大規模移動体ネットワーク機器ファームウェア開発へのソフトウェアプロダクトライン適用事例

大塚 潤[†] 河原畑 光一[†] 岩崎 孝司[†]
内場 誠[†] 中西 恒夫[††]
久住 憲嗣[††] 福田 晃[††]

分散拠点の総開発人員 300 名を超える大規模プロジェクトにおいて，フィーチャ数 2,000 を越える移動体ネットワーク機器のファームウェアのプロダクトライン開発を行った．プロダクトライン全体に対してフィーチャモデリングを実施し，製品間共通性／相違性の全体像を把握した．そのうえで，リリースが約束されている少数の製品を対象に，プロダクトライン開発の効果が早くかつ高く現れるよう，コア資産を構築するフィーチャを厳選した．限定的なプロダクトライン開発にもかかわらず，これまでの製品個別開発と比べて，品質，コスト，工期のすべてにおいてプロダクトライン開発部には大きな改善が見られ，プロダクトライン開発適用外の部分も含めた全体の改善を牽引した．ソースコード自動生成手法の併用がその改善に大きく貢献した．

Developing a Large-Scale Product Line of Mobile Network Equipment Firmwares

Jun Ohtsuka,[†] Kouichi Kawarabata,[†] Takashi Iwasaki,[†]
Makoto Uchiba,[†] Tsuneo Nakanishi,[††] Kenji Hisazumi[††]
and Akira Fukuda[††]

Product line engineering (or PLE) was applied to development of firmware for mobile network equipments having more than 2,000 features in a large scale project with more than 300 engineers in distributed sites. Feature modeling was performed for the whole product line to know a perspective view of commonality and variability among the products. Core assets were constructed for the features that we chose carefully for earlier and higher gain of PLE development based on the perspective view. Although PLE development was applied to a limited part of the system, in terms of quality, cost and development period, PLE development brought a considerable amount of improvement for PLE developed portions and led to successful improvement of the whole system. The improvement was largely a result of automatic generation of source codes which is performed in the development.

1. はじめに

富士通九州ネットワークテクノロジーズ（株）（以下，富士通 QNET）は，富士通（株）の出資により設立された設計開発専門会社である．設計開発の専門会社として開発資産の部品化や再利用の取組みは従来から実施してきたが，個々の技術者の属人的な判断のもと，再利用ができそうな開発資産をリポジトリに蓄積する部品指向のボトムアップ型再利用が主であったため，再利用による開発効率化が頭打ちとなっていた．

そのような中，国内の技術メディアや地域の組込みソフトウェア技術者のコミュニティを通してプロダクトライン開発手法に関する情報に触れる機会が増し，ビジネス，アーキテクチャ，プロセス，組織面から再利用環境を築き上げるプロダクトライン開発手法（PLE: Product Line Engineering）[1] に注目するようになった．文献 2)，3) で述べたように，2007 年に始まる活動を通し，富士通 QNET は PLE の全社的導入に成功した．

富士通 QNET の事業領域は，ネットワークシステムソリューションの開発，ならびにネットワーク機器や画像システムのハードウェア，ファームウェア，ソフトウェアの開発である．2007 年に始まる全社的導入ではこれら事業領域の多くにおいて PLE の導入が試み

† 富士通九州ネットワークテクノロジーズ（株）
Fujitsu Kyushu Network Technologies Ltd.
†† 九州大学
Kyushu University

られた．品質，コスト，工期のいわゆる QCD 面の大幅な改善を達成できたプロジェクトもあれば，PLE 適用の意味を見いだせなかったプロジェクトもあった．たとえば，パケット伝送装置プロダクトラインのファームウェア開発プロジェクトでは，性能要求を実現すべくアセンブリ言語レベルでの擦り合わせによる最適化を製品ごとに行っており，単純なモジュールの抜き差しや条件つきコンパイルによる製品間相違性の実現が困難であった．そのため，実装工程より前の工程には PLE を適用できたものの，実装工程では PLE を適用できず，PLE 適用の効果は期待ほどには得られなかった．また，プリント基板／装置開発のプロジェクトでは，ほとんどの製品間相違性は，プリント基板／装置開発部門の所掌範囲外である FPGA や ASIC などの内部回路で実現されているため，PLE 適用の意味そのものが見いだされなかった．本稿では，こうしたプロジェクトのうち，もっとも大規模に PLE を適用し，かつもっとも大きな成功をおさめた，移動体ネットワーク機器プロダクトラインのファームウェア開発プロジェクトについて報告する．

本稿は以下のように構成されている．第 2 節では，本プロジェクトで開発した移動体ネットワーク機器プロダクトラインの概要と本プロジェクトにおける PLE 導入の動機について述べる．第 3 節では，本プロジェクトにおける開発組織，開発チームの教育，開発プロセスについて述べる．本プロジェクトではまず，プロダクトライン全体を対象とする大規模なフィーチャモデリングを実施した．第 4 節ではそれについて述べる．第 5 節ではコア資産のスコーピングについて，第 6 節ではコア資産の開発について述べる．本プロジェクトではタスク間通信機能関連のフィーチャがコア資産のスコープに入れられ，同機能に関係するソースコードの自動生成が行われた．第 7 節では，本プロジェクトにおける QCD 面での見積りと評価，ならびに結果に対する考察を示す．第 8 節で関連研究を紹介し，第 9 節で本稿を総括する．

