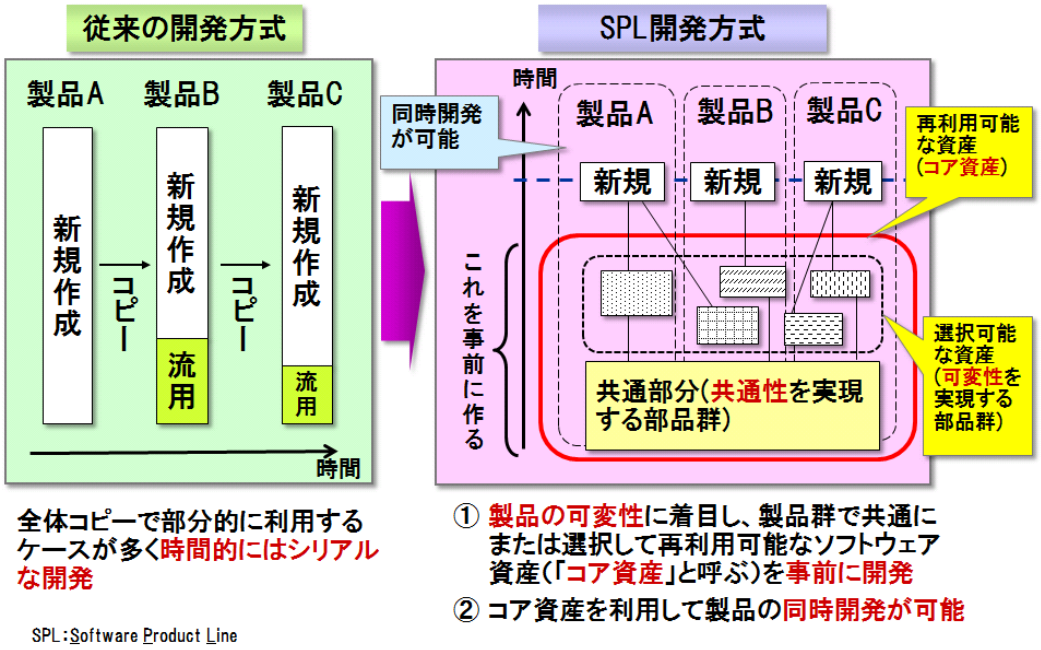


図 92-2 従来の開発方式と SPL 開発の比較

(2) SPL 開発の基盤となる技術革新

SPL 開発はソフトウェア開発の大きなパラダイムシフトであると言われているようであるが、SPL 開発の考え方自体にはそれほど新しいものはなく、要素技術としても特有なものはないと感じている。SPL 開発に特有なものではないが、あえて革新的な概念と思えるのは、ソフトウェアの外部特性としての視点（フィーチャ指向）、ソフトウェアの部品化としての視点（コンポーネント指向）、およびソフトウェア制御の反転の視点（Dependency Injection：依存性の注入）である。これら 3 つの概念は、ソフトウェアの依存性に関する課題を解決する上で重要ないくつかの指針を与えてくれているからである。

フィーチャ指向は、ソフトウェアの再利用単位を従来のモジュールやクラスのレベル（ライブラリなど）から、より大きなレベル（コンポーネント、コンテナ、フレームワークなど）に引き上げる概念を提供している。また、コンポーネント指向は、ソフトウェアの分割統治や可変性に関する実装概念を提供している。さらに依存性の注入は、ソフトウェア部品の依存関係を間接的な関係に置換える概念を提供している。これらの概念が SPL 開発を現実的な開発方式として扱えるようにしている技術的な基盤であると考えている。

(3) SPL 開発の開発プロセスと全体イメージ

SPL 開発の標準開発プロセスには、事前に製品系列の共通基盤や部品群となるソフトウェア、その他の再利用可能な資産（これらを総称して「コア資産」と呼ぶ）を開発するドメイン・エンジニアリングプロセスと、コア資産を再利用して製品系列における個々の製品となるソフトウェアを開発するアプリケーション・エンジニアリングプロセスの 2 つの開発プロセスがある。図 92-3 に SPL 開発の標準開発プロセスの概要を示す。

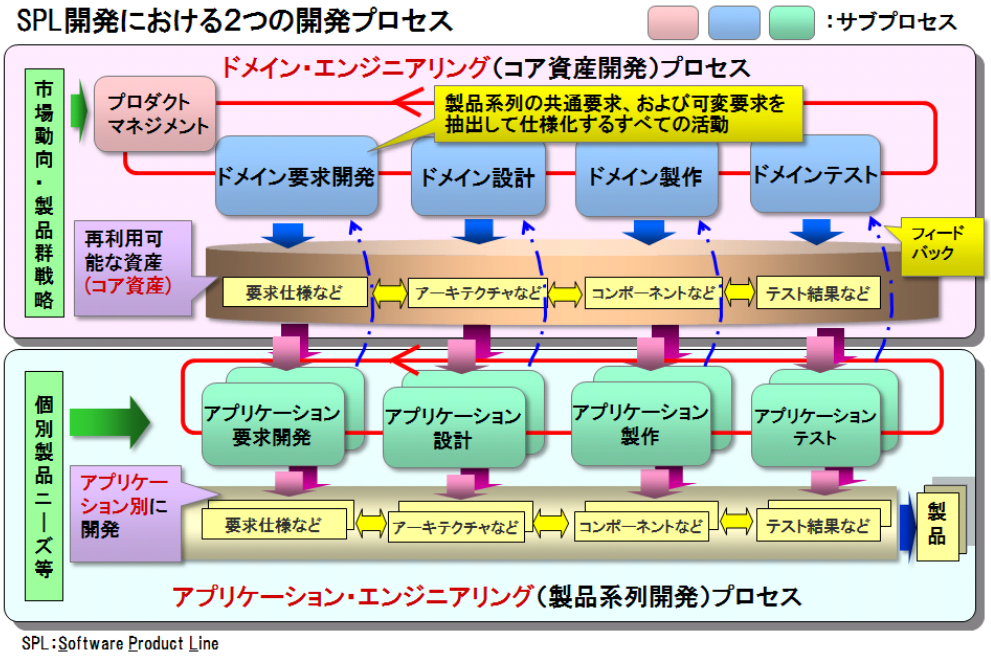


図 92-3 SPL 開発の標準開発プロセス

SPL 開発のドメイン・エンジニアリングとアプリケーション・エンジニアリングの 2 つの開発プロセスにより製品が開発されるまでの流れについて、その全体イメージを図 92-4 に示す。

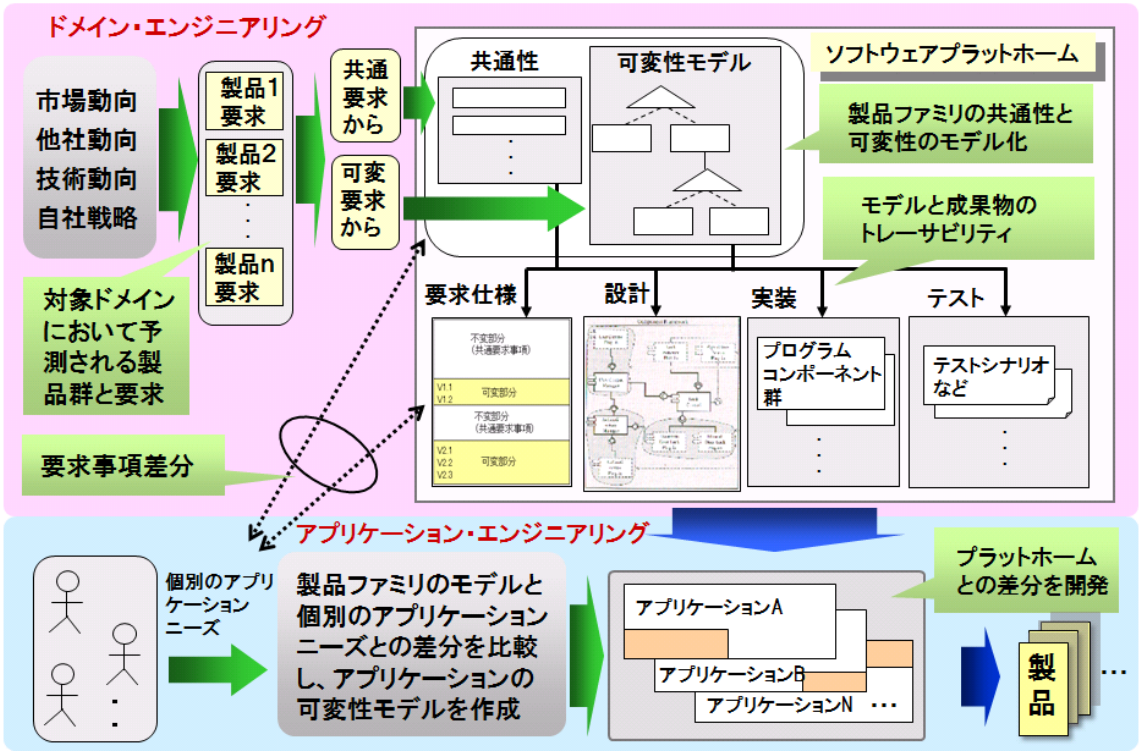


図 92-4 SPL 開発の全体イメージ

(4) SPL 開発に期待できる効果

SPL 開発の導入により期待できる効果として、大きくは 2 つあると言われている。1 つは、開発コストの大幅な低減である。SPL 開発では、ドメイン・エンジニアリングプロセスにおけるコア資産開発への先行投資が発生するが、その後のアプリケーション・エンジニアリングプロセスにおいては、コア資産を再利用した低コストの製品系列開発が可能となり、単一システムの開発コストと比較した場合、理論的には約 3 システム目で損益分岐点に達すると言われている。もう 1 つは、市場投入までの時間（Time to Market）の大幅な短縮である。開発当初は、ドメイン・エンジニアリングプロセスによる再利用可能なコア資産の開発を行うため、製品の市場投入はできないが、一旦コア資産の開発が終了すれば、製品系列の開発はコア資産を再利用した製品系列の同時並行開発が可能となり、1 製品当たりの開発期間は大幅に短縮することができると言われている。図 92-5 は、SPL 開発に期待できる 2 つの効果について説明している。

上記以外の効果として、品質の面でもコア資産を再利用した製品開発を繰返すことにより、大幅な品質向上が期待できる。

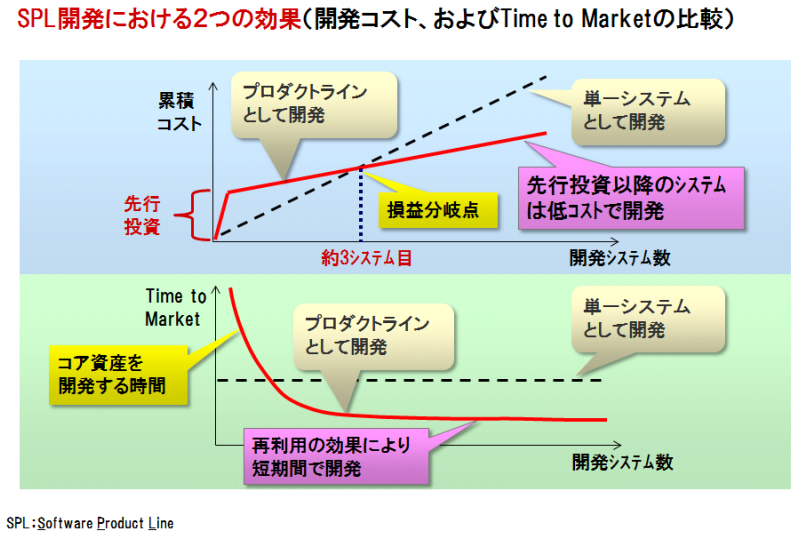


図 92-5 SPL 開発に期待できる効果

1.2. モデル指向 SPL 開発の特徴

モデル指向 SPL 開発とは、ソフトウェア開発の現場において、後述の SPL 開発導入に関する問題点を解決する目的で、SPL 開発とモデル指向開発を融合させ考案した製品系列の開発手法である。モデル指向 SPL 開発の大きな特徴は、開発の超上流である製品系列の分析から製品仕様の分析、製品系列アーキテクチャの設計、および製品ソフトウェア設計までのアクティビティにおいて以下のような要素技法を有することである。

(1) 製品戦略を具現化する製品フィーチャ分析技法

製品ソフトウェアのプロダクトライン開発において最も重要なアクティビティは、製品系列

開発の過去（あれば既存製品）と未来（今後の製品戦略）にわたり、製品系列全体を対象として製品の持つ特徴、または特性（フィーチャ）となる製品機能群を、視点を決めて分析すること（この分析を「製品フィーチャ分析」と呼ぶ）である。ここでの視点とは、製品機能における共通性と可変性の視点である。SPL 開発の重要なポイントは、「共通化可能な部品がどれだけあって、選択による交換可能な部品がどれだけあるか」という指針を持つことである。製品フィーチャ分析の分析結果によって製品開発コストの長期的な見積が可能となる。ただし、共通性と可変性には、製品フィーチャ分析の段階で製品機能としてユーザから見える外部的なものと、ソフトウェアの分析、設計の段階でソフトウェア構造として開発者から見える内部的なものがある。すなわち、開発の前に行う開発コスト見積では、外部的な共通性と可変性だけしか見えないことになる。

図 92-6 は製品の可変性について説明している。可変性は可変点と可変部という概念を持ち、外部と内部、時間と空間といった視点を持っている。これらの概念と視点を念頭におき、製品の可変性について体系的に分析することが必要となる。

