

プロダクトライン開発向け開発資産表現FRMモデルの 提案

服部, 勇祐
九州大学大学院 システム情報科学府情報工学専攻博士後期課程

平川, 剛
株式会社ネットワーク応用技術研究所

芦原, 秀一
株式会社ネットワーク応用技術研究所

中西, 恒夫
九州大学大学院システム情報科学研究院情報知能工学部門

他

<https://doi.org/10.15017/17892>

出版情報：九州大学大学院システム情報科学紀要. 15 (1), pp.37-42, 2010-03-26. 九州大学大学院システム情報科学研究院

バージョン：

権利関係：



プロダクトライン開発向け開発資産表現 FRM モデルの提案

服部勇祐* · 平川 剛** · 芦原秀一**
中西恒夫*** · 久住憲嗣† · 福田 晃***

FRM Model: An Asset Representation for Software Product Lines

Yusuke HATTORI, Go HIRAKAWA, Shuichi ASHIHARA,
Tsuneo NAKANISHI, Kenji HISAZUMI and Akira FUKUDA

(Received December 11, 2009)

Abstract: Product line (PL) is a development method eases software development for more kind's software by requirement from the market. In PL development, commonalities and variabilities between each product are represented as features mean characteristics and functions. We proposed Reusable Asset Specification (RAS) based representation of core assets for the PL. However, RAS has development asset information and relation information between features and development assets. It is convenient to develop this information separately. We propose a FRM model for development assets in the PL. This model consists of feature model and divided RAS. Furthermore, we developed a tool which assist software development with FRM model, and show a case study of ECU development with FRM model.

Keywords: Product line, Software engineering, Asset, Representation, Model, RAS

1. はじめに

市場における顧客からの多様な要求に応えるべく、多品種開発のための開発方法論、プロダクトライン開発方法論¹⁾が注目されている。プロダクトライン開発方法論では、特定の市場領域向けの製品(プロダクト)を複数品種開発する際に、それらプロダクト間の共通性と相違性を明らかにした上で共有の開発資産を構築する。そして、事前に決められたプロセスでそれら共有開発資産を再利用することで高品質、低コスト、短納期での複数プロダクトの並行開発を図る。

プロダクトライン開発では多くのプロダクトを扱うために、プロダクト間の相違性の識別が重要である。この識別に用いられるのがフィーチャモデル⁶⁾である。フィーチャとは、顧客や開発者が認識できるプロダクトの機能や特性を表す抽象概念であり、プロダクトライン中のプロダクト間の相違はそれぞれのプロダクトが備えるフィーチャによって識別される。

所望のプロダクトの機能や性質をフィーチャによって定義し、共有の開発資産からプロダクトを導出するためには、フィーチャと共有開発資産の間のトレーサビリティを確保しなければならない。また、プロダクトライン開発方法論における共有開発資産は、コードのみならず、

ユースケース等の要求からクラス図等の構造モデル、シーケンス図等の振舞モデルなど異なる抽象度、異なる観点、異なる表現の文書を含む。フィーチャから関連する文書の全体あるいは一部へのトレーサビリティを実現するためにはツールの支援が欠かせない。そうしたツールにおいて多様な表現形式の文書を包括的に取り扱う仕組みとして、筆者らはすでにRASを用いたプロダクトライン開発資産表現を提案している⁹⁾。

RAS⁵⁾はOMGが標準化した再利用可能開発資産の定義であり、多様な資産を包括的に扱うことができる。このRASとフィーチャを組み合わせて表現することにより、フィーチャを含めたプロダクトライン開発における統一的な資産表現を実現した。

しかし、RASはフィーチャから各種表現の開発資産へのトレーサビリティリンクを設ける目的においては過剰な仕様となっている。プロダクトライン開発においてはRASのサブセットで十分表現可能である。また、フィーチャと開発資産間のトレーサビリティを表現するためにはフィーチャ、開発資産、フィーチャと開発資産間の関連の3つの要素が必要となる。元の提案手法ではこのうち開発資産と関連がRASで表現されることとなる。しかし、関連はフィーチャと開発資産が存在しないと作成できず、関連を作成するまではRASが不完全な状態で存在することとなる。不完全なRASに対してはモデルの要素の漏れ抜けやモデル間の整合性の検証等を行うことが難しい。