2. PLE 適用プロジェクトについて

本節では，移動体ネットワーク機器プロダクトラインの概要と同機器の開発における PLE 導入の動機について述べる．移動体ネットワーク機器プロダクトラインの開発プロジェクトは，富士通 QNET において初めて PLE を適用した実プロジェクトのひとつであり，プロジェクトと並行して各種ガイドラインや開発プロセスの拡張や改善が進められた．あわせて本節では，本プロジェクトにおける導入着手時の開発現場の状況について述べる．

2.1 移動体ネットワーク機器プロダクトラインの概要

移動体ネットワーク機器プロダクトラインの概要をビジネス面と技術面から述べるとともに，当該プロダクトラインにおいて製品間相違性が生じる理由を挙げる．

ビジネス面：プロジェクト着手時点で，同じ顧客への 2 製品のリリースは確定しており，それら 2 製品の仕様は確定していた．また，他の顧客への第 3 の製品の開発が計画されていたが，明確になっているのはコンセプトレベルまでであり，仕様までは確定していなかった．□

技術面：開発は地理的に離れた 4 拠点に行われ，実装には C 言語ならびにアセンブリ言語が使用された．開発要員は平均 100 名程度，開発規模は数百 KLine 以上，開発工期はおよそ 1 年以上（詳細はいずれも非公開）が予定された．なお，実績値はプロジェクト開始当時のこの見積値とほぼ一致していた．表 1 にプロジェクトの概要と各種データ（実績値）を示す．□

製品間相違性：この移動体ネットワーク機器プロダクトラインではどの製品も関連する技術標準に準拠するように作られている．この技術標準は無線周波数帯域幅，データ変調方式，経路多重化方式等のフィーチャを規定しているが，フィーチャの選択には余地があり，すべての機器ですべてのフィーチャを実現する必要はなく，顧客によって必要とされているフィーチャのみを選択的に実現することとなる．また，この技術標準は機能の追加や削除を伴う改訂が行われる．こうした技術標準上の製品間相違性は，過去の開発経験に基づいて，プロジェクト開始時から相当に想定することが可能であった．

技術標準外の機能，たとえば装置の内部状態や通信状態を監視するフィーチャなどは顧客によって求められるものが異なり，顧客ごとに製品をカスタマイズする必要が多分に生じる．しかし，これらの技術標準外の機能は，顧客によって全く異なる機能が求められるわけではなく，ある程度似通ったものが求められる．

非機能面に関する製品間相違性も存在する．機器の設置場所によって異なる必要通信容量を満足したり，製品コストを下げるために，ハードウェアが変更され，それに伴ってソフトウェアも変更される．□

最初の 2 製品については限られた予算と期限内にどうにかして製品開発を行わなければならない状況であった．顧客からのコスト削減と納期短縮の要求は極めて厳しかったが，製品ベースの従来の再利用による

表 1　プロジェクト概要

開発プロダクト	モバイルネットワークシステム
開発タイプ	システムファームウェア
開発言語	C 言語／アセンブリ言語
開発規模	数百 KLine 以上（詳細非公開）
開発工期	1 年以上（詳細非公開）
開発人員	平均約 100 名（最大約 300 名）
開発体制	地理的に離れた 4 拠点での開発

コスト削減はすでに限界に近く，製品群を俯瞰して成果物を資産化し，全体最適化したかたちで再利用を図るほかにコストを削減する術がなかった．幸いなことに最初の 2 製品のリリースは決まっており仕様も確定していた．PLE 開発の初期の成功事例として宣伝される CelsiusTech のケーススタディ[4] と似るこのような状況は，プロジェクトへの PLE 導入を強く動機づけることとなった．

2.2　PLE 導入の動機と導入着手時の開発現場の状況

プロジェクトへの PLE 導入は図 1 のスケジュールに沿って進められた．

本プロジェクトは，富士通 QNET が PLE を全社的に導入する過程において，最初に PLE を導入する実プロジェクトであった．プロジェクト開始時点に，全社的なプロダクトライン開発手法導入ガイドライン，ならびに製品間共通性／相違性解析のためのフィーチャモデリング実施ガイドライン等のガイドラインは用意されていたが，これらのガイドラインはパイロットプロジェクトや文献調査を通して作成されたものであり，実プロジェクトで運用されたことはなく，継続的な改善が必要なものであった．そのためプロダクト開発に並行して，各開発工程で発見されたノウハウをガイドラインにフィードバックし，ガイドラインの改善を進めていった．

開発プロセスについても PLE への対応が必要であった．富士通 QNET は CMMI レベル 3 を達成しており，開発タイプごとに標準的な開発プロセスを定義しているが，これらの開発プロセスは PLE には対応していなかった．そのため開発プロセスについても，プロダクト開発に並行して，PLE 対応できるように拡張，改善していく必要があった．

3.　PLE 開発実施のための開発組織と開発プロセス

本節では，PLE 開発実施にあたっての組織構成，開発チームの教育，標準開発プロセスの PLE 拡張について述べる．

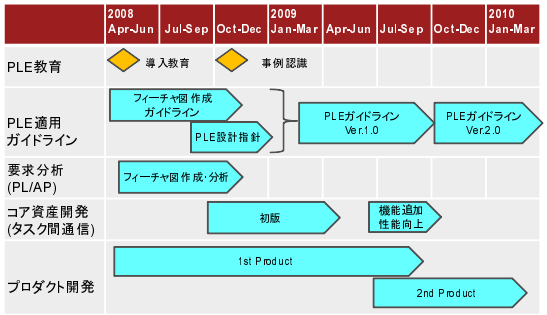

図 1　PLE 導入スケジュール

3.1　開発組織の構成

PLE の重要概念として，製品間で共有するコア資産の開発に携わるドメインエンジニアリングとコア資産を再利用して個別の製品を開発するアプリケーションエンジニアリングの役割上の分離がある．本プロジェクトにおいても，当然，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は分離することとし，ドメインエンジニアリングチームは移動体ネットワーク機器プロダクトライン全体の共通性／相違性解析（フィーチャ分析），コア資産のスコーピング，アーキテクチャの決定に携わる役割を担い，アプリケーションエンジニアリングチームはドメインエンジニアリングチームの成果に基づいて設計以降のシステム開発する役割を担うこととした．