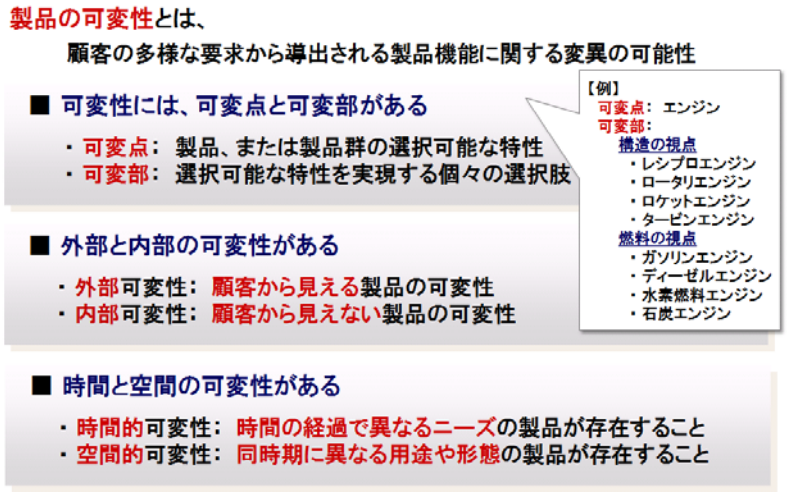


図 92-6 製品の可変性

製品フィーチャ分析では、フィーチャマップ、フィーチャマトリクスを使用したフィーチャモデリングを実施し、製品系列全体の製品機能におけるコア資産化率（＝コア資産化可能な製品機能数÷全製品機能数×100）を算出することで、モデル指向 SPL 開発の適用可否判断を行うことができる。

(2) 製品系列の基盤となるアーキテクチャ設計技法

SPL 開発には、コア資産を作り出すためのドメイン・エンジニアリングプロセスがある。ここでのコア資産の定義は、製品系列開発の中で共通的、または選択的に再利用可能なソフトウェア資産のことである。本プロセスにおける重要なアクティビティとして、製品系列の共通的な基盤となるソフトウェアアーキテクチャを分析、設計して構築することが挙げられる。すなわち、製品フィーチャ分析で得られた共通性と可変性の分析結果から、さらに製品の仕様分析を行い、製品系列のコア資産となる共通基盤やソフトウェア部品（一般に「ソフトウェアコン

ポーネント」と呼ばれている）の構成を設計し、製品のソフトウェアアーキテクチャを確定させることである。製品の仕様分析にモデル指向開発手法における要件分析技法*を適用することで、製品としてのソフトウェアの構造と振る舞いを論理的に決めていくことができる。

＊注記：モデル指向開発手法の要件分析技法についての詳細は、別稿の『統合モデル技術によるユーザ主体のソフトウェア開発手法「モデル指向開発（MOD)」の紹介』を参照のこと。

図 92-7 にソフトウェアアーキテクチャ設計における製品ソフトウェアの代表的な構成例を示す。

代表的な構成例としては、コア資産コンポーネントを組合せたコンポーネント構成とフレームワークを中心としたフレームワーク構成がある。

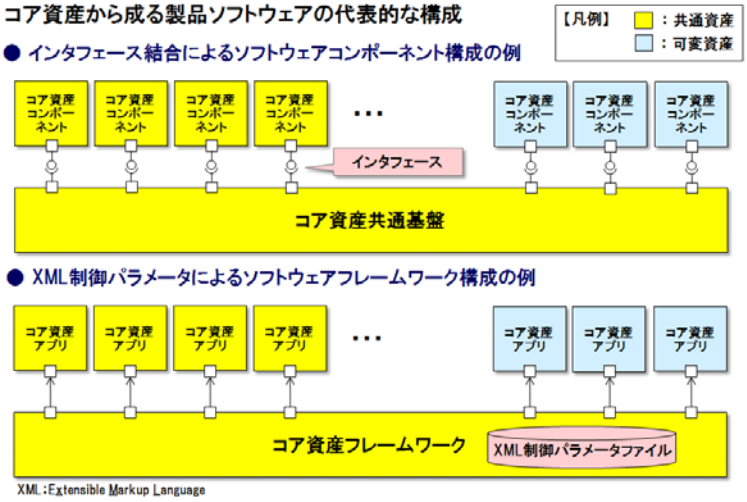


図 92-7 モデル指向 SPL 開発におけるコア資産構成例

(3) 製品機能を実現するフィーチャコンポーネント設計技法

製品フィーチャ分析にて明確にした製品機能群の共通性や可変性から、製品系列全体を通して共通化する機能、ある製品群の範囲内で共通化する機能、および製品間で選択可能な必須機能やオプション機能などが導出できる。

ドメイン・エンジニアリングプロセスにおけるアクティビティには、製品フィーチャ分析で明確にした製品機能群について、製品系列開発の中で再利用可能なソフトウェアコンポーネント（これを「コア資産コンポーネント」と呼ぶ）として分析、設計していくことがある。コア資産コンポーネントは、共通的に再利用可能なものと選択して再利用可能なものに分類することができる。さらにコア資産コンポーネントを製品機能単位にカプセル化されたもの（これを特に「フィーチャコンポーネント」と呼ぶことにする）と後述する製品機能と対応しないソフトウェアの内部構造単位にカプセル化されたもの（これを「インナコンポーネント」と呼ぶ）に分類する。そこで共通性、可変性が明確になった製品機能について独立した単位となる粒度で切り出し、ソフトウェアコンポーネントに割当て、その構造と振る舞いを設計していく。切り出された製品機能の仕様からソフトウェアの構造と振る舞いを論理的に導出していくため

の有効な方法として、モデル指向開発手法の要件分析技法*を適用する。本技法の適用により、ソフトウェアコンポーネントのインタフェースと内部構造を論理的に導出し、フィーチャコンポーネントとして設計していくことができる。図 92-8 にフィーチャコンポーネントのイメージを示す。

＊注記：モデル指向開発手法の要件分析技法についての詳細は、別稿の『統合モデル技術によるユーザ主体のソフトウェア開発手法「モデル指向開発（MOD)」の紹介』を参照のこと。

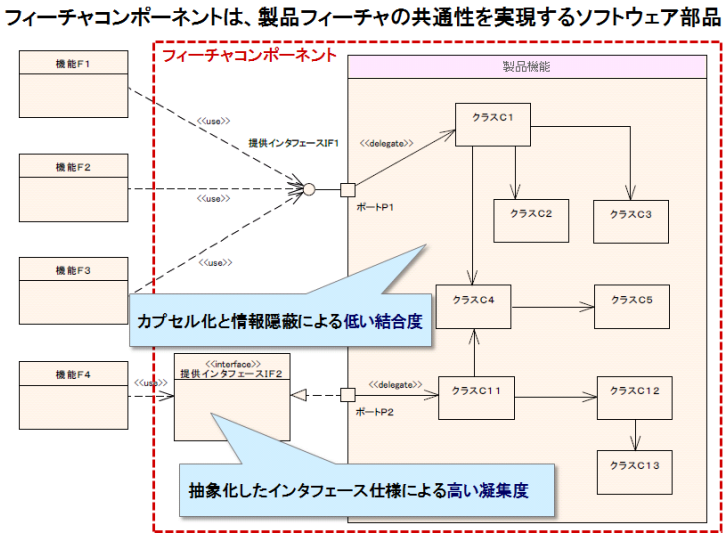


図 92-8 フィーチャコンポーネントのイメージ

(4) 設計仕様によるインナコンポーネント設計技法

製品の可変性には、顧客から見える外部的な可変性（外部可変性）と顧客から見えない内部的な可変性（内部可変性）がある。また、製品の共通性についても同様なことが言える。製品フィーチャ分析から得られる外部的な製品の共通性や可変性からフィーチャコンポーネントが導出できるが、この時点では内部的な共通性や可変性については見えていない。内部的な共通性や可変性が見えてくるのは、製品機能の仕様分析後であり、ソフトウェアアーキテクチャ設計以降になる。内部的な共通性や可変性が見えてきた段階で、ソフトウェアの構造と振る舞いについて結合度と凝集度を考慮しながら再利用性や交換可能性の視点で切り出し、カプセル化することになる。

外部的な共通性や可変性に対応させるフィーチャコンポーネントの場合は、コンポーネントの単位が決まっているため、インタフェース仕様が設計しやすいが、内部的な共通性や可変性に対応させるインナコンポーネントの場合は、全体構造から切り出す時点で内部構造と共にインタフェース仕様を設計する必要がある。インナコンポーネントの単位を切り出す際の考慮点は、できる限り内部の凝集度を高くし、外部との結合度を低くすることである。また、インタフェース仕様を設計する際の留意点は、インナコンポーネントが責務を果たす上で必要となる条件と情報、および責務として提供すべき情報について、その振る舞いをモデル化して確定させることである。図 92-9 にインナコンポーネントを切り出すイメージを示す。

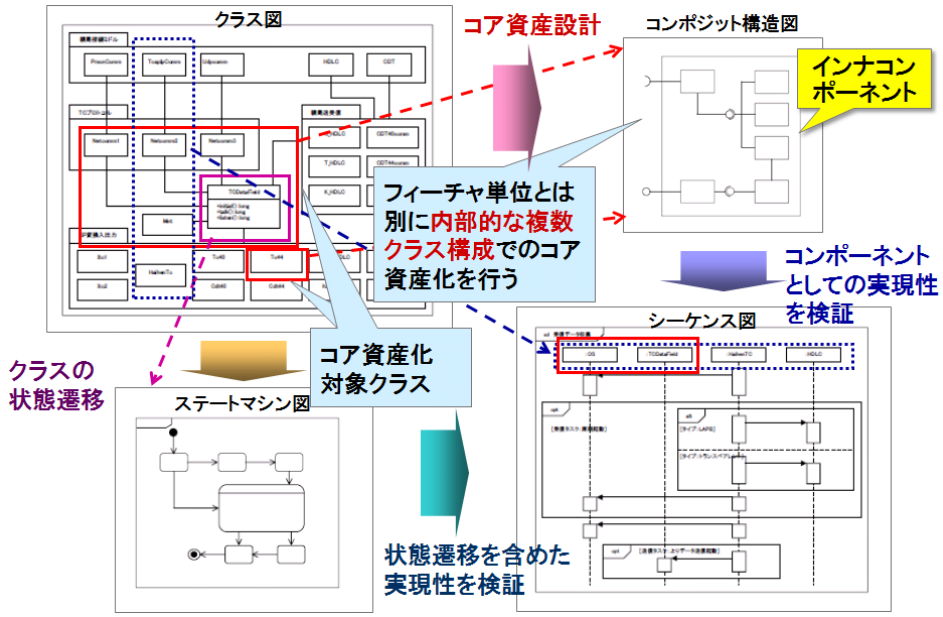


図 92-9 内部構造から切り出すインナコンポーネント

(5) コア資産を評価・管理する構成管理技法

SPL 開発において管理すべきソフトウェア資産としては、製品系列開発で繰返して再利用されていくコア資産と製品系列開発の中で個々に開発される製品固有のアプリケーション資産（略して、アプリ資産と呼ぶ）に大別できる。アプリ資産は個々の製品系列開発の中で管理すればよいが、コア資産は製品系列全体にわたるソフトウェア資産であるため、個々の製品系列開発とは別の独立させた管理プロセス（これを「コア資産管理プロセス」と呼ぶことにする）が必要であると考えている。

コア資産には、製品系列全体で共通的に再利用されるもの（「共通コア資産」と呼ぶ）と製品系列の製品に応じて選択して再利用されるもの（「可変コア資産」と呼ぶ）の 2 種類がある。

また、コア資産のバージョン変更は基本的になく、構成管理対象となるコア資産には以下のものがある。

- 機能仕様書
- 設計仕様書
- テスト仕様書
- ソース形式プログラム
- バイナリ形式プログラム（製品系列開発への提供用）
- コンポーネント利用ガイド（インタフェース、適用条件など）

2. 適用手法の目的

モデル指向 SPL 開発手法を考案した主な目的としては、以下に述べるような開発現場における製品系列ソフトウェア開発の問題点を解決することであった。

2.1. 開発現場の問題点と対応策としての課題

製品系列を持つソフトウェア開発の現場では、単一製品のソフトウェア開発と同様の派生開発を行っている。すなわち、製品系列における新製品や機能の追加、変更の開発は、旧製品をベースとした改造として開発を進めているのが現状である。このような派生開発の形態で進めているため、品質、効率、および構成管理の面、すべてにおいてさまざまな問題が発生していた。例えば、ソフトウェアアーキテクチャの劣化、ソフトウェアの不必要な肥大化、類似ソフトウェアの無計画な増大などを原因とする品質、効率の低下や構成管理の破綻が挙げられる。

また、当社では派生開発が全体の 80%以上を占めていることから、派生開発で発生しているさまざまな問題点の比重が大きくなっている。その中に製品系列のソフトウェア開発も含まれており、製品系列という特性が無視され、単一製品の派生開発と同様に扱われている場合が多い。そこで、品質や効率、および構成管理の面で改善することを目的に SPL 開発を導入したいと考える管理者も出てきていた。

このような状況から製品系列を持つソフトウェアの派生開発に SPL 開発を導入し、品質、効率、および構成管理の問題を解消し、さらに SPL 開発の効果である開発コストの大幅な低減を目指すことを考えた。しかしながら、SPL 開発の導入に関しては表 92-1 のような問題点があり、ソフトウェア開発の現場に適用することが極めて困難になっていた。

表 92-1 SPL 開発導入の問題点

No.	問題点
1	SPL 開発は概念的に理解しやすい反面、進め方の具体的な手順が示されていない
2	製品機能の可変性をソフトウェア構造につなげる具体的方法が確立されていない
3	製品系列開発を繰返していく過程で再利用に関する評価、管理の方法が不明である