そこで、RASのうち開発資産を表現する部分と関連を表現する部分を抽出し、開発資産を表現する部分と関連

平成21年12月11日受付

* 情報工学専攻博士後期課程

** 株式会社ネットワーク応用技術研究所

*** 情報知能工学部門

† 九州大学システムLSI研究センター

を表現する部分をそれぞれ独立したモデルとすることにより、それぞれのモデルに対する種々の検証を可能にし、トレーサビリティリンク構築時の効率向上を図る。

本稿ではフィーチャ、フィーチャと開発資産間の関連、開発資産を表現するモデルをそれぞれ、Feature Model, Relation Model, Material Modelとし、それらを組み合わせてプロダクトライン開発資産を表現するFRMモデルを提案する。また、このFRMモデルを扱うために、開発支援ツールを作成する。そして、このFRMモデルを用いたプロダクトライン開発のケーススタディを示す。

本稿の構成は以下の通りである。まず2章でRASおよびRASを用いた資産表現を紹介し、3章でFRMモデルの説明を行う。4章でFRMモデル向けに作成した開発支援ツールの概要を述べ、5章でFRMモデルを用いたケーススタディを示す。6章で既存の開発支援ツールの紹介を行い、7章でまとめと今後の課題について述べる。

2. RASを用いた資産表現

2.1 RAS

RASは、Rationalによって定義され、OMGによって標準化された、再利用可能な資産を表現するための定義である。ここでいう資産は、与えられた状況における問題を解決するためのものと定義されている。資産は複数の成果物を含む。成果物とは、ソフトウェア開発のライフサイクルで出力される以下に例示するようなものである。

- 要求
- ドキュメント
- モデル
- ソースコードファイル
- テストケース
- 開発効率向上のために利用したスクリプト

RASを用いることにより、開発資産の抽象化と、RASを開発資産の中間表現として扱うことによる、多種の開発資産の統一的な扱いが可能になる。また、標準化されていることによる、ツール開発の効率の向上も期待できる。

RASのUMLモデルをFig. 1に示す^{†1}。

RASではそれぞれの資産は、開発資産全体の属性を記述するための要素であるAsset要素を中心とした以下の4つの部分に分けて表現される。

Classification 資産(Asset)を分類するための記述子(Descriptor)の一覧である。Descriptorはその資産の品質や特徴を記述したもので、分類に用いる。Descriptorをグループ化した記述子群(DescriptorGroup)や、概念的枠組みを定義する文脈

(Context)も用いられる。ContextはAsset内の他の要素の意味情報を説明するために用いられる。

Solution 資産に対して与えられた問題への解決策(Solution)を構成する成果物(Artifact)を記述する。この要素には、Artifactの種類を表現するArtifactType, Artifactの依存関係を示すArtifactDependency, Artifactの拡張および変更される箇所を示す変化点(VariabilityPoint), ArtifactとContextの関係を示すArtifactContextを記述する。

Usage 資産の導入やカスタマイズ、利用する際のルールからなり、資産を利用(Usage)する際に行うべき作業の内容や手順を記述する。ここで記述される要素は、Assetを利用するために行う作業(Activity), Activityの適用先のVariabilityPointおよび、具体的な作業内容を示すVariabilityPointBinding, Assetによって適用されるActivityを格納するAssetActivity, Artifactによって適用されるActivityを格納するArtifactActivity, Contextによって適用されるActivityを格納するContextRefがある。

RelatedAsset その資産とほかの資産との関連(RelatedAsset)を記述する。これは、Asset間の関連を表現するための要素であり、関連するAssetそして関連の種類を表現する。

2.2 RASを用いた資産表現

フィーチャと開発資産間のトレーサビリティを実現し、多岐にわたる開発資産を統一的に表現するために、プロダクトライン開発資産全体を、フィーチャモデルとRASを用いて表現する。RASのArtifactによって個々の開発資産を表現し、Solution内の要素を用いて開発資産の属性や、開発資産間の関連を表現する。また、RASのDescriptorをフィーチャに基づいて作成し、フィーチャモデルのフィーチャと、RASのDescriptorを結びつけ、DescriptorGroupでまとめ、Contextでフィーチャの組み合わせを表現する。このフィーチャの組み合わせであるContextからContextRef、このContextすなわちフィーチャの組み合わせの際にArtifactに対して必要な作業を示すActivity、この作業が適用されるArtifactのVariabilityPointを示すVariabilityPointBindingを作成することにより、フィーチャとArtifactの関連を表現する。Fig. 2は、資産表現の概要を示したものである。