しかしながら，最初からプロジェクト内の技術者をドメインエンジニアリングチームとアプリケーションエンジニアリングチームに明確に分離していたわけではない．本プロジェクトでは，従前のプロジェクト通り，開発組織はシステムの機能部ごとにサブチームに分割されていた．PLE 導入前には，システムにおいて中心的な 7 つの機能を担当する各々サブチームにおいて，技術リーダを含む 3 名，合計約 20 名の技術者に PLE の学習を求めた．PLE 開発着手後の分析段階では，各機能に精通するこれら約 20 名の技術者が製品間共通性／相違性分析，アーキテクチャ分析，リファクタリング分析を行った．ドメインエンジニアリングチームとアプリケーションエンジニアリングチームは未分化であり，少数の技術者がプロダクトライン全体の分析に携わる，いわばドメインエンジニアリングチームだけがあるような状態であった．設計工程以降は，この 20 名の中の 1 人と新たに雇い入れた 7 名の技術者とで，少数精鋭型の独立したドメインエンジニアリングチームを組織した．他の技術者はアプリケーションエンジニアリングに回った．図 2 に工程ごとのチーム編成の変化を示す．

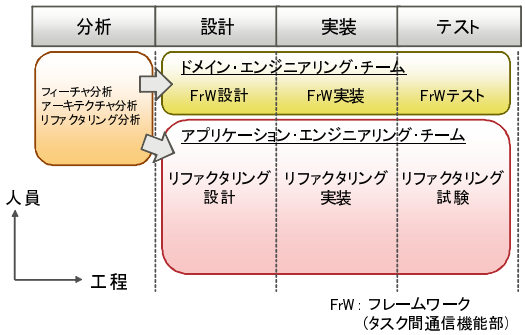

図 2 開発人員構成

分析工程においてドメインエンジニアリングチームとして製品間共通性／相違性分析，アーキテクチャ分析，リファクタリング分析に携わった技術者が，設計工程以降においてアプリケーションエンジニアリングチームに回ったことは，両チームにおいて製品間共通性／相違性に対する認識を共有し，コア資産を構築ならびに再利用するうえで好ましい結果をもたらした．

3.2 開発チームの教育

本プロジェクトのような大規模プロジェクトでは，PLE への理解を深めるための教育を提供し，PLE の思想と手法をプロジェクト内の技術者の末端にまで浸透させる必要があった．

以下，ドメインエンジニアリングに携わる技術者とアプリケーションエンジニアリングに携わる技術者に提供した教育内容を示す．最初に行ったのは導入教育であり，ここでは特に PLE 適用による効果を中心に説明し，新規開発手法適用に対する心理的障壁を取り除くことに注力した．ドメインエンジニアリングについてはその役割上，PLE 開発についての充分な理解が必要なため，PLE の概要に加え，社内外の事例を通した PLE 導入の諸課題の認識，コア資産のスコーピング，具体的な取組み方法に関する学習と情報交換を行った．

- ドメインエンジニアリング（コア資産開発）：
  - PLE 導入教育：開発技術に絞ったフィーチャ分析や二層開発体制などの PLE 概論
  - 社内外の PLE 事例認識： 社外での PLE 取組み動向と社内 PLE 実践事例の認識
  - 外部講師を招いた講演会や技術交流会
- アプリケーションエンジニアリング（製品開発）：
  - PLE 導入教育：開発技術に絞ったフィーチャ分析や二層開発体制などの PLE 概論

3.3 標準開発プロセスの PLE 拡張

従来の開発プロセスを大きく変えることは開発プロジェクト全体への影響が大きく，PLE 導入の大きな障壁となりかねない．富士通 QNET は CMMI に取り組んでおり，開発タイプごとに標準的な開発プロセスを定義しているが，プロジェクト開始時点ではこれらのプロセスは PLE に対応していなかった．開発プロジェクトへの開発プロセス変更の影響を極力減らすため，社に定着している CMMI の標準プロセスを徐々に PLE 対応に拡張することにした．

1 製品目の開発では，システムファーム開発用に定義している CMMI 準拠の標準プロセスを用い，このプロセスから外れない形で PLE 開発を進めた．たとえば，当初より重要視されていた製品間の共通性／相違性分析は定義済みの分析工程の中で実施された．2 製品目の開発では，このプロセスにスコーピング，フィーチャ分析，コア資産計画等の PLE 独特の工程を追加し PLE 対応とした．

結果，PLE 導入による開発プロセスへの影響を比較的小さくおさえることができた．表 2 に WBS の数による各工程での影響を示す．表が示すように，228 個の標準プロセスに対して，PLE で追加されたプロセスは 12 個と全体の 5%となった．このように PLE 固有のプロセスは工程全体の一部であり，ほとんどのプロセスを既存の開発プロセスであった．

表 2 開発プロセス毎の WBS 数

プロセス		WBS の数		
		(1) 標準プロセス数	(2)PLEで追加されたプロセス	(2) のうちテーラリング可能なプロセス
要求管理		28	3	2
エンジニアリング	分析	21	2	1
	設計	37	2	0
	実装	57	3	1
	テスト	85	2	0