以上の問題点について解決策を考える上で、これらの問題点がどのような関係になっているのかを調査し、SPL 開発とモデル指向開発手法の融合について、SPL の考え方にモデル指向開発の各種技法をどのように適用するかを具体的に決めていくことにした。そして解決策としての方法論を確立させ、SPL 開発をソフトウェア開発の現場に導入できるようにすることを考えた。そのための解決すべき課題として表 92-2 の項目を設定した。

表 92-2 SPL 開発導入の課題

No.	課　題
1	SPL 開発を導入して製品開発を進めるための具体的な開発手順を示す
2	製品機能の可変性をソフトウェア構造につなげる具体的方法を確立させる
3	製品系列開発を繰返していく過程で再利用に関する評価、管理の方法を示す

上述の課題を解決できれば、ソフトウェア開発の現場における SPL 開発導入の問題点解決に対応でき、品質、効率、および構成管理の問題を解消することができると考え、そのための手段としてモデル指向 SPL 開発手法を考案するに至った。

2.2. 課題解決のための目標と着眼点

なぜモデル指向 SPL 開発が課題解決の手段として有効なのか、その理由をここで明らかにしていきたい。そのため、モデル指向 SPL 開発手法の適用が、設定した課題を解決していくための目標、および着眼点について述べる。

(1) **課題 1**：SPL 開発を導入して製品開発を進めるための具体的な開発手順を示す

［**目標**］SPL 開発におけるコア資産開発のためのドメイン・エンジニアリングと製品系列開発のためのアプリケーション・エンジニアリングについて、書籍の一般的な概念や断片的な活動の解説だけでは、製品系列を持つソフトウェア開発の現場で手順を具体化して開発業務につなげることが極めて困難である。開発現場に適した具体的な手順としての開発プロセスが必要になる。SPL 開発の考え方は概念的に理解しやすく、大きな効果を期待できそうであることも直感的に理解しやすい。しかし、ソフトウェア開発の現場では、組織的な活動として、どこから、どのように手を着けていけば無理なく導入が可能となるのかが分からないでいる。また、規模が大きな開発方式なだけに、その分導入のリスクも大きくなることが懸念される。そのため、開発手順に関する障壁をすべて解決する具体的な開発プロセスを示す。

［**着眼点**］一般に SPL 開発は製品戦略ロードマップを入力として開始される。SPL 開発におけるコア資産開発のドメイン・エンジニアリングでは、製品系列のドメインを対象としてドメイン要求開発、ドメイン設計、ドメイン製作、ドメインテストといったサブプロセスで進められていく。また、SPL 開発における製品系列開発のアプリケーション・エンジニアリングでは、製品系列の個々の製品を対象としてアプリケーション要求開発、アプリケーション設計、アプリケーション製作、アプリケーションテストといったサブプロセスで進められていく。これらの各サブプロセスにおいて、具体的な開発のアクティビティの流れと各アクティビティで作成される具体的な成果物を定義することで、ソフトウェア開発の現場に導入できるようにすることを考えた。そして、これらの具体的なアクティビティと作成する成果物にモデリング技術を適用することにした。

図 92-10 にドメイン・エンジニアリングに対応したモデル指向 SPL 開発のコア資産開発プロセス概要、図 92-11 にアプリケーション・エンジニアリングに対応したモデル指向 SPL 開発の製品系列開発プロセス概要を示す。モデル指向 SPL 開発では、両プロセスにおいて製品系列全体を対象にしたプロセスか、固有の各製品を対象にしたプロセスかの違いだけで、各アクティビティで適用するモデリングメソッドは同一であり、ソフトウェア開発の現場へ導入しやすくしている。

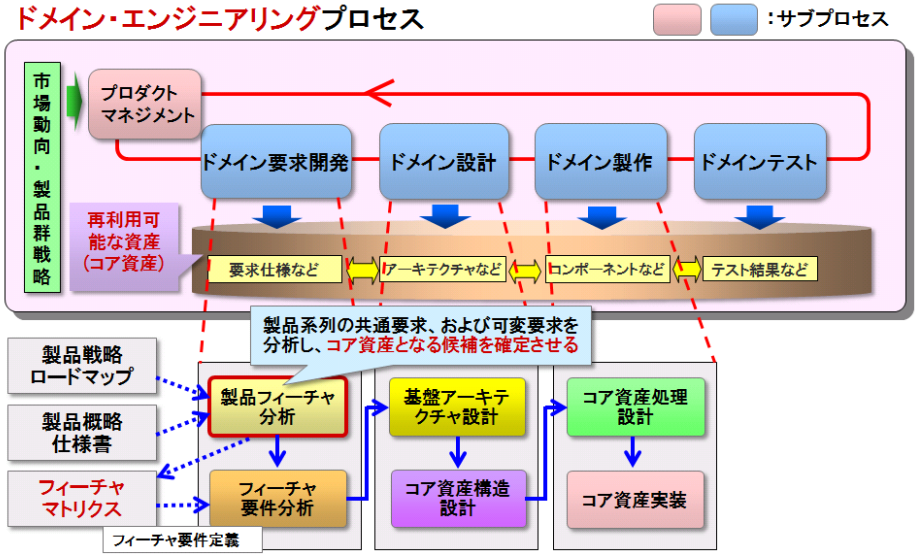


図 92-10 モデル指向 SPL 開発のコア資産開発プロセス概要

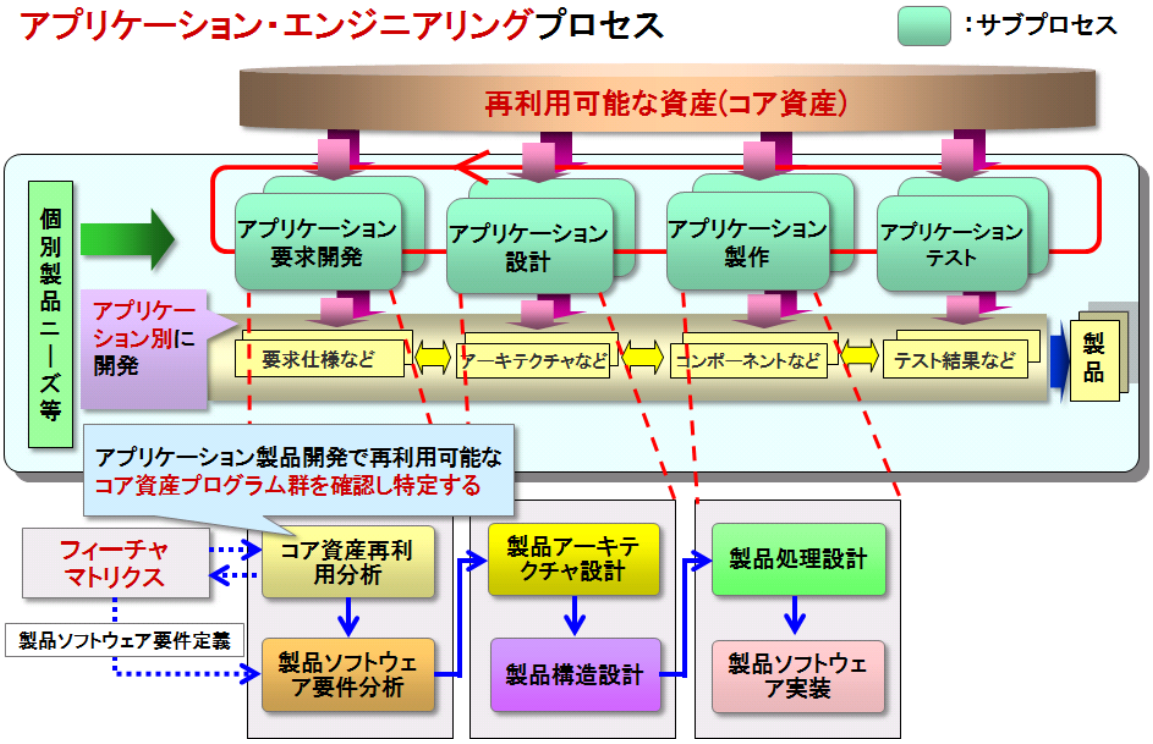


図 92-11 モデル指向 SPL 開発の製品系列開発プロセス概要

図 92-12 にモデル指向 SPL 開発の各サブプロセスにおける成果物としてのモデルを示す。ここで、コア資産開発と製品系列開発の両プロセスで作成されるモデルは同一種類であることから、両プロセスの各サブプロセスにおける本質的なアクティビティは同一であると言える。

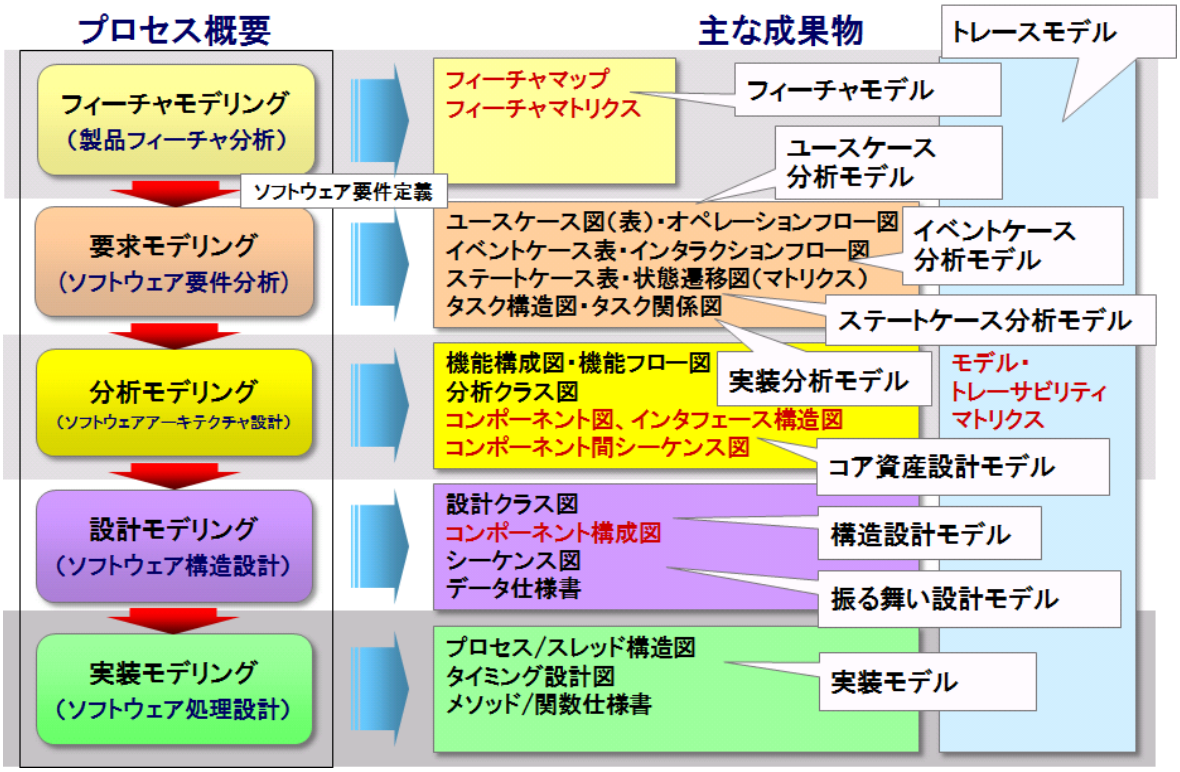


図 92-12 モデル指向 SPL 開発のプロセスと成果物としてのモデル

図 92-12 のプロセスにおける各サブプロセス(またはフェーズとも言う)の名称は、ドメイン・エンジニアリングプロセスとアプリケーション・エンジニアリングプロセスの両プロセスにおける各サブプロセスの名称を本質的なアクティビティの観点から共通化した名称としている。

(2) **課題 2**：製品機能の可変性をソフトウェア構造につなげる具体的方法を確立させる

［目標］製品機能の可変性は、機能の選択可能性につながり、現在把握できている機能のバリエーションだけでなく、将来の機能強化や機能拡張への可能性を広げることのできる性質である。この性質をソフトウェア構造に置換える上で最も適している要素がソフトウェアコンポーネントである。ソフトウェアコンポーネントは、抽象化された汎用的なインタフェースを持ち、粒度の大きな責務でカプセル化、および情報隠蔽された独立性の高いソフトウェア構造である。モデル指向開発手法を適用することによって、製品機能の可変性からソフトウェアコンポーネント構造へ論理的につなげる方法を確立させる。

［着眼点］製品機能の可変性からソフトウェアコンポーネント構造までは、かなりの隔たりがあり、その間をつなげる何らかの具体的な分析手順が必要になる。そこで製品機能の可変性を分析する方法から考えなければならない。製品系列における製品機能の可変性を分析する以前に、製品戦略ロードマップから今後市場にリリースしていく製品に織り込む固有な機能を明確にすることが必要になる。その上で製品系列全体の製品機能について、共通性と共に可変性を分析し、分析結果をモデル化していく（これを「フィーチャモデリング」と呼ぶ）。モデル化するには、製品機能全体が俯瞰（ふかん）できて、かつ各製品機能の可変点と可変部が見えるような分析モデルが最適である。そのモデルで明確になった可変点と可変部からソフトウェアコンポーネント