さて、この手法ではDescriptor, Context, ContextRef, Activity, VariabilityPointBindingがフィーチャと開発資産間の関連を示している。しかし、この関連はフィーチャと開発資産が共に存在しないと作成することができない。プロダクトライン開発のアプローチとしては、プロダクトラインを一から構築するHeavyWeight戦略と、既存資産から構築するLightWeight戦略の二種類の開発アプロー

†1 元のモデルの図では要素名が小文字で始まる名前表現されている。本稿では要素名と属性名を区別をつけるため、キャピタライズを行った要素名を利用している。

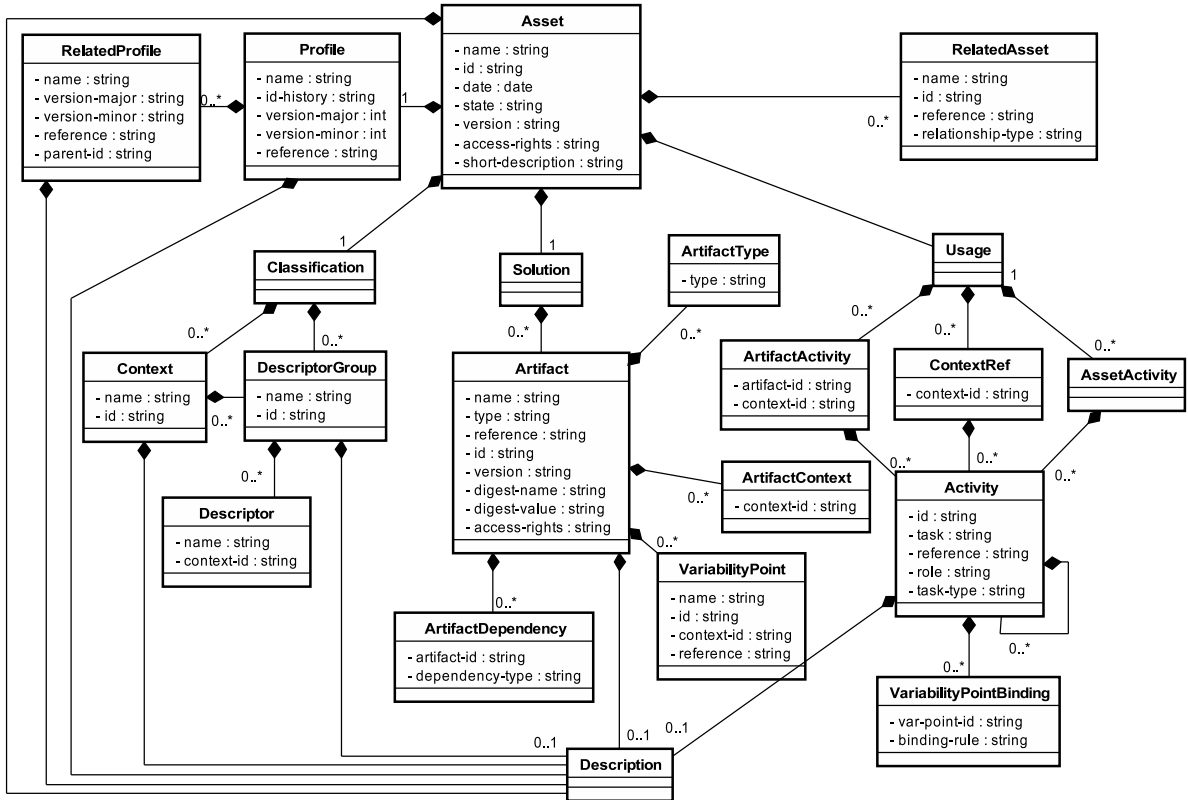


Fig. 1 RAS UML model⁵⁾.

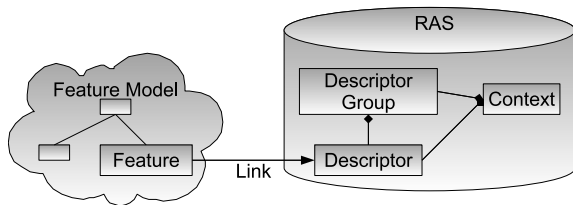


Fig. 2 Asset representation by RAS.