開発プロセスとあわせて，各種の開発規約についても PLE に対応すべく再整備する必要があった．たとえば，アプリケーションエンジニアリングにおいてコア資産の不用意な変更を防止すべく開発規約の再整備がなされた．

4. 製品間共通性／相違性の分析

本プロジェクトでは，移動体ネットワーク機器プロダクトラインの製品間の共通性／相違性分析において，システム全体にわたる大規模なフィーチャモデリングを実施した．本節では，フィーチャモデリングの詳細について述べる．

4.1 フィーチャモデリングの導入

富士通 QNET では，製品間の共通性と相違性の分析に，社の複数の異なる開発タイプの開発現場において，効果的であるとの声が高かったフィーチャ分析を用いている．

フィーチャ分析は，Kang によるドメイン分析手法 Feature Oriented Domain Analysis(FODA)[5] であり，製品間の共通性と相違性を各製品が備えるフィーチャの相違として離散的に捉え，あわせて各製品におけるフィーチャの選択上の制約，ならびにフィーチャ間の意味的依存関係を分析し，フィーチャモデルと呼ばれる樹形決定木で製品間の共通性と相違性を表現する手法である．

富士通 QNET では，フィーチャモデリングに際して，同じ Kang らによる FODA に基づくプロダクトライン開発手法 Feature Oriented Reuse Method (FORM)[6] の 4 階層モデルも導入している．FORM では，フィーチャモデルを上から Capability（能力）層，Operating Environment（動作環境）層，Domain Technology（ドメイン技術）層，Implementation Technique（実装技術）層の 4 階層に分け，フィーチャを階層状に配置する．Capability 層には外部から見えるサービスや動作，非機能要件に関係するフィーチャを，Operating Environment 層にはソフトウェアやハードウェアのインタフェースやプラットホームに関係するフィーチャを，Domain Technology 層にはドメイン固有の概念や標準に関係するフィーチャを，Implementation Technique 層にはドメイン非依存の実装手段やアーキテクチャスタイル，算術アルゴリズム等に関係するフィーチャを配置する．このように階層状に配置されたフィーチャによって，各フィーチャを担当するモジュールの階層関係を開発の初期段階でおおよそ決めることが可能となる．

4.2 フィーチャモデリングの実施

本プロジェクトでは，移動体ネットワーク機器プロダクトライン全体に対してフィーチャモデリングを実施した．当該プロダクトラインの製品は，製品出荷後 10 年以上の長期にわたって顧客に利用され続け，その間再利用を前提としたエンハンス開発が繰り返される．そのため，プロダクトラインの全体像を把握したうえで，コア資産化の候補となるフィーチャをより多く抽出し，計画的かつ効率的なコア資産開発を継続していく必要がある．システム全体のフィーチャモデルを構築し，フィーチャの直交的な分解と類似フィーチャの整理を進めていくことで，

- プロダクトライン全体に必要な機能・要件仕様の整理
- 共通上位モジュールの発掘とインターフェースの整理

が可能となる．フィーチャモデリングのコストは従前の個別開発方式とくらべて追加となるものの，本プロジェクトではこれらのフィーチャモデリングの効用に，より多くの利点を認めた．

プロダクトライン全体の大規模なフィーチャモデリングを実施すべく，開発チームは当該プロダクトラインの機能部ごとに分かれてフィーチャモデリングを実施し，その後各機能部のフィーチャモデルを統合し，システム全体のフィーチャモデルを構築することとした．システムの各機能部を担当するサブチームの 3 名，全体で約 20 名の技術者がフィーチャモデリングに携わった．本プロジェクトに先立つパイロットプロジェクトにおいても認識されていた問題であるが，成果物としてあがってくるフィーチャモデルはモデリング実施者の考え方やスキルに大きく依存し，フィーチャモデリングの属人性が改めて問題として認識された．

- Capability 層フィーチャをどこまで細分化すればよいかあいまいであったこと
- 作成者によってフィーチャの階層化判断が異なること

が属人性の主たる原因として分析された．そこで，フィーチャモデリング実施者全員が集まった集中検討会を開催し，対面で徹底的に議論し，記載内容の統一を図った．

後のフィーチャモデルの改版でも一貫性と統一性を維持することができるようにすべく，議論の結果新たに生まれたルールは，プロジェクト開始時に用意されていたフィーチャモデリング実施ガイドラインに反映し，同ガイドラインを実務的，かつ完成度の高いものに改善していった．また，社外の専門家を招いてフィーチャモデルのレビューを実施し記載内容の充実を図った．

フィーチャモデリングにはのべ約 200 時間を要した．ドメインエンジニアリングチームはアプリケーションエンジニアリングチームを顧客と見立て，2 製品目のハードウェア構成の変更を考慮に入れつつ，求められる機能と性能の相違をフィーチャモデル上で整理していった．前述の通り，分析工程でフィーチャモデリングに参加した技術者は，設計工程以降でドメインエンジニアリングチームとアプリケーションエンジニアリングチームに分かれた．そのため製品間共通性／相違性に関する認識を両チームで効果的に共有することができた．

5. コア資産のスコーピング

本プロジェクトでは，プロダクトラインレベルの再利用と，従来通りの製品レベルの再利用が混在するかたちで開発が行われた．プロダクトラインレベルの再利用とは，ドメインエンジニアリングチームが開発，保守するコア資産を，アプリケーションエンジニアリングが規定された通りの方法で再利用することをいう．一方，製品レベルの再利用とは各プロダクトにおいて，アプリケーションエンジニアリングチームの裁量において過去の資産を再利用することをいう．