の単位とインタフェース仕様を導出し、製品機能の要求仕様を分析することで、論理的にソフトウェアコンポーネントの構造と振る舞いを確定させていく。

図 13 にモデル指向開発手法におけるソフトウェアコンポーネントの定義を示す。

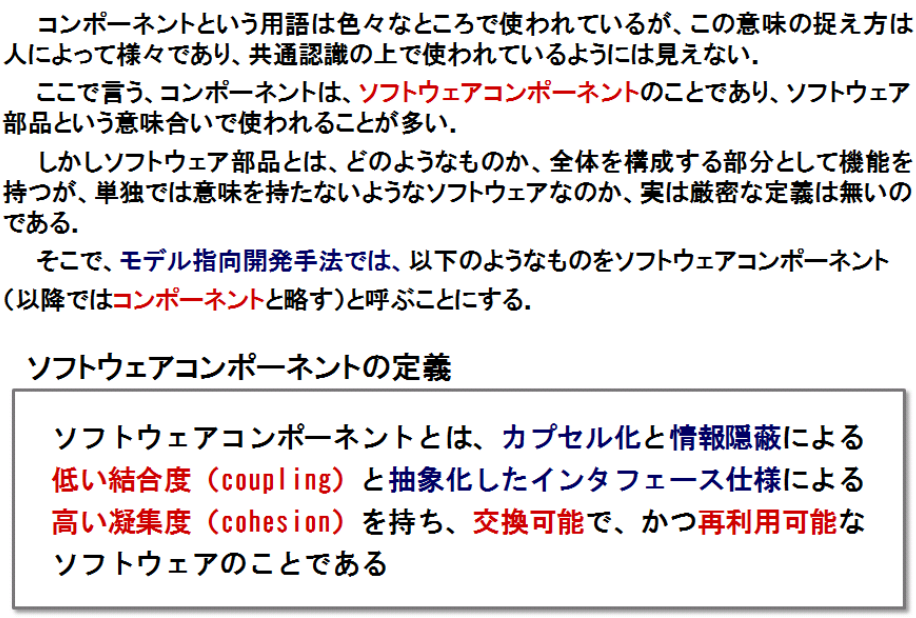


図 92-13 モデル指向開発におけるコンポーネントの定義

コア資産は SPL 開発における再利用可能なソフトウェア資産であり、製品フィーチャ分析から導出される製品系列の共通基盤や製品機能単位のフィーチャコンポーネント、およびソフトウェアの設計段階で導出される内部的なインナコンポーネントが該当する。図 92-14 にモデル指向 SPL 開発におけるコア資産の要件に関する定義を示す。

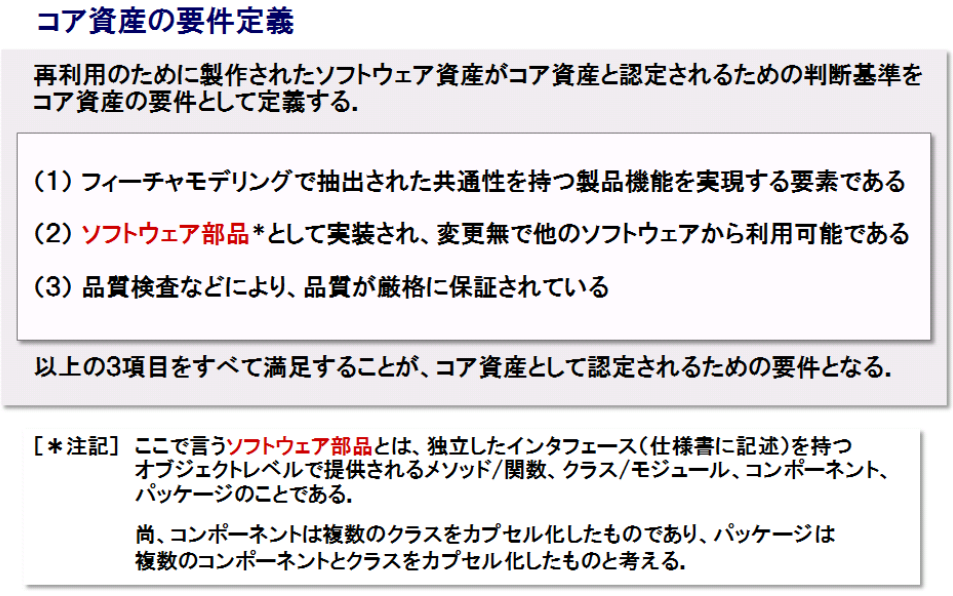


図 92-14 モデル指向 SPL 開発におけるコア資産の要件

(3) **課題 3**：製品系列開発を繰返していく過程で再利用に関する評価、管理の方法を示す

［**目標**］SPL 開発では、コア資産開発のドメイン・エンジニアリングと製品系列開発のアプリケーション・エンジニアリングのプロセスが提案されているが、開発したコア資産について再利用と構成管理の視点で評価、管理するプロセスが示されていない。ソフトウェア開発の現場でコア資産開発と製品系列開発を繰返し継続していく中で、開発プロセスとは別に必要となる管理プロセスである。本プロセスにおける具体的な手順と方法を示すことで、ソフトウェア開発の現場へ SPL 開発を導入できるようにする。

［**着眼点**］コア資産は最初の製品系列開発が開始される前に事前に開発されるのが一般的であるが、実際の開発においては、製品系列開発を繰返す過程で必要となるコア資産やコア資産の機能拡張などが発生する。そこで、製品系列開発が開始された後にコア資産の再利用性評価や追加開発、および機能拡張などに対応するプロセスが必要となる。このプロセスでは製品系列の個々の製品とは異なるコア資産のみを対象とするため、製品系列開発のプロセスとは別な独立したプロセスにすべきと考えた。これを「コア資産管理プロセス」と呼び、プロセスに必要なアクティビティとそれらの手順や方法を整理してまとめ、ガイドすることにした。

3. 適用手法の対象と方法論

3.1. 適用対象の製品・プロジェクト

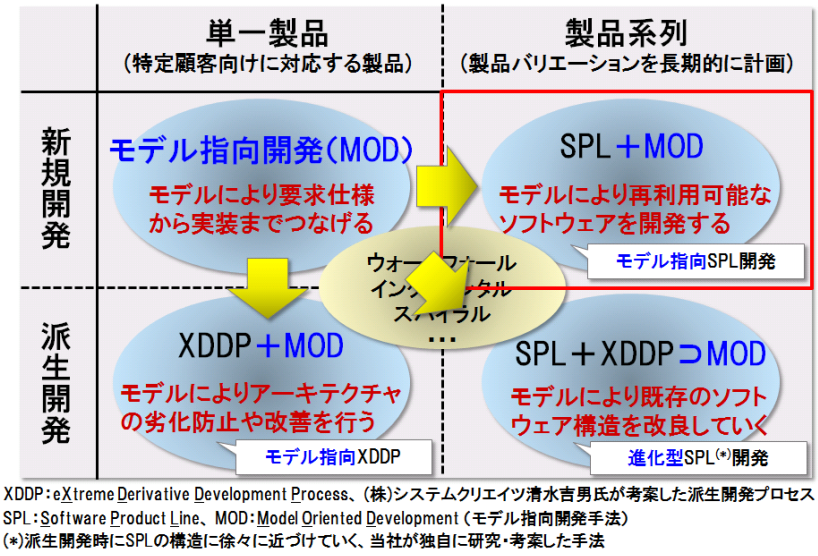
モデル指向開発手法の適用対象の製品として単一製品と製品系列を持つ製品に分類し、適用対象のプロジェクトとして新規開発と派生開発に分類すると、図 92-15 のように 4 象限の適用対象となる。モデル指向開発の適用対象として最初のターゲットは、単一製品の新規開発であり、ここでモデル指向開発の基盤技術が確立された。次の適用対象としてターゲットになったのが、今回紹介しているモデル指向 SPL 開発手法である。

その後、最も難易度の高い適用対象である製品系列の派生開発にチャレンジして、その方法論を完成させた。これは、モデル指向 SPL 開発に派生開発プロセス手法 XDDP（eXtreme Derivative Development Process）を融合させ、派生開発の中で再利用可能なコア資産を段階的に開発していく SPL 開発手法（これを「進化型 SPL 開発」と呼ぶ）である。

適用対象として最後のターゲットにしたのが単一製品の派生開発であった。これは、派生開発プロセス手法 XDDP にモデル指向開発を融合させた開発手法（これを「モデル指向 XDDP」と呼ぶ）である。

本稿では、その中のモデル指向 SPL 開発について、その方法論を解説している。

「モデル指向 XDDP」、「進化型 SPL 開発」の方法論については、別な機会の紙面で解説することにしたい。



XDDP：eXtreme Derivative Development Process、（株）システムクリエイツ清水吉男氏が考案した派生開発プロセス
SPL：Software Product Line、MOD：Model Oriented Development（モデル指向開発手法）
(*)派生開発時にSPLの構造に徐々に近づけていく、当社が独自に研究・考案した手法

図 92-15 モデル指向 SPL 開発の適用対象開発

モデル指向 SPL 開発の適用実績としては、制御系システム、組込み系システム、および情報系システムの各分野にわたっている。適用実績のある分野について、図 92-16 に示す。

■ 製品系列の新規・リエンジニアリング開発に適用

- 制御系システム
  - 社会システム
  - 公共システム
  - 交通電力システム
  - 交通制御システム
  - 電力システム
  - 産業制御システム
- 組込み系システム
  - 伝送システム
  - 車載システム
- 情報系システム
  - 産業情報システム
  - パッケージソフト

■ 製品系列の派生開発に適用

- 派生開発プロセス手法XDDPとの融合
  進化型SPL開発手法として社内派生開発へ適用

XDDP: eXtreme Derivative Development Process

図 92-16 モデル指向 SPL 開発の適用対象分野

3.2. モデル指向 SPL 開発の方法論

ソフトウェア開発方法論の立場から、モデル指向 SPL 開発において確立させた点を中心として、開発プロセスのフェーズ別に開発現場における有効性の根拠について述べる（モデル指向 SPL 開発のコア資産開発プロセスと製品系列開発プロセスの詳細については、図 92-21、図 92-22 の各プロセスフローを参照のこと、ただし、成果物としてのモデルに省略有り）。

(1) コア資産開発プロセス

本プロセスは、SPL 開発のドメイン・エンジニアリングプロセスについて、モデル指向開発手法のモデリングフェーズを融合させ、具体的なアクティビティで構成した製品系列全体の再利用可能なコア資産を構築するための開発プロセスである。

図 92-17 は、コア資産開発プロセスの各アクティビティについて、その概要を説明している。

図 92-17 コア資産開発プロセスのアクティビティ

（a） 製品フィーチャ分析（フィーチャモデリング）

戦略的な視点で組織的に作成された開発製品群全体における過去から、現在、将来にわたる製品戦略ロードマップが製品フィーチャ分析の基になる。分析は、製品戦略ロードマップからブレークダウンして各製品の特徴を明確にした「製品概略仕様書」を作成するアクティビティから始まる。次に製品系列の製品群全体について、フィーチャモデリングにより製品機能の共通性と可変性を分析し、フィーチャモデルとして「フィーチャマップ」（マインドマップ表記技法をフィーチャモデリングに応用したモデル指向 SPL 開発特有の成果物）を作成する。フィーチャマップを考案した理由は以下の通りである。

通常、フィーチャモデルは、フィーチャ図や可変性図で表記されることが多い。しかし、これらの図で製品系列全体のフィーチャを表現する方法は、開発現場において現実的ではないと判断している。製品系列全体のフィーチャは大規模、かつ複雑であり、部分的な表記にはよいが、全体を表現するには不向きと考えているからである。

フィーチャマップでは、各製品機能について共通性、および可変性（可変点）の分類として「必須」（共通）、「任意選択」（オプション）、「複数選択」（最低 1 つ選択）、「排他選択」（択一的選択）のいずれかを付す。製品機能における可変性の分析で重要な点は、可変点が存在する粒度（言い換えると可変点が無くなる）まで機能（可変部）を分解していくことである。また、製品機能間に依存関係がある場合は、該当する機能（可変部）に依存関係の属性を明記しておく。

フィーチャマップのレビュー完了後、フィーチャマップの情報を階層化した製品機能と

して縦軸にし、製品概略仕様書の情報を横軸にして「フィーチャマトリクス」を作成する。本マトリクスによって製品系列全体におけるコア資産化可能な製品機能の割合や各製品機能が持つバリエーションなど、SPL 開発の指針となる重要な情報が可視化できる。製品フィーチャ分析の結果を以ってモデル指向 SPL 開発の最終的な適用判断を行うことになる。図 92-18 は、製品戦略ロードマップからフィーチャマトリクス作成までの製品フィーチャ分析の流れを説明している。