ち²⁾があるが、フィーチャから作成、開発資産から作成のいずれの場合でも、フィーチャと開発資産が共に完成しない限り、Relationの部分は作成することができず、不完全なRASのままとなってしまう。このことは、RASに対する種々の検証が遅れることを意味し、開発プロセス全体のスケジューリングにも悪影響を与える。また、RAS自体が大きなモデルであり、フィーチャと開発資産間のトレーサビリティ構築作業に本来かけるべきでない時間がかかっているという問題も挙げられる。

これらの問題を解決するために、RASを開発資産を示す部分と、フィーチャと開発資産間の関連を示す部分に分割し、RASのサブセットとすることで、開発資産のみ、関連部分のみのモデルの扱いを実現し、トレーサビリティ構築の効率の向上を図る。また、プロダクト

ライン開発に必要な部分のみを抽出することにより、RASを扱う開発コスト低減を実現する。

3. FRMモデル

3.1 概要

RASを用いたプロダクトライン開発資産表現を行う場合、前節で述べたように、DescriptorからActivityに至る要素がフィーチャと開発資産との関連を示している。これらの要素を抽出し、フィーチャの組み合わせとそのフィーチャ群に対応する開発資産間の関連を示したものをRelation Modelと定義する。また、開発資産そのものを表現するSolution部分を抽出し、ファイル、もしくは関数や変数などのファイル内要素を単位とし、開発資産を表現するものをMaterial Modelと定義する。そして、これらとFeature Modelを用いてプロダクトラインにおける開発資産を表現したものをFRMモデルと定義する。通常の開発資産とプロダクトラインにおける開発資産の違いはフィーチャに関連付けられているか否かである。

FRMモデルのクラス図をFig. 3に示し、以下、それぞれのモデルの解説を行う。

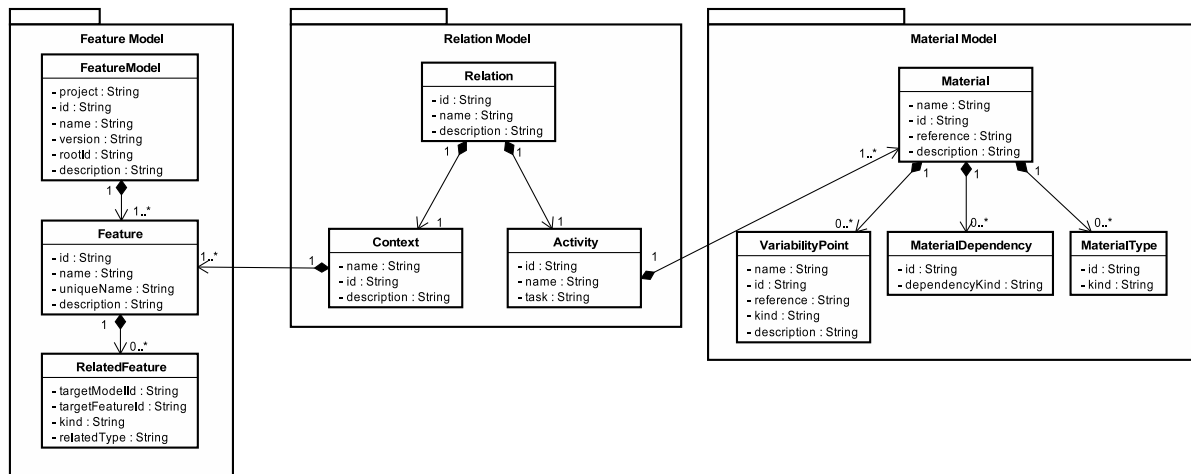


Fig. 3 Class diagram of FRM model.

3.2 Feature Model

Feature Modelではプロダクトラインの有するフィーチャ全てを表現する。構成要素はFeature Model自身の属性を示すFeature Model, それぞれのフィーチャを示すFeature, フィーチャ間の関連を示すFeature Relationである。

3.3 Relation Model

Relation Modelではフィーチャとマテリアルの関連を記述する。構成要素はRelation Model自身の属性を示すRelation, フィーチャの選択を示すContext, マテリアルの選択を示すActivityである。DescriptorはContextから直にFeatureに対してリンクを作成することにより消滅, ContextRefはRelationがその役割を果たすため消滅, VariabilityPointBindingはActivityに統合する。

3.4 Material Model

Material Modelでは、開発資産を表現する。構成要素はMaterial Model自身の属性を示すMaterial Model, 開発資産の個々の要素を示すMaterial, Materialの種類を示すMaterialType, Artifact間の依存関係を示すMaterialDependency, 使用される場面による, Materialの変更すべき箇所, 変更すべき内容を示すVariability Pointである。