前述の製品間の共通性／相違性分析によって抽出されたフィーチャは 2,000 以上であった．しかし，大規模プロダクトラインにおいて全フィーチャをコア資産化するのはそもそも非現実的であること，開発メンバの PLE 習熟度から考えて大人数のドメインエンジニアリングチームの組織は困難であること，初期コスト増の制限と納期が厳しいことから，最大の効果が期待できるフィーチャに限ってコア資産化する戦略を採ることとした．

このコア資産のスコーピングは FORM の４つの階層に基づいて実施することとした．移動体ネットワーク機器プロダクトラインにおける，各階層におけるフィーチャの総数と可変フィーチャ数（Optional または Alternative なフィーチャの数）を表 3 に示す．また，本表は後述する Operating Environment 層への新規配置は含まない．

表 3 FORM 各層におけるフィーチャ数

	フィーチャ数	可変フィーチャ数
Capability 層	289	20
Operating Environment 層	131	23
Domain Technology 層	681	74
Implementation Technique 層	911	78
合計	2,012	195

以下，FORM の 4 層モデルの各層ごとに，フィーチャをコア資産のスコープに含めるか否か，本プロジェクトにおいてどのように意思決定したのかをまとめる．

Capability 層：製品のサービスや特徴に関係するこの層のフィーチャによって，システム仕様書やシステムテストの抽象度に相当する製品間相違性がよく理解されるようになった．しかし，コア資産化の労力の割に品質，コスト，工期への改善インパクトが期待できなかったこと，予算，要員，納期の制約が厳しかったことから，コア資産のスコープ外とした．□

Operating Environment 層：OS やミドルウェアに関係するこの層のフィーチャは，再利用可能な部品として社で既に実装，保守されており，これまでも製品レベルの再利用が効率的に行われていた．改めてコア資産化するメリットを見いだせないため，原則的には本プロダクトラインのコア資産のスコープ外とした．ただし，後述の Implementation Technique 層の分析の後，2 製品間の相違性を隠蔽するためのフィーチャがこの層に追加され，それについては例外的にコア資産のスコープとされた．□

Domain Technology 層：技術標準により規定されている処理，具体的には通信プロトコル処理，優先制御，キュー管理機能等が大部分を占めるこの層のフィーチャは，Operating Environment 層同様，再利用可能な部品として社の既有資産中に存在していた．今回の開発では納期やドメインエンジニアリングにつぎ込める技術者と予算が限られ，また初めての PLE 開発で練度も決して高いわけではなかったため，コア資産化については急がないこととした．PLE 開発を実施するのは設計工程まで，つまり設計工程までの成果物をコア資産化することとした．既有再利用可能部品については，ドメインエンジニアリングチームではなく，アプリケーションエンジニアリングチームによって将来のコア資産化に向けてリファクタリングを施すことにした．□

Implementation Technique 層：2 製品目で予定されているハードウェアの変更によって生じる 1 製品目との製品間相違性について分析を行った．ハードウェア実装機能の相違性や，性能，帯域などの要件に起因する実装差などを整理，分析する中で，タスク間通信機能の製品間相違性を隠蔽し，各プロダクトから共通的に利用できるインタフェースを設ければ，アプリケーションの独立性が確保でき，2 製品目での開発資産の流用率を大幅に向上できることがわかった．しかしながら，このタスク間通信には広帯域が必要であり，既存の OS やライブラリで提供されている機能でその性能要件を満たすことは不可能であった．そのため，新規に独自のタスク間通信機能を実装する必要があった．タスク間通信機能に関するフィーチャは 129 あったが，これらフィーチャに関わる製品間相違性を隠蔽するタスク間通信機能のソースコード生成フィーチャを Implementation Technique 層ではなく，Operating Environment 層に新規に配置し，コア資産のスコープとした．□

プロダクトラインの定義から，Domain Technology 層フィーチャがコア資産化の優先度が最も高くなるも

のと予見していたが，本プロジェクトでは品質，コスト，工期の改善を最大化する観点から，Implementation Technique 層フィーチャのコア資産化にもっとも優先度が置かれることとなった．

6. コア資産開発

コア資産のスコーピング結果に基づいてコア資産開発に着手した．すでに述べているとおり，8 名で構成される少数精鋭型のドメインエンジニアリングチームがタスク間通信機能関連のフィーチャを実現するフレームワークを開発し，コア資産とすることになった．一般的にフレームワーク設計を行う場合，インタフェースや処理の共通化を優先するために組込みファームウェアにおいて重要となる処理性能が犠牲にされがちである．このプロダクトラインでは，マルチコアプロセッサ上での並列処理を行うが，タスクのコアへの割当とタスク間通信のための一般的な仕組みを，性能を犠牲にすることなく設けることが困難であった．特に，タスクのコアへの割当は製品によって異なり，また製品のチューンアップの間に頻繁に変更される可能性があった．そこで，本プロジェクトではソースコード自動生成手法を用いてタスク間通信に高度の柔軟性を与え，目標性能を満たす戦略を採った．

パイプラインハザードはタスク間通信の性能に大きな影響を与えるため，条件分岐を可能な限り除いてパイプラインハザードを避けなければならない．本プロジェクトでは，タスク間通信における条件判断を実行時の判断とコンパイル時の判断に分離した．実行時の条件判断は本質的に除くことができないが，コンパイル時の条件判断はソースコード生成時に除くことができる．ドメインエンジニアリングチームは，パラメータテーブルに従い，最小限の条件分岐で高いデータ転送性能を保つようなソースコードを生成するソースコード自動生成器を開発した．