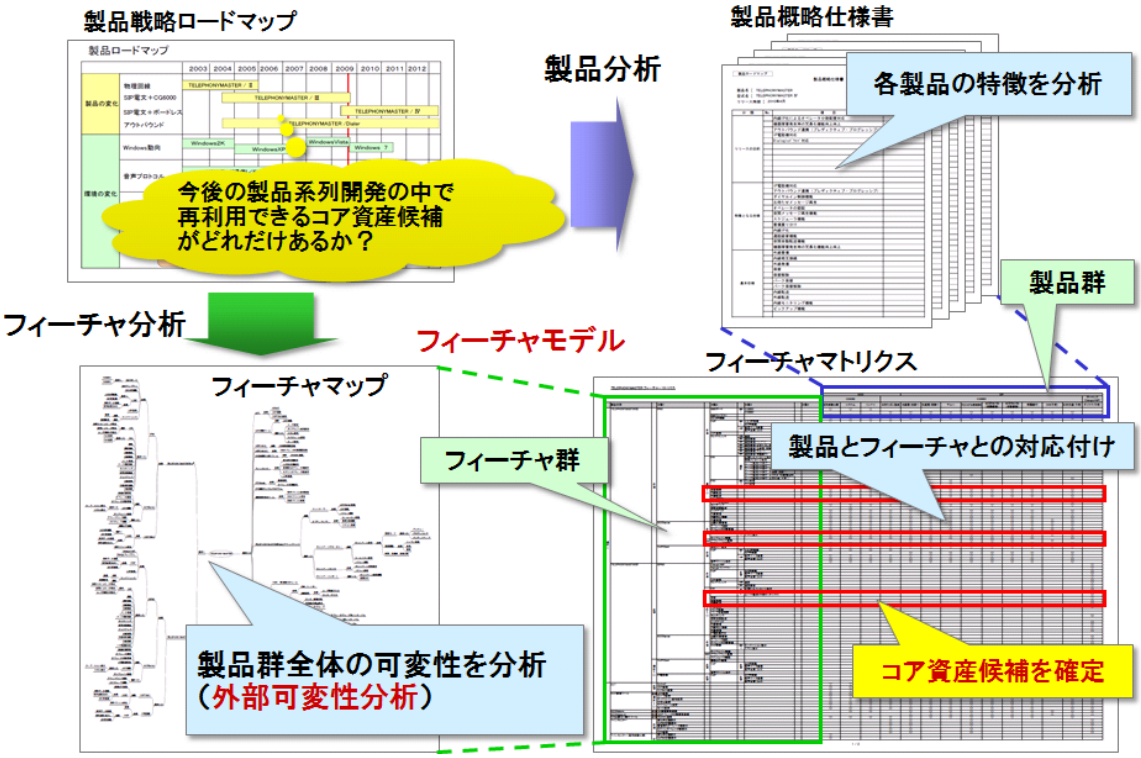


図 92-18 製品フィーチャ分析の概要

ここで、フィーチャ要件定義はモデリングの対象にしていないので説明を省略する。

(b) フィーチャ要件分析（要求モデリング）

フィーチャ要件定義にて定義した製品系列の要求仕様、および外部インタフェース仕様を基にして、製品フィーチャ分析で導出された製品系列全体で必須となる共通機能と選択可能な共通機能について要件分析を行う。要件分析では、モデル指向開発手法の要件分析技法*を適用して以下の 3 種類の視点による分析を必要に応じ組合せて実施する。

① ユースケース分析

ユーザインタフェースによるソフトウェアの使い方の視点で要件を分析する。

外部インタフェース仕様に画面が含まれる場合は、並行して画面仕様設計を行う。

- ユースケース：ソフトウェアの使い方のパターン
- シナリオフロー：ユースケースに対応した詳細な振る舞い

（対象に応じてオペレーションフロー、処理フロー、またはデータフローを作成）

② ステートケース分析

ソフトウェアの外部や内部に対する動作条件の視点で要件を分析する。

・ ステートケース：振る舞いの前提となる状態のパターン

・ ステートマシン：トリガによる状態遷移と状態における振る舞い

（状態遷移マトリクス：ステートマシンの抜け漏れチェック用に作成）

③ イベントケース分析

ソフトウェアの外部や内部に対する相互作用の視点で要件を分析する。

・ イベントケース：イベントを媒体とした相互作用のパターン

・ インタラクションフロー：入力イベントに対応した詳細な振る舞い

＊注記：モデル指向開発手法の要件分析技法についての詳細は、別稿の『統合モデル技術によるユーザ主体のソフトウェア開発手法「モデル指向開発（MOD)」の紹介』を参照のこと。

図 92-19 に 3 種類の視点による分析を組合せた要件分析のプロセス例を示す。

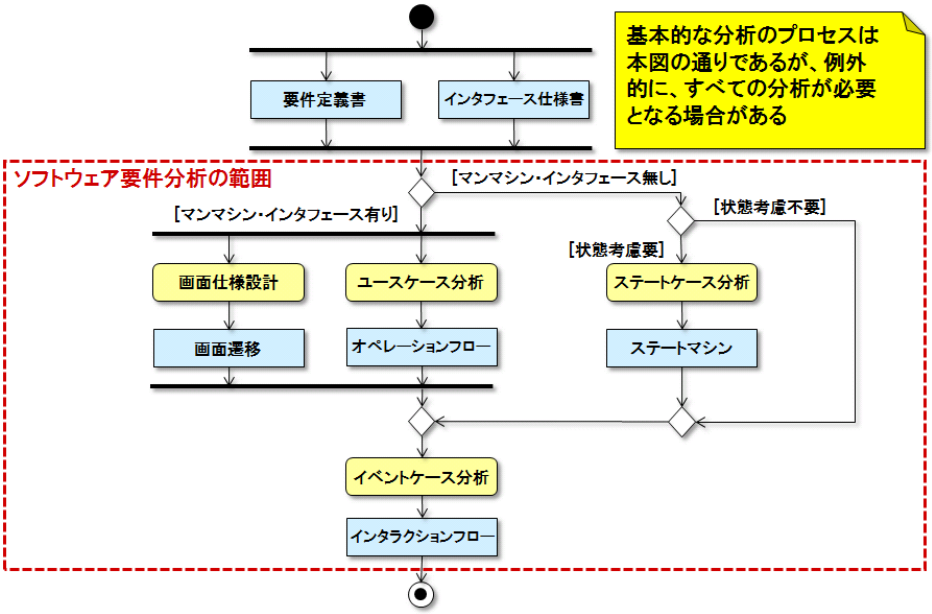


図 92-19 要件分析のプロセス例

(c) 基盤アーキテクチャ設計（分析モデリング）

要件分析の結果から、責務の視点で製品機能全体を大きな責務（コンポーネントの単位)、製品機能内に存在するデータと処理に関する役割単位を小さな責務（クラス）として捉え、それらの構造を抽出してコンポーネント、クラスに割当てながら、全体の論理的なアーキテクチャを決定していく。また、振る舞いの順序性、および並列性の視点で実行環境としてのタスクの構成と構造を抽出して、プロセス、スレッドに割当てるための実装レベルのアーキテクチャも決定していく。

(d) コア資産構造設計（設計モデリング）

コア資産として定義されたフィーチャコンポーネントの内部構造とインタフェース仕様を設計する。インナコンポーネントの切り出しと設計もこの段階で行う。内部構造の設計

内容は設計クラス図で表記し、インタフェース仕様の設計内容はインタフェース構造図で表記する。また、内部構造として持つ永続データの論理構造も設計する。最後にフィーチャコンポーネントのインタフェース仕様に応じたシーケンス図を作成し、構造設計の妥当性を検証する。

(e)　コア資産処理設計（実装モデリング）

製品基盤としての実行環境となるプロセス、またはスレッドの設計やコア資産としてのフィーチャコンポーネントを構成する各クラスの処理メソッドである関数仕様を設計する。また、コンポーネント内で管理する永続データ構造の実装設計を行う。

(2)　製品系列開発プロセス

本プロセスは、SPL 開発のアプリケーション・エンジニアリングプロセスについて、モデル指向開発手法のモデリングフェーズを融合させ、具体的なアクティビティで構成した製品系列における個々の製品ソフトウェアを構築するための開発プロセスである。

図 92-20 は、製品系列開発プロセスの各アクティビティについて、その概要を説明している。

図 92-20 製品系列開発プロセスのアクティビティ

(a)　コア資産再利用分析（フィーチャモデリング）

コア資産開発プロセスの製品フィーチャ分析にて作成されたフィーチャマップとフィーチャマトリクスを基に今回の開発対象である製品について再利用するフィーチャ（コア資産）の製品機能と製品固有のフィーチャ（アプリ資産）である製品機能について共通性、可変性の視点で分析する。その際、新たに再利用できる製品機能があれば、コア資産管理プロセスにて製品機能のコア資産化を進めることになる。また、再利用するフィーチャに追加変更が必要になった場合には、コア資産管理プロセスにて別なコア資産として開発することになる。これらの分析結果は、既存のフィーチャマップとフィーチャマトリクスにフィードバックして更新しなければならない。

ここで、製品ソフトウェア要件定義はモデリングの対象にしていないので説明を省略する。

(b) 製品ソフトウェア要件分析（要求モデリング）

製品ソフトウェア要件定義にて定義した今回の開発である製品の要求仕様、および外部インタフェース仕様を基にして、製品固有のフィーチャ（アプリ資産）である製品機能について要件分析を行う。要件分析には、モデル指向開発手法の要素技法である 3 種類の視点による分析（ユースケース分析、ステートケース分析、イベントケース分析）を必要に応じ組合せて実施する。

(c) 製品アーキテクチャ設計（分析モデリング）

コア資産開発プロセスにおいて構築済みの基盤アーキテクチャを参照アーキテクチャとして、製品ソフトウェア要件分析の結果から製品機能の単位をコンポーネントに割当て、製品機能内のデータと処理に関する責務を抽出してクラスに割当てながら、製品固有のアーキテクチャを設計していく。

(d) 製品構造設計（設計モデリング）

固有の製品機能を割当てたソフトウェアコンポーネントの内部構造とインタフェース仕様を設計する。コア資産化可能なインナコンポーネントの切り出しについてもこの段階で検討する。内部構造の設計については設計クラス図を作成し、インタフェース仕様の設計についてはインタフェース構造図を作成する。また、内部構造として持つ永続データの論理構造も設計する。固有の製品機能を割当てたソフトウェアコンポーネントについては、インタフェース仕様に応じたシーケンス図を作成し、構造設計の妥当性を検証する。最後に製品全体の振る舞いについて、製品で扱うイベント単位にコンポーネント間のシーケンス図を作成し、製品アーキテクチャの妥当性を検証する。

(e) 製品処理設計（実装モデリング）

製品ソフトウェアとしての実行環境となるプロセス、またはスレッドの設計や固有の製品機能を割当てたソフトウェアコンポーネントを構成する各クラスの処理メソッドである関数仕様を設計する。また、コンポーネント内で管理する永続データ構造の実装設計を行う。