3.5 RASとの互換性

前節で示したモデルは、2.2で述べたフィーチャモデルとRASへの変換が可能である。

Feature Model内の要素はそのままフィーチャモデルとして利用される。また、Feature Model内のFeatureとRASのDescriptorを作成し、DescriptorGroupでまとめる。

Relation Model内の要素は、ContextからRAS内のContextおよび、それぞれのContextに対応するContextRefを作成、ActivityからRAS内の関連するContextRefにぶら下がるActivityを作成する。RelationはContextとActivityの組に名前や説明を加えるものなので、RASでは使用しない。

Material Model内の要素は、MaterialがRAS内のArtifact, MaterialTypeからRAS内のArtifactType, MaterialDependencyからRAS内のArtifactDependency, VariabilityPointからRAS内のVariabilityPointを作成する。また、Materialに関連するActivityに対して、Materialの有するVariabilityPointのidとActivityのtask属性からVariabilityPointBindingを作成する。

これらに加え、ここまでで作成した要素をまとめるClassification, Solution, Usage要素を加え、さらにこれらをまとめるAsset要素を作成すればRASとして利用可能となる。

4. FRMモデルを用いたプロダクトライン開発支援ツール

4.1 概要

筆者らは、フィーチャ選択に基づいたプロダクトライン資産導出を行うため、提案したFRMモデルを利用したプロダクトライン開発支援ツールを作成した。Fig. 4はツールの概要および処理の流れを表す図である。また、スクリーンショットの一部をFig. 5に示す。このツールはEclipseのプラグインとして開発されており、以下の機能から成り立っている。

- フィーチャモデル管理機能
- リレーションモデル管理機能
- マテリアルモデル管理機能
- プロダクト導出機能

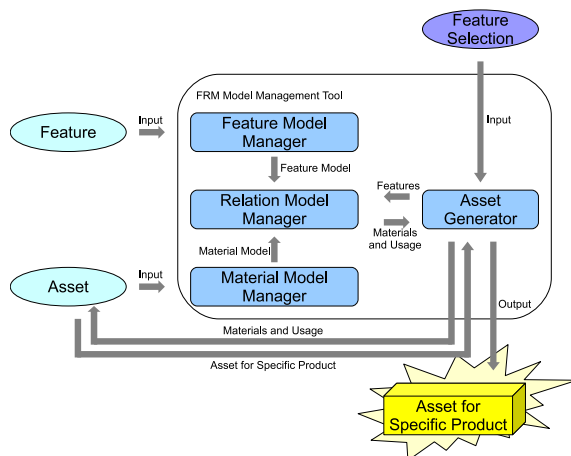


Fig. 4 Product line support tool by using FRM model.

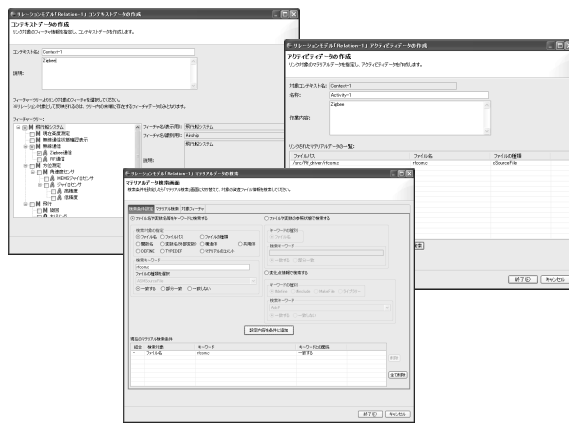


Fig. 5 The screenshot of our tool.