フレーム開発において，ドメインエンジニアリングチームはフィーチャモデルを参考に，アプリケーションエンジニアリングチームとレビューを重ねて部品としてのインターフェース仕様を策定した．アプリケーションエンジニアチームは，製品要求に従って，プロセッサコアへのタスク割当，送受信タスクの論理定義を記述したタスク間通信コンフィギュレーションを記述する．ソースコード自動生成器は，コンフィギュレーション記述に従って，論理／物理アドレスマップ，受信バッファの実体宣言とアドレス配置，転送先アドレス定義等を記述したパラメータテーブルを含む，タスク間通信機能ソースコードを生成する．

本フレームワーク自体が，今回の PLE 導入を成功に導いた最も有益なコア資産となり，品質，コスト，工期面での大幅な改善に貢献した．また，今回は Domain Technology 層のフィーチャのコア資産化を見送ったが，将来のコア資産化に向けて，アプリケーションエンジニアリングチームは関係する既存資産のリファクタリングを並行して行った．

7. 見積りと実績，考察

本節では，本プロジェクトにおける品質，コスト，工期のいわゆる QCD の 3 点における見積りと実績について述べ，本プロジェクトの結果に対する考察を示す．

7.1 品質，コスト，工期面での見積りと実績

本プロジェクトでは，製品間の共通性／相違性分析をシステム全体に対して行ったが，設計工程以降で PLE を適用したのはタスク間通信機能に限定している．それにも関わらず，従来の製品個別開発と比して，QCD いずれの面でも大幅な改善を見ることができた．これらの改善はタスク間通信機能のソースコード自動生成に依るところが大きい．

品質：これまで類似製品の開発において 2 製品目で生じる不具合の数は 1 製品目のおおよそ 1/3 程度であった．ソースコード自動生成器が生成したタスク間通信機能のソースコードにおける不具合の数は，2 製品目において 1 製品目の 1/10 となり，PLE 適用部分では品質面での大幅な向上が見られた．ソースコード自動生成器自身の不具合はわずかであり，生じた不具合のほとんどはコンフィギュレーションファイルの記述ミスによるものであった．

PLE 適用対象外の部分も含めたシステム全体を見た場合，PLE 適用箇所が限定的であったため，2 製品目で生じた不具合の数は 1 製品目のおおよそ 1/3 であり，従来とそれほどの差は生じなかった．

コスト：タスク間通信機能を PLE 開発するに先立って，PLE を適用した場合と適用しなかった場合のタスク間通信機能の開発コスト比較を行った．図 3 は見積値による両者の比較を示している．PLE を適用していない部分，すなわちタスク間通信機能に直接関係しないフィーチャの開発コストはグラフに含まれていない．

タスク間通信機能を PLE 開発しなかった場合，すなわち製品毎に開発を行った場合の第 n 製品を出すまでのタスク間通信機能の累積開発コスト $TC_{NPL}(n)$ は，共通機能開発コストを CC，第 i 製品の個別機能開発コストを CV_i，開発製品数を n とすると次式で

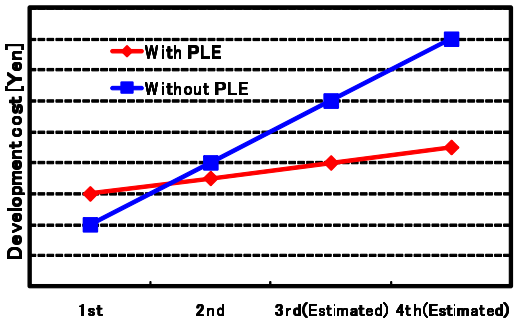

図 3 PLE 適用有無と開発コストの関係

表される．

$$TC_{NPL}(n) = CC + \sum_{k=1}^{n} CV_k$$

通信用バッファ制御等の製品間で共通に使用できる機能が共通機能，異なるハードウェアの制御やタスクのコアへの割当など製品によって変更が必要となる機能が個別機能に含まれている．経験的にタスク間通信機能はその特性上，製品個別開発量がほぼ同じである．ゆえに，近似的に $CV_1 = CV_2 = \cdots = CV_n$ であり，これを CV と定義すれば，上の式は以下のように表せる．

$$TC_{NPL}(n) = CC + CV \times n$$

PLE を適用してソースコード自動生成器を用いた場合，初期コストはかかるものの製品個別開発量は削減することができる．ソースコード自動生成器の開発コストの見積値を CC_{gen}，ソースコード自動生成器が生成するソースコードを人手で開発した場合に必要なコストの見積値を CV_{man}，タスク間通信コンフィギュレーションファイルの作成コストの見積値を CV_{conf} とすれば，第 n 製品を出すまでのタスク間通信機能の累積開発コスト TC_{PL} は次式で表される．

$$TC_{PL}(n) = CC + CC_{gen} + (CV - CV_{man} + CV_{conf}) \times n$$

なお，$CV - CV_{man}$ はソースコード自動生成を行ってもなお製品個別の設定が必要なハードウェア設定等のコード記述に関するコストに相当する．

PLE 適用による損益分岐点，すなわち $TC_{PL} < TC_{NPL}$ を満足する最小の整数 n_{BE} は以下のようになる．

$$n_{BE} = \lceil \frac{CC_{gen}}{CV_{man} - CV_{conf}} \rceil$$

CC_{gen}，CC_{man}，CC_{conf} に見積値を代入したところ，$n = 1.67$，すなわち $n_{BE} = 2$ と算出され，2 製品目で黒字転換が期待できるものと見積もられたため，PLE 開発を実施することとした．実績値は見積値と大差なく，実際に 2 製品目で初期投資の回収を済ませ，利益を出すことができた．