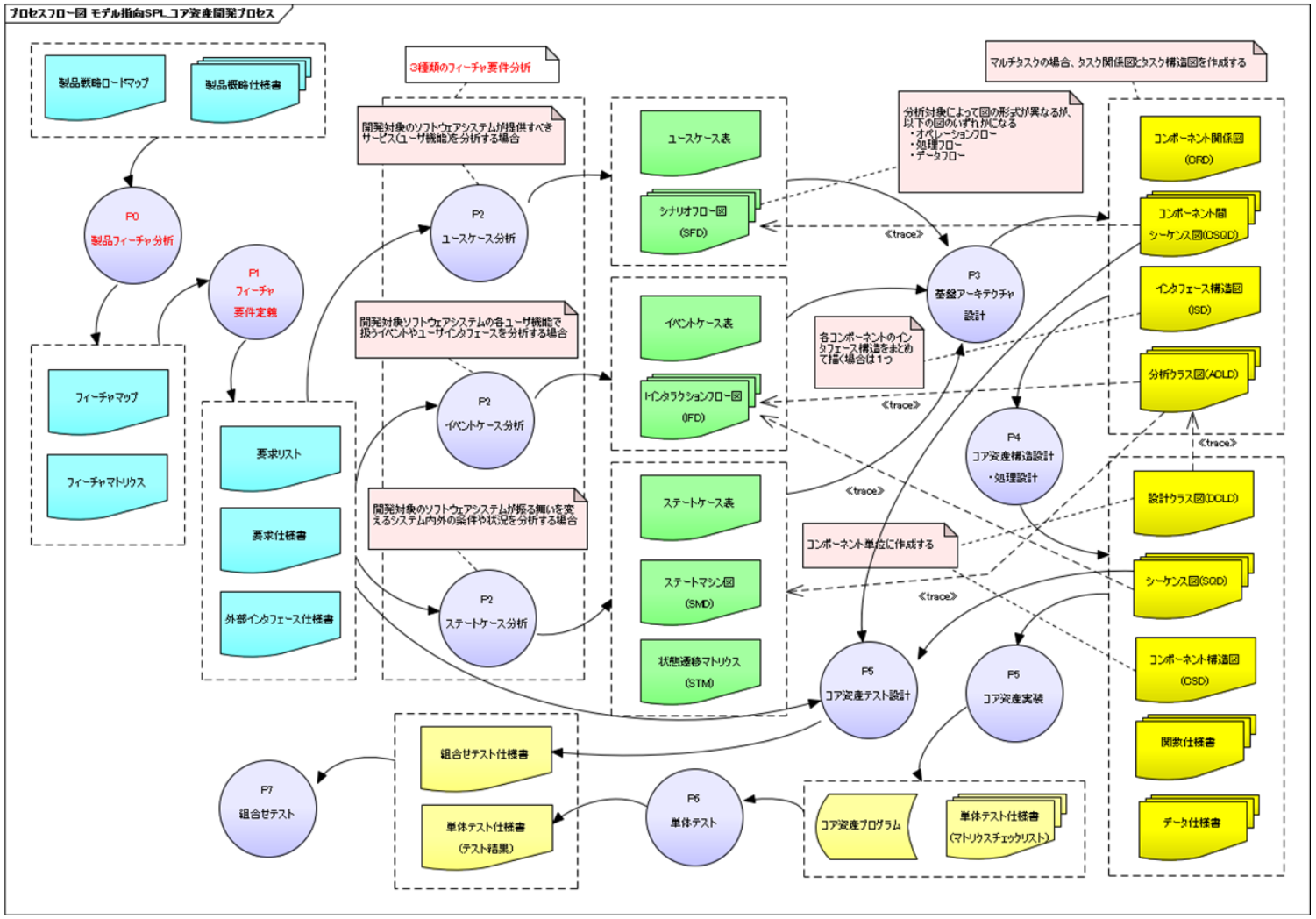


図 92-21 モデル指向 SPL 開発のコア資産開発プロセスフロー

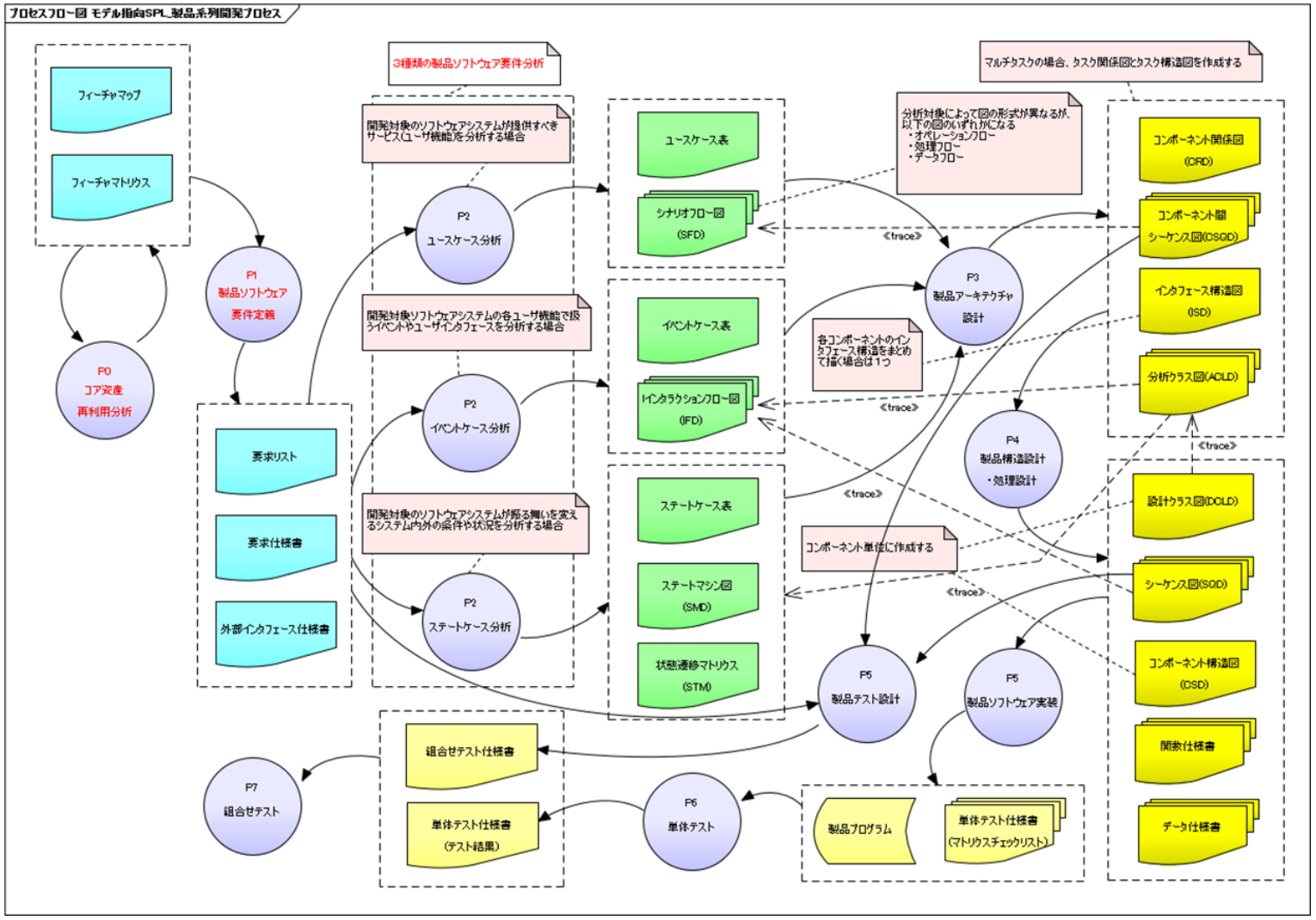


図 92-22 モデル指向 SPL 開発の製品系列開発プロセスフロー

(3) コア資産管理プロセス

本プロセスは、SPL 開発のアプリケーション・エンジニアリングプロセスに対応した製品系列開発プロセスと並行して進めるコア資産の再利用に関する維持管理と評価、およびコア資産の追加変更のための管理プロセスである。

図 92-23 は、モデル指向 SPL 開発におけるコア資産管理プロセスの流れと製品系列開発のアプリケーション・エンジニアリングプロセスとの関係について示している。

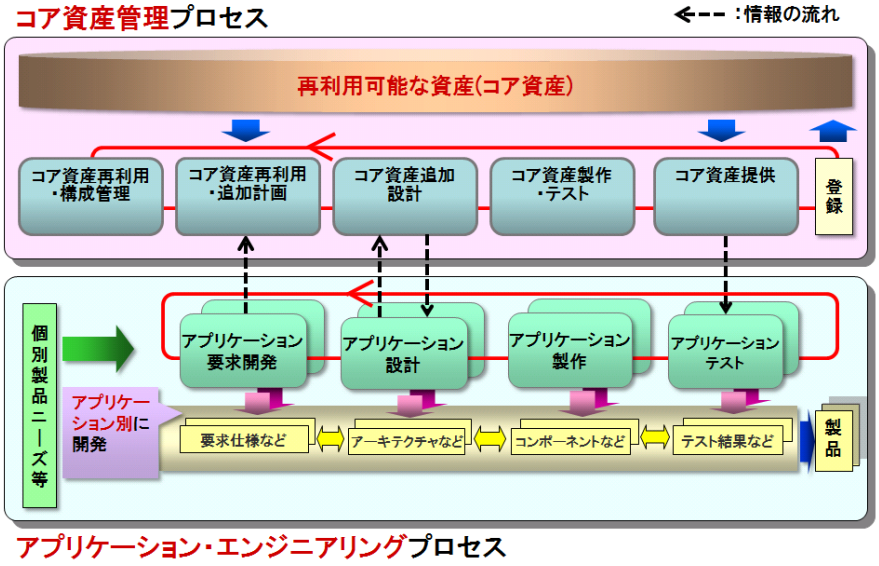


図 92-23 モデル指向 SPL 開発におけるコア資産管理プロセスの位置付け

(a) コア資産再利用・構成管理

コア資産開発プロセスで事前に開発された製品系列のコア資産コンポーネント、および本プロセスで追加開発されたコア資産コンポーネントについて、再利用の評価や製品系列開発で再利用するための提供、およびコア資産全体の構成管理を行う。

(b) コア資産再利用・追加計画

製品系列開発におけるコア資産再利用の要求やコア資産追加開発の要求を受けて、コア資産提供と追加開発のための計画を立てる。

(c) コア資産追加設計

製品系列開発からのコア資産追加開発の要求があった場合、コア資産の要求仕様を基に要件分析とコンポーネント設計を行う。

(d) コア資産製作・テスト

コア資産の追加開発要求により設計したコア資産コンポーネントの実装、およびテストを行う。

(e) コア資産提供

追加開発したコア資産コンポーネントを製品系列開発に提供する。製品系列開発において製品ソフトウェアの総合テスト実施時、インタフェース仕様に変更やコア資産コンポー

ネントに不良が発生した場合は、対応して再提供を行う。

図 92-24 にコア資産管理プロセスで管理対象とするコア資産形態の分類を示す。

○:適用有、△:場合により適用、ー:適用無

コア資産形態	結合方式		
	コンパイルーリンク	バイナリ形式ーリンク	ダイナミックリンク
パッケージ	△	○	ー
コンポーネント	ー	○	ー
クラス/モジュール	ー	○	○
メソッド/関数	ー	○	○

★ メソッド/関数内の部分的なプログラムコード群は、コア資産形態の対象として扱わない。

図 92-24 コア資産形態の分類

コア資産管理プロセスでは、上述の管理以外にコア資産の再利用に関して継続的な評価を行っていく。評価指標としては、コア資産側からの視点として「コア資産利用率」を使用し、製品系列側からの視点として「コア資産適用率」を使用している（図 92-25、図 92-26、図 92-27 を参照）。

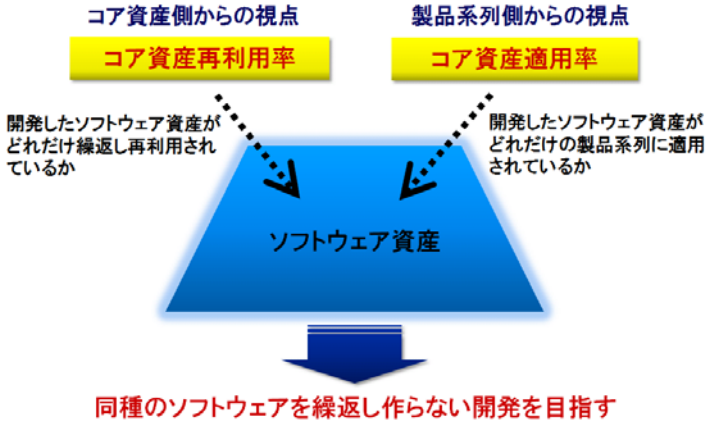


図 92-25 SPL 開発適用効果の定量的評価指標

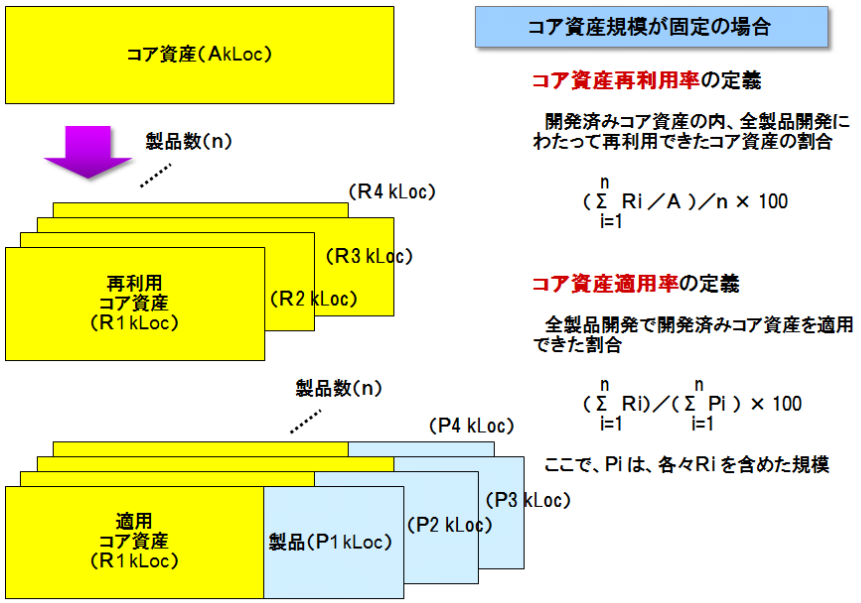


図 92-26 コア資産の再利用率と適用率

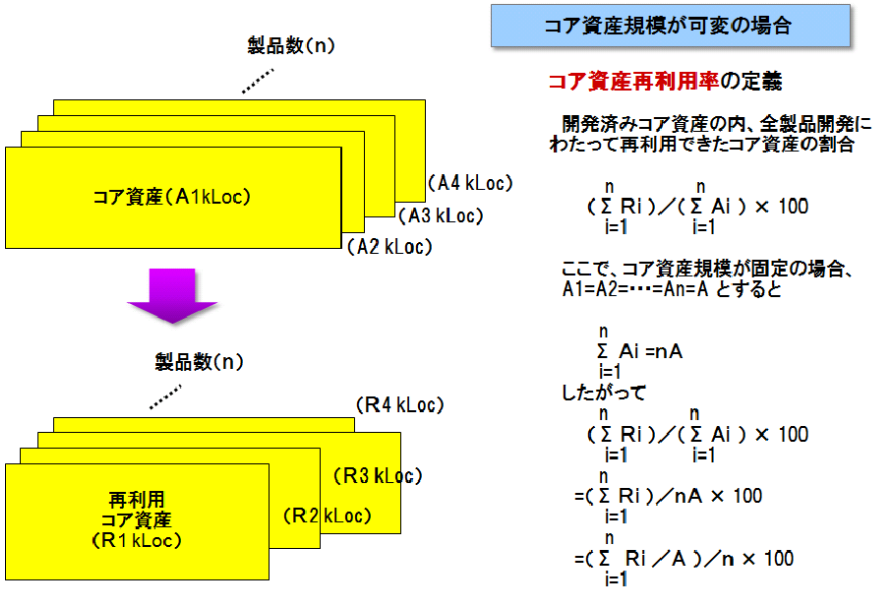


図 92-27 可変なコア資産の再利用率

3.3. 課題解決の展開

前述の 2．1 で提示した 3 つの課題に対して、モデル指向 SPL 開発手法の適用によって、2．2 で提示した課題解決のための目標と着眼点に基づき、どのように課題を解決していったのか、具体的な展開について説明する。

課題 1：SPL 開発を導入して製品開発を進めるための具体的な開発手順を示す

［**着眼点**］一般に SPL 開発は製品戦略ロードマップを入力として開始される。SPL 開発におけるコア資産開発のドメイン・エンジニアリングでは、製品系列のドメインを対象としてドメイン

要求開発、ドメイン設計、ドメイン製作、ドメインテストといったサブプロセスで進められていく。また、SPL 開発における製品系列開発のアプリケーション・エンジニアリングでは、製品系列の個々の製品を対象としてアプリケーション要求開発、アプリケーション設計、アプリケーション製作、アプリケーションテストといったサブプロセスで進められていく。これらの各サブプロセスにおいて、具体的な開発のアクティビティの流れと各アクティビティで作成される具体的な成果物を定義することで、ソフトウェア開発の現場に導入できるようにすることを考えた。そして、これらの具体的なアクティビティと作成する成果物にモデリング技術を適用することにした。