PLE 適用対象外の部分も含めたシステム全体での開発コストについては，2 製品目でアプリケーションの仕様変更や機能追加が生じているため比較評価できていない．主観的な評価となるが，フィーチャモデリング等の PLE 導入による追加コスト分は，製品間共通性／相違性の可視性向上に伴う品質向上，製品間の共通部の発見，発掘によるコスト削減によって十分に回収できていると認識している．

工期：タスク間通信機能をコア資産化し PLE 開発を実施した結果，2 製品目の開発工期を PLE 開発を実施しなかった場合の見積値の 1/4 に開発工期を短縮することができた．

PLE 適用対象外の部分を含めたシステム全体での開発工期については，プロダクトライン全体のフィーチャモデリングを実施したため，1 製品目の開発工期が従来通りの開発を行った場合の見積りに対して 3%ほど延びている．ドメインエンジニアリングチームを組織してアプリケーション開発と並行してコア資産開発を実施したため，コア資産開発によって開発工期が延びることはなかった．

2 製品目の開発工期は 1 製品目の開発工期の約 40%となった．これまで経験的に 2 製品目の開発工期は 1 製品目の約 70%であったため相当の工期短縮がなされている．工期短縮にはさまざまな要因が関係しており，短縮された約 60%のうち，PLE 開発がもたらした短縮と PLE 開発以外の要因がもたらした短縮とを分離して評価するのは困難である．PLE 開発がもたらした工期短縮は感覚的には約 60%のうちの 10 数%相当分である．

PLE 開発による工期短縮は，システム全体レベルのフィーチャモデリングによる全体理解とタスク間通信機能の自動生成によってもたらされた．工期短縮をもたらした PLE 開発以外の要因としては，1 製品目の開発メンバのほとんどが 2 製品目を担当したことが挙げられる．1 製品目の仕様，機能，ソースコードの理解など新メンバがプロジェクトに参入する際にかかる工数，いわゆる立上げ工数が不要となった．また，2 製品目における改変の際に，改変の影響範囲が把握できていたためバグが作り込まれることもなかった．もうひとつの要因としては，1 製品目と 2 製品目の要求仕様の差が通常のプロジェクトよりも小さかったことが挙げられる．ハードウェア変更に伴うファームウェ

アの大幅な作り替え，タスクの再配置が発生したものの，上流工程の仕様書や評価に用いるテストデータの多くを再利用できた．

7.2 フィーチャモデリングの効果

本プロジェクトでは，ドメインエンジニアリングのコスト増になるものの，プロダクトライン全体に及ぶ大規模なフィーチャモデリングを実施している．フィーチャモデル自体が最終成果物たる製品の製造に直結することはなく，またフィーチャモデリング，特にフィーチャの意味を考え階層構造に構成していく作業には相当の時間を要する．そのため，PLE 導入時には，フィーチャモデリングに要する経済的，時間的コストを負いきれるのかが常に問題になる．本プロジェクトにおいても，フィーチャモデリングはプロジェクト初期のコスト増の要因になったが，フィーチャモデリングの成果はその後の QCD 改善に大きく役立った．

フィーチャモデリングの利点は，製品間共通性／相違性の大局的把握が可能となり，それに伴って開発時の意思決定が適切に下せるようになるところにある．

外部に見える機能の製品間相違性については，これまでの開発でもドキュメントによって明確に認識されていた．一方，構造や機能分割など内部仕様上の製品間相違性については，各技術者が過去の製品開発経験に基づく見識を暗黙的に持っているに過ぎず，それが明示的に表現され整理されることはなかった．そのため製品の変化への備えは属人的かつ局所的になされがちであった．今回，4 階層のフィーチャモデリングを実施した結果，機能やサービスといった外部仕様上の製品間相違性が，OS やミドルウェアなどの動作環境，通信標準などのドメイン技術，ハードウェアやソフトウェアなどの実装技術と明確に関係づけられ，それがプロジェクトの関係者全員で認識できるようになった．異なる拠点またはチームによって個別に開発される機能部を越えて，共通の要素，異なる要素が理解されるようになり，製品内で再利用される部品の共通化が大きく進んだ．

さらに，プロダクトライン全体のフィーチャモデルを作成した結果，PLE 適用効果を期待できる部分，すなわち製品間の変化が大きく，かつ開発したコア資産によってその変化を吸収しやすい箇所（本プロジェクトではタスク間通信部）を絞ることが容易になった．

7.3 得られた知見

本プロジェクトを通して得られた知見を以下にまとめたい．

小さなスタート：1 製品目の開発成功がプロダクト群開発継続の必須条件であることが多いため，PLE 開発の練度が上がるまでは，コア資産のスコープをむやみに広げてはならない．まずは確実な成功を体験した後，その後，コア資産化率を向上させることが成功の鍵であると言える．

スコープの厳選：プロダクト毎に受託して開発する設計開発会社は，マーケティング計画に基づいたプロダクトラインを見越した先行投資が困難であるという，PLE 開発実施上の宿命的な難点を持つ．そうであっても，PLE 開発の初期投資を早期に回収すべく，リリースが約束されている少数製品を対象に，PLE の適用効果が早くかつ高く現れるフィーチャにコア資産のスコープを絞り，PLE を限定的に適用するだけでも品質，コスト，工期の改善は可能である．

チーム全体の意識改革：PLE 開発を成功させるためには，仕組みを作るだけではなく，開発メンバの意識を変えることが極めて重要である．PLE 導入当初，経営層は PLE 導入に対して楽観的，開発メンバは悲観的であった．開発メンバは目の前の製品開発で精一杯であり，本当に開発するか不透明な先のプロダクトのことまでを考えるゆとりがない．流用開発で充分だと思っている開発メンバには経営層やプロジェクトリーダの思いはなかなか届かない．事前教育を通して PLE のパラダイムを開発メンバに十分に浸透させておいたうえで，さらにプロジェクトリーダには信念を持って粘り強くメンバを説得するタフさが求められる．

8. 関 連 研 究

リリースの約束されている少数の製品に対して PLE 開発を導入している事例としては，CelsiusTech の艦船制御プロダクトラインの古典的事例[4] が挙げられる．

本事例のように，システム全体のフィーチャモデリングを最初に実施し，フィーチャモデルをソフトウェアの再利用性と保守性向上の手引きとする事例として，Kang らによるクレジットカード認証システムでの事例[7] がある．同事例では，フィーチャモデル上でクレジットカード認証システムで要求の変更が生じやすい箇所を特定し，変更に強いかたちにコンポーネントの階層構造を整理している．

本事例同様，少数精鋭型のチームを組織して製品間共通性／相違性分析を実施し，PLE を導入した事例としては，日立製作所と日立ハイテクノロジーズの事例[8] がある．本事例では共通性／相違性分析以降，独立したドメインエンジニアリングチームでコア資産を開発したが，同事例ではドメインエンジニアリングチームを分離せず，コア資産開発を製品開発チームに割り振るかたちで PLE 開発を実施している点で異

なる．

9. ま と め

以上，本稿では，マーケティング計画に基づいたPLE 実施が困難な設計開発専門会社において，移動体ネットワーク機器プロダクトラインのファームウェアを大規模 PLE 開発した成功事例を報告した．

プロジェクト実施に先立ち，PLE の学習と各種ガイドラインの整備を全社的に行い，新手法導入に関する技術的，心理的障壁の逓減を図った．プロダクトラインの全体像を把握すべくフィーチャモデリングを実施し，2,000 を越えるフィーチャを見つけ出した．このような大規模な PLE において，全フィーチャをコア資産とすることはコスト，納期の点で不可能であったため，FORM4 階層モデルの階層ごとにコア資産のスコープに入れるフィーチャを厳選した．スコーピングの結果，タスク間通信機能に関係するフィーチャのみコア資産を開発することとしたが，部分的な PLE の導入にもかかわらず，従来の派生開発をとった場合を下回る累積コストで 2 製品目の開発を成し遂げた．PLE 非適用部も含めた，システム全体での品質については従来の再利用開発と変わらなかったが，工期については従来の再利用開発を大きく上回る短縮を実現した．もっともこれらの改善は PLE 非適用部での改善に依るところも少なくない．しかし，ソースコード自動生成手法を導入しタスク間通信機能部のコード生成を行ったことが，PLE 適用部の品質，コスト，工期の改善に大きく寄与し，さらには PLE 非適用部も含めたシステム全体の改善を牽引した．

大規模開発では，立上げ時にはより早くより高い効果が見込める個所に PLE 開発を限定的に適用し，コア資産の肥大化と複雑化を避け，その後，チームの練度にあわせて徐々にコア資産を膨らませていく戦略を推奨したい．また，PLE の導入を成功さえるには，プロセスや組織の漸進的改革，開発チームの意識改革，信念を持ったプロジェクトリーダによるリーダシップが不可欠である．

今後の取り組みとしては，今回の PLE 適用や各種教育によって開発チームの PLE 開発の習熟度が上がってきたことを踏まえて，3 製品目以降の開発におけるコア資産の拡充，すなわち PLE 適用範囲の拡大が挙げられる．具体的には Domain Technology 層フィーチャのコア資産化を進めるため，各開発工程での生産物の構成管理方法の追記等，ガイドラインの充実を図る予定である．

参 考 文 献

1) P. Clements and L. Northrop, *Software Product Lines: Practice and Patterns, Addison-Wesley*, 2001.
2) T. Iwasaki, M. Uchiba, J. Ohtsuka, K. Hachiya, T. Nakanishi, K. Hisazumi and A. Fukuda, “An Experience Report of Introducing Product Line Engineering across the Board,” *Proc. Software Product Line Conf. (SPLC) 2010*, Vol.2, pp.255–258, Sep. 2010.
3) 岩崎 孝司, 内場 誠, 大塚 潤, 八谷 浩二, 中西 恒夫, 久住 憲嗣, 福田 晃, 「プロダクトライン開発手法の全社的導入に関する一事例報告」, 組込みシステムシンポジウム 2010（ESS2010）論文集, pp.125–130, Oct. 2010.
4) L. Brownsword and P. Clements, “A Case Study in Successful Prooduct Line Development,” CMU/SEI-96-TR-016, Technical Report, Software Engineering Institute, Carnegie Mellon University, Sep. 1996.
5) K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, “Feature-Oriented Domain Analysis (FODA) Feasibility Study,” Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Nov. 1990.
6) K. C. Kang, Jaejoon Lee, and Patrick Donohoe, “Feature-Oriented Product Line Engineering,” *IEEE Software*, Vol.9, No.4, pp.58–65, July/August 2002.
7) K. C. Kang, J. J. Lee, B. Kim, M. Kim, C. W. Seo, and S. L. Yu, “Re-engineering a Credit Card Authorization System for Maintainability and Reusability of Components: A Case Study,” *Proc. 9th Int. Conf. on Software Reuse (ICSR2006)*, pp.156–169, Jun. 2006.
8) Y. Takebe, N. Fukaya, M. Chikahisa, T. Hanawa, and O. Shirai, “Experiences with Software Product Line Engineering in Product Development Oriented Organization,” *Proc. Software Product Line Conf. (SPLC) 2009*, pp.275–283, Sep. 2009.