［展開］SPL 開発のドメイン・エンジニアリングプロセス、アプリケーション・エンジニアリングプロセスについて、各々モデル指向 SPL 開発のコア資産開発プロセス、製品系列開発プロセスを対応付け、各プロセスにモデル指向開発手法を融合させて、具体的なアクティビティと成果物としてのモデルを定義した。各プロセスの全体は、図 21、図 22 のプロセスフローに示したので、それらを構成する各フェーズのアクティビティと成果物について以下に説明する。

(1) コア資産開発プロセス

① P0：製品フィーチャ分析

製品戦略ロードマップ、製品概略仕様書を基に製品系列の共通性、可変性を分析する。

成果物：フィーチャマップ、フィーチャマトリクス

② P1：フィーチャ要件定義

コア資産となる製品系列機能の要求仕様、および外部インタフェース仕様を定義する。

成果物：要求リスト、要求仕様書、外部インタフェース仕様書

③ P2：フィーチャ要件分析

コア資産要求仕様のユースケース、ステートケース、イベントケース、および機能間の関係を分析する。

成果物：ユースケース表、シナリオフロー図（オペレーションフロー図など)、
ステートケース表、ステートマシン図（状態遷移図)、状態遷移マトリクス、
イベントケース表、インタラクションフロー図
タスク構造図・タスク関係図

④ P3：基盤アーキテクチャ設計

製品系列全体の共通基盤やコア資産コンポーネントのアーキテクチャを設計する。

成果物：機能関係図、機能フロー図、
コンポーネント関係図（コンポーネント図)、コンポーネント間シーケンス図、
コンポーネントのインタフェース構造図と分析クラス図

⑤ P4：コア資産構造設計・処理設計

各コア資産コンポーネントの内部構造と振る舞い、およびデータ構造を設計する。

成果物：コンポーネントの設計クラス図とコンポーネント構造図、データ仕様書、
シーケンス図、プロセス/スレッド構造図、タイミング設計図、関数仕様書

⑥ P5：コア資産テスト設計

コア資産コンポーネントの組合せテスト方式と仕様項目、テストデータを設計する。
成果物：組合せテスト仕様書

⑦ P5：コア資産実装
コア資産コンポーネントのプログラミングと単体テスト方式と仕様項目を設計する。
成果物：コア資産プログラム、単体テスト仕様書

⑧ P6：単体テスト
コア資産コンポーネントの各クラスにおける関数プログラムの単体テストを実施する。
成果物：単体テスト仕様書（テスト結果）

⑨ P7：組合せテスト
各コア資産コンポーネント内クラス間とコンポーネント間の組合せテストを実施する。
成果物：組合せテスト仕様書（テスト結果）

(2) 製品系列開発プロセス

① P0：コア資産再利用分析
開発対象製品の製品共通機能と製品固有機能を分析し再利用するコア資産を確定する。
成果物：フィーチャマップ更新版、フィーチャマトリクス更新版

② P1：製品ソフトウェア要件定義
開発対象となる製品固有機能の要求仕様、および外部インタフェース仕様を定義する。
成果物：要求リスト、要求仕様書、外部インタフェース仕様書

③ P2：製品ソフトウェア要件分析
製品固有機能要求仕様のユースケース、ステートケース、イベントケースを分析する。
成果物：ユースケース表、シナリオフロー図（オペレーションフロー図など）、
ステートケース表、ステートマシン図（状態遷移図）、状態遷移マトリクス、
イベントケース表、インタラクションフロー図
タスク構造図・タスク関係図

④ P3：製品アーキテクチャ設計
再利用するコア資産を含めた開発対象となる製品全体のアーキテクチャを設計する。
成果物：機能関係図、機能フロー図、
コンポーネント関係図、コンポーネント間シーケンス図（分析シーケンス図）、
コンポーネントのインタフェース構造図と分析クラス図

⑤ P4：製品構造設計・処理設計
製品固有コンポーネントの内部構造と振る舞い、およびデータ構造を設計する。
成果物：コンポーネントの設計クラス図とコンポーネント構造図、データ仕様書、
シーケンス図、プロセス/スレッド構造図、タイミング設計図、関数仕様書

⑥ P5：製品テスト設計
製品固有コンポーネントの組合せテスト方式と仕様項目、テストデータを設計する。
成果物：組合せテスト仕様書

⑦ P5：製品実装

製品固有コンポーネントのプログラミングと単体テスト方式と仕様項目を設計する。

成果物：製品プログラム、単体テスト仕様書

⑧ P6：単体テスト

製品固有コンポーネントの各クラスにおける関数プログラムの単体テストを実施する。

成果物：単体テスト仕様書（テスト結果）

⑨ P7：組合せテスト

各製品固有コンポーネント内クラス間とコンポーネント間の組合せテストを実施する。

成果物：組合せテスト仕様書（テスト結果）

課題 2：製品機能の可変性をソフトウェア構造につなげる具体的方法を確立させる

［着眼点］製品機能の可変性からソフトウェアコンポーネント構造までは、かなりの隔たりがあり、その間をつなげる何らかの具体的な分析手順が必要になる。そこで製品機能の可変性を分析する方法から考えなければならない。製品系列における製品機能の可変性を分析する以前に、製品戦略ロードマップから今後市場にリリースしていく製品に織り込む固有な機能を明確にすることが必要になる。その上で製品系列全体の製品機能について、共通性と共に可変性を分析し、分析結果をモデル化していく（これを「フィーチャモデリング」と呼ぶ)。モデル化するには、製品機能全体が俯瞰（ふかん）できて、かつ各製品機能の可変点と可変部が見えるような分析モデルが最適である。そのモデルで明確になった可変点と可変部からソフトウェアコンポーネントの単位とインタフェース仕様を導出し、製品機能の要求仕様を分析することで、論理的にソフトウェアコンポーネントの構造と振る舞いを確定させていく。

［展開］モデル指向 SPL 開発では、再利用の単位を極力大きな粒度の責務とするため、製品のフィーチャである製品機能に着目する。製品フィーチャ分析の結果として得られるフィーチャマップ、フィーチャマトリクスにおける可変性の可変点をフィーチャコンポーネントとして大きな粒度の責務で切り出すポイントとした。すなわち、製品機能の可変性において可変点をコンポーネントのインタフェースに割当て、可変部をフィーチャコンポーネントに割当てることにする。ただし、これだけの発想では、製品機能の可変性をソフトウェア構造につなげることはできない。そこで、製品機能の要求仕様について、モデル指向開発手法の要件分析技法*を適用することで、製品機能の可変性をフィーチャコンポーネント構造へつなげる具体的な方法を確立させた。

以下に製品機能の可変性からフィーチャコンポーネント構造へつなげる手順を説明する。

(1) 可変点を基に可変部をフィーチャコンポーネントの単位とする

フィーチャモデリングの結果から、可変点は、「必須」（共通)、「任意選択」（オプション)、「複数選択」（最低 1 つ選択)、「排他選択」（択一的選択）のいずれかに分類される。

(2) 可変点をフィーチャコンポーネントのインタフェースに割当てる

可変点をインタフェースに割当てるとは、コンポーネント選択の基準となる情報をインタフェースとしてパラメータ化するということである。

(3) 製品機能の要求仕様をユースケース、ステートケース、イベントケースで分析する

モデル指向開発手法の要件分析技法*である 3 種類の視点における分析法を組合せて、各製品機能について定義された要求仕様の振る舞いを詳細化して分析する。

(4) イベントケース分析のインタラクションフローから責務を抽出してクラスに割当てる

3 種類の要件分析技法*において、直接ソフトウェア構造につなげることを可能にするのは、イベントケース分析であり、作成するインタラクションフローの中で抽出する責務である。

(5) フィーチャコンポーネントのクラス構成を設計する

イベントケース分析で抽出した責務をクラスに割当て分析レベルのクラスの構造とクラス構成を決め、それらを集約して設計レベルのクラス構造とクラス構成を設計する。

(6) フィーチャコンポーネントのインタフェース構造を設計する

コンポーネントの責務と構造を考慮して、コンポーネントの外部から提供が必要なデータとコンポーネントの外部へ提供すべきデータを決定しインタフェースの関数仕様を設計する。

(7) フィーチャコンポーネント単位に処理シーケンスを作成し構造の妥当性を検証する

各コンポーネントの構造とインタフェース構造から、イベントケース分析で作成したインタラクションフロー単位に処理シーケンスを作成する。

(8) フィーチャコンポーネントの内部構造を決定する

フィーチャコンポーネント単位の処理シーケンスにてコンポーネント構造の妥当性を検証できた段階で、最終的なフィーチャコンポーネントのコンポーネント構造図を作成する。

＊注記：モデル指向開発手法の要件分析技法についての詳細は、別稿の『統合モデル技術によるユーザ主体のソフトウェア開発手法「モデル指向開発（MOD)」の紹介』を参照のこと。

課題 3：製品系列開発を繰返していく過程で再利用に関する評価、管理の方法を示す

［着眼点］コア資産は最初の製品系列開発が開始される前に事前に開発されるのが一般的であるが、実際の開発においては、製品系列開発を繰返す過程で必要となるコア資産やコア資産の機能拡張などが発生する。そこで、製品系列開発が開始された後にコア資産の再利用性評価や追加開発、および機能拡張などに対応するプロセスが必要となる。このプロセスでは製品系列の個々の製品とは異なるコア資産のみを対象とするため、製品系列開発のプロセスとは別な独立したプロセスにすべきと考えた。これを「コア資産管理プロセス」と呼び、プロセスに必要なアクティビティとそれらの手順や方法を整理してまとめ、ガイドすることにした。

［展開］モデル指向 SPL 開発では、SPL 開発の標準プロセスとして定義されていない管理プロセスとして「コア資産管理プロセス」を定義し、さらに本プロセスを構成する具体的なフェーズとアクティビティ、および成果物を定義した。図 92-23 にコア資産管理プロセスの全体と製品系列開発を行うアプリケーション・エンジニアリングプロセスとの関係を示したので、以下にプロセスを構成する各フェーズのアクティビティと成果物について説明する。

(1) コア資産再利用・構成管理

製品系列開発で再利用するコア資産の提供管理やコア資産全体の構成管理を行う。

成果物：コア資産提供管理表、コア資産構成管理表

(2) コア資産再利用・追加計画

製品系列開発からのコア資産の再利用、追加開発の要求を受けて計画を立てる。

成果物：コア資産提供計画、コア資産追加開発計画

(3) コア資産追加設計

製品系列開発からのコア資産追加開発要求によりコア資産コンポーネントを設計する。

成果物：コンポーネントの設計クラス図とシーケンス図、
コンポーネント構造図、関数仕様書、データ仕様書

(4) コア資産製作・テスト

コア資産追加設計で設計したコア資産コンポーネントの実装、およびテストを行う。

成果物：コア資産プログラム、単体・組合せテスト仕様書

(5) コア資産提供

追加開発したコア資産コンポーネントを製品系列開発に提供し製品テストに対応する。

成果物：コア資産コンポーネント利用ガイド、コア資産プログラム（バイナリ形式）

4. 適用手法の評価と確認できた効果

4.1. 課題解決に対する評価

2.2 で提示した 3 つの課題の解決に向けた目標の達成度について、その評価を以下に示す。

(1) **課題 1**：SPL 開発を導入して製品開発を進めるための具体的な開発手順を示す

［**目標**］SPL 開発におけるコア資産開発のためのドメイン・エンジニアリングと製品系列開発のためのアプリケーション・エンジニアリングについて、書籍の一般的な概念や断片的な活動の解説だけでは、製品系列を持つソフトウェア開発の現場で手順を具体化して開発業務につなげることが極めて困難である。開発現場に適した具体的な手順としての開発プロセスが必要になる。SPL 開発の考え方は概念的に理解しやすく、大きな効果を期待できそうであることも直感的に理解しやすい。しかし、ソフトウェア開発の現場では、組織的な活動として、どこから、どのように手を着けていけば無理なく導入が可能となるのかが分からないでいる。また、規模が大きな開発方式なだけに、その分導入のリスクも大きくなることが懸念される。そのため、開発手順に関する障壁をすべて解決する具体的な開発プロセスを示す。

［**評価**］モデル指向 SPL 開発手法として製品戦略ロードマップを基に再利用可能なコア資産を開発するコア資産開発プロセスとコア資産を再利用して個々の製品系列製品を開発する製品系列開発プロセスの具体的なフェーズを定義した。そして、2 つの開発プロセスの各フェーズにおける具体的なアクティビティと成果物を定義することで、ソフトウェア開発の現場で SPL 開発を導入して製品開発を進めるための具体的な開発手順を示すことができた。

(2) **課題 2**：製品機能の可変性をソフトウェア構造につなげる具体的方法を確立させる

［**目標**］製品機能の可変性は、機能の選択可能性につながり、現在把握できている機能のバリエーションだけでなく、将来の機能強化や機能拡張への可能性を広げることのできる性質である。この性質をソフトウェア構造に置換える上で最も適している要素がソフトウェアコンポーネントである。ソフトウェアコンポーネントは、抽象化された汎用的なインタフェースを持ち、

粒度の大きな責務でカプセル化、および情報隠蔽された独立性の高いソフトウェア構造である。モデル指向開発手法を適用することによって、製品機能の可変性からソフトウェアコンポーネント構造へ論理的につなげる方法を確立させる。

［評価］モデル指向 SPL 開発のフィーチャモデリングを適用した製品フィーチャ分析によって製品機能の共通性と可変性を分析し、モデル指向開発手法の要件分析技法*を適用した製品機能の要件分析によって可変点と可変部をソフトウェアコンポーネントのインタフェースと内部構造に割当てることができた。そして、製品機能の可変性をコア資産コンポーネントのソフトウェアにつなげる具体的方法を確立することができた。

＊注記：モデル指向開発手法の要件分析技法についての詳細は、別稿の『統合モデル技術によるユーザ主体のソフトウェア開発手法「モデル指向開発（MOD)」の紹介』を参照のこと。

(3) **課題 3**：製品系列開発を繰返していく過程で再利用に関する評価、管理の方法を示す

［目標］SPL 開発では、コア資産開発のドメイン・エンジニアリングと製品系列開発のアプリケーション・エンジニアリングのプロセスが提案されているが、開発したコア資産について再利用と構成管理の視点で評価、管理するプロセスが示されていない。ソフトウェア開発の現場でコア資産開発と製品系列開発を繰返し継続していく中で、開発プロセスとは別に必要となる管理プロセスである。本プロセスにおける具体的な手順と方法を示すことで、ソフトウェア開発の現場へ SPL 開発を導入できるようにする。

［評価］モデル指向 SPL 開発手法では、SPL 開発の標準開発プロセスであるドメイン・エンジニアリングプロセスとアプリケーション・エンジニアリングプロセスに加え、第 3 のエンジニアリングプロセスとしてコア資産管理プロセスを定義した。そして、コア資産管理プロセスにおける各フェーズのアクティビティと成果物を定義し、ソフトウェア開発の現場でコア資産の再利用に関する評価と管理の具体的な方法を示すことができた。

4.2. 確認できた効果

これまでモデル指向 SPL 開発を適用した開発プロジェクトにて確認できている効果について、図 92-28 に示す。以降、確認できた効果を定量的効果と定性的効果に分類して説明する。

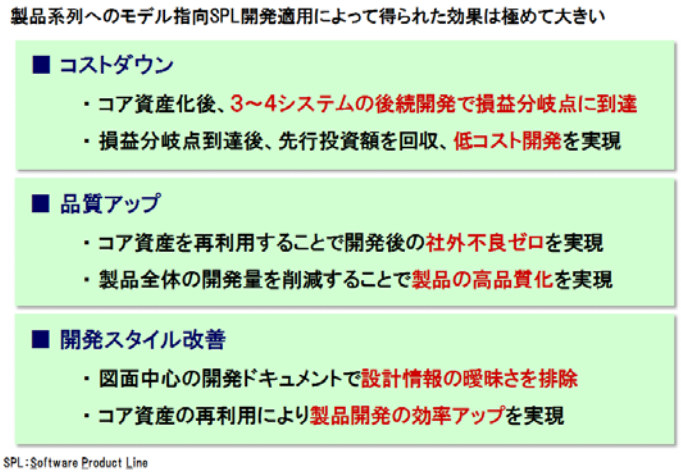


図 92-28 モデル指向 SPL 開発の適用で確認できた効果

(1) 定量的効果

SPL 開発における開発コストは、単一システムとして開発した際の開発コストと比較すると、理論上、約 3 システム目で損益分岐点に達すると言われている。実際にモデル指向 SPL 開発を適用した開発プロジェクトにおいて、開発実績を評価したところ、理論通りの効果が得られたことに驚きながら、有効な開発手法であることが評価できた。

モデル指向 SPL 開発適用後、評価時点で得られた従来開発との工数比較、および累積工数の推移に関する評価例について、その内容を図 92-29 に示す。

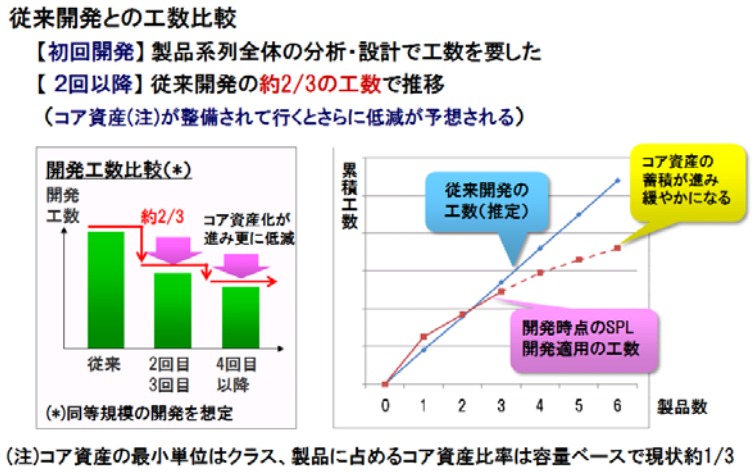


図 92-29 確認できた適用の定量的評価

モデル指向 SPL 開発適用の品質に関する定量的効果としては、適用した開発プロジェクトにて開発された製品について、納入後の不良発生件数を 0 に抑えられることが確認されている。

(2) 定性的効果

モデル指向 SPL 開発の適用によって得られた定性的な効果は、適用前の期待を大幅に上回るものであった。当初期待していた開発量縮減と開発期間短縮による開発コストの低減、および製品の市場競争力強化だけでなく、コア資産を再利用することによる製品品質の長期的な安

定化が図れるようになった。また、製品のアーキテクチャを意識した開発方針や開発計画によるアーキテクチャ中心の開発スタイルに移行することができた。適用によって得られた具体的な効果の内容について図 92-30 に示す。

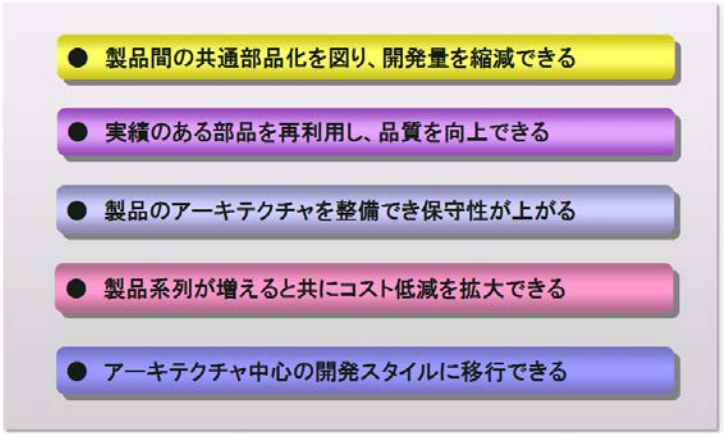


図 92-30 確認できた適用の定性的効果

5. 今後の取組みと課題

5.1. 今後の取組み

現在も多くの開発プロジェクトでは、製品系列の開発について単一システム開発として扱い派生開発を繰返していたり、無計画に類似のソフトウェアを繰返し開発したりしている。例え開発が無事に終了できたとしても、開発コストの面から客観的に評価した場合に、長期的には大きなロスコストにつながっている可能性がある。そこで今後の製品系列開発について、以下のような取組みを進めていき、モデル指向 SPL 開発の適用によってソフトウェア開発の現場に大きな効果が得られることを示していきたい。

(1) SPL 開発の普及

SPL 開発は、製品系列の開発において、コスト、品質の両面で大きな効果を引き出すことが確認できているが、組織的、かつ長期的で先行投資を伴う開発方式のため、導入するには開発部門におけるトップの戦略と判断が必要となる。このハードルをクリアするためには、開発部門において製品系列開発についての戦略的なロードマップを作成しなければならない。製品戦略ロードマップが作成できれば、SPL 開発へのスタートラインに立つことができ、これが SPL 開発を普及させる鍵となる。

(2) 開発支援ツールの活用

SPL 開発において支援ツールが必要となるのは、再利用可能なコア資産を含む製品系列の個々のソフトウェア資産の構成管理やトレーサビリティ管理の領域である。モデル指向 SPL 開発では、各種のモデルが開発成果物となるため、支援ツールには、モデルを通じて個々の製品を構成するコア資産とアプリ資産を体系的に管理し、製品ソフトウェアとしてビルドする機

能が求められる。これらの要求を実現するには、モデリングツール、構成管理ツール、およびトレーサビリティ管理ツールの各々におけるツール間の連携が必要となる。

5.2. 今後の課題

モデル指向 SPL 開発手法の適用範囲を拡張することにより、SPL 開発を導入する開発現場にもたらす効果をより大きなものにしたいと考えている。そのための課題としては、以下の 2 項目が挙げられる。

(1) テストプロセスのプロダクトライン化

SPL 開発の対象となる開発プロセスとしては、製品フィーチャ分析からソフトウェア実装後に実施するテストプロセスまでである。テストプロセスにおけるコア資産の対象には、テスト方式、テスト仕様、テスト実行環境、テストデータ生成、テスト結果評価など、多岐にわたる。これらにモデリング技術を適用することで、再利用可能なコア資産としてのテストモデルを構築することができると考えている。

テストプロセスにおけるコア資産化の仕掛けを検討する前に、ソフトウェアテストモデリングについての方法論を確立させなければならない。これは、「モデル指向テスティング」として、モデル指向開発手法と融合させるべき分野であり、今後の課題の 1 つに位置付けている。

(2) システムプロダクトライン開発

元来、プロダクトライン開発は、ハードウェアエンジニアリングの分野で考案され発展してきた開発方式であり、その後、ソフトウェアエンジニアリングの分野にも導入されたものである。そのような経緯から、次に導入される対象としては、システムエンジニアリングの分野をおいて考えることはできない。モデリング技術についても、MBSE（Model Based Systems Engineering）に代表されるように、その対象範囲をソフトウェアエンジニアリングの分野からシステムエンジニアリングの分野に拡大されつつある。

このような状況の中、今後、検討しなければならない方向性としては、製品系列におけるハードウェアとソフトウェアをシステム製品として一体化して扱い、系統的に開発していくシステムレベルのプロダクトライン開発であると考えている。それには、ハードウェアとソフトウェア間の依存性をどのように制御して管理していくかということが大きな課題となる。

参考文献

1. ソフトウェアプロダクトライン

[1] ソフトウェアプロダクトライン―ユビキタスネットワーク時代のソフトウェアビジネス戦略と実践　Paul Clements、Linda Northrop、 前田 卓雄訳、日刊工業新聞社、2003.10

[2] 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　クラウス・ポール、ギュンター・ベックレ、フランク・ヴァン・デル・リンデン、林 好一・吉村 健太郎・今関 剛訳、エスアイビー・アクセス、2009.1

[3] マルチパラダイムデザイン　James O. Coplien、金澤 典子・平鍋 健児・羽生田 栄一訳、ピアソン・エデュケーション、2001.12

[4] SOFTWARE FACTORIES ソフトウェアファクトリー　Jack Greenfield、Steve Cook、Keith Short、Stuart Kent、野村 一行訳、日経 BP 社、2005.12

[5] ジェネレーティブプログラミング　クシシュトフ・チャルネッキ、ウールリシュ・W・アイセンアッカー、津田 義史・今関 剛・朝比奈 勲訳、翔泳社、2008.4

2. ソフトウェアアーキテクチャ

[6] ソフトウェアアーキテクチャ―ソフトウェア開発のためのパターン体系　Frank Buschmann、Hans Rohnert、Michael Stal、Regine Meunier、Peter Sommerlad、金澤 典子他訳、近代科学社、2000.12

[7] 実践ソフトウェアアーキテクチャ　Len Bass、Rick Kazman、Paul Clements、前田 卓雄他訳、日刊工業新聞社、2005.10

[8] システムアーキテクチャ構築の実践手法　ピーター・イールズ、ピーター・クリップス、榊原 彰・西原 裕善・吉田 幸彦他訳、翔泳社、2010.12

[9] アーキテクチャ中心設計手法　アンソニー・J・ラタンゼ、橘高 陸夫他訳、翔泳社、2011.1

[10] システムアーキテクチャ構築の原理～IT アーキテクトが持つべき 3 つの思考　Nick Rozanski、Eoin Woods、榊原 彰・牧野 祐子訳、翔泳社、2008.12

[11] ソフトウェアシステムアーキテクチャ構築の原理 第 2 版 IT アーキテクトの決断を支えるアーキテクチャ思考法　Nick Rozanski、Eoin Woods、榊原 彰・牧野 祐子訳、SB クリエイティブ、2014.9

[12] ビューティフルアーキテクチャ　Diomidis Spinellis、Georgios Gousios、久野 禎子・久野 靖訳、オライリージャパン、2009.11

[13] マイクロサービスアーキテクチャ　Sam Newman、佐藤 直生・木下 哲也訳、オライリージャパン、2016.2

3. ソフトウェアコンポーネント

[14] オブジェクト指向とコンポーネントによるソフトウェア工学―UML を使って　Perdita Stevens、Rob Pooley、児玉 公信訳、ピアソン・エデュケーション、2000.9

[15] コンポーネントモデリングガイド―カタリシスベースのコンポーネント開発　Fondatao Inc.、長瀬 嘉秀・今野 睦訳、ピアソン・エデュケーション、2001.9

[16] UML コンポーネント設計―コンポーネントベースソフトウェアのための開発プロセス

John Cheesman、John Daniels、長瀬 嘉秀・今野 睦訳、ピアソン・エデュケーション、2002.5

4. ソフトウェア再利用

[17] ソフトウェア・モデリング―ソフトウェア再利用のための設計パラダイム　片岡 雅憲、日科技連出版社、1988.9

[18] ソフトウェア再利用技術―オープン・ソフトウェアの開発パラダイム　片岡 雅憲、日科技連出版社、1992.4

[19] ドメイン分析・モデリング―これからのソフトウェア開発・再利用基幹技術　伊藤 潔、杵嶋 修三、田村 恭久、廣田 豊彦、吉田 裕之、共立出版、1996.8

[20] ソフトウェア再利用ガイドブック―アーキテクチャー、プロセス、組織の変革による再利用ビジネス成功への道　Ivar Jacobson、Patrik Jonsson、Matin Griss、杉本 宣男・落合 修・吉田 幸彦・ 田中 正仁訳、トッパン、1999.10

掲載されている会社名・製品名などは、各社の登録商標または商標です。

独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター (IPA/SEC)