以下、それぞれについて説明する。

フィーチャモデル管理機能 この機能ではフィーチャモデルを扱う。現状では、フィーチャモデルを表すXMLファイルを読み込んで保持し、表示するだけであり、編集機能は有さない。

リレーションモデル管理機能 この機能ではリレーションモデルを扱う。フィーチャモデル、マテリアルモデルを読み込んで、それらの関連づけを行う。また、多数のマテリアルを読み込むことが予想されるため、関連付けるマテリアルを検索する機能を有している。

マテリアルモデル管理機能 この機能ではマテリアルモデルを扱う。手作業もしくはリバースツールを用いてマテリアルを登録。そして、登録されたマテリアルの属性表示や、マテリアルに対する詳細情報の付加を行うことができる。

資産導出機能 この機能では、インスタンスモデルを元に、各プロダクト向けの開発資産および、その使用方法を導出する。インスタンスモデルは、フィーチャ

モデル同様XMLファイルを読み込み、編集機能は有さない。

これらの機能により、フィーチャモデルからのフィーチャの選択によるプロダクト導出を実現する。

4.2 ツールを用いたプロダクト開発

Fig. 4の処理の流れを説明する。まず、フィーチャおよび、フィーチャを実現するための開発資産を作成する。作成する順番は既存資産の有無、プロダクトライン開発の開発アプローチ法によって異なる。作成したフィーチャはフィーチャモデル管理機能に入力する。開発資産はマテリアルモデル管理機能に入力して管理する。開発資産に関しては、C言語ソースコードはリバースツールによって自動的にインポートされるが、それ以外の資産はリバース手法⁷⁾⁸⁾を用いて手作業でマテリアル登録が必要がある。フィーチャモデルとマテリアルモデルが管理された後にこの二つのモデルからリレーションモデルを作成する。ここまででFRMモデルの構築は完了であり、これによってフィーチャとマテリアルを結びつけ、フィーチャと開発資産間のトレーサビリティを実現する。

FRMモデル構築後は各プロダクトの生成を行う。まず、各プロダクトに対応するフィーチャを選択し、プロダクト導出機能に入力する。このフィーチャに対応するリレーションを検索し、リレーションの情報から必要なマテリアルとマテリアルの使用方法を取得する。そしてマテリアルの情報から開発資産を取得し、生成したいプロダクトに必要な開発資産と利用法を取得することができる。

5. ECU 開発によるケーススタディ

筆者らは開発ツールおよびFRMモデルの有用性を確かめるべく、Engine Control Unit(ECU)の開発によるケーススタディを行った。ECUとは自動車のエンジンを制御する装置であり、エンジンの形状によって製品が派生する。

使用した開発資産は、C言語ソースコード、アセンブラソースコード、各種設計ドキュメント(Office, JUDE形式等)、ビルドに必要な構成ファイル等合わせて454ファイルであった。また、このプロダクトライン開発の規模は、派生製品が3つで、開発時間はおよそ350時間であった。

この開発資産をツールによってFRMモデルに変換し、プロダクトの導出を行った。また、ツールの有用性を確かめるために、フィーチャと開発資産の関連を表を用いて手作業で管理し、フィーチャの選択から各開発資産の抽出を行った。

この、モデルの作成から開発資産の導出までの開発時間を計測したところ、FRMモデルを用いた場合がおよそ17時間、表を用いて管理する方法を用いた場合がおよそ36時間と倍以上の差がみられた。特にフィーチャと開発

資産間の関連付け作業で大きく差が出、関連する開発資産を一括して検索し登録できるFRMモデルに比べ、手作業で表に登録する作業は時間がかかることから時間に差が付いたものと考えられる。

この開発時間は開発規模や作業者によって変動することが予想されるが、ある程度はツール利用の効果が示されたと考える。

6. 既存のプロダクトライン開発資産管理ツール

6.1 pure::variants

pure::variants³⁾はpure-systems社によって開発、販売されているEclipseベースのプロダクトライン開発支援ツールである。このツールでの資産管理はFamily Modelという開発資産をツリーで管理するモデルを採用している。Family ModelもRAS同様に多様な資産の管理が可能であり、フィーチャとのリンクを作成することによりフィーチャと開発資産間のトレーサビリティを実現している。しかし、RASと異なり標準化はされていない。

6.2 Gears

Gears⁴⁾はBigLever社によって開発、販売されているプロダクトライン開発支援ツールである。このツールでの資産管理はLogicと呼ばれる記述を用いて行われる。Logicは特有の記法を用いてフィーチャと開発資産間の関連を表現する記述である。このLogicも標準化はされていない。

本研究は、標準化されたRASを用いて開発資産表現を行うことにより、標準化準拠によるメリットの享受を目指しているため、これらのツールとはアプローチが異なる。

7. まとめと今後の課題

プロダクトライン開発を行う際には、フィーチャと開発資産間のトレーサビリティ確保が重要である。本稿に先立つ文献⁹⁾においては、このトレーサビリティ表現のために、フィーチャとRASを用いた表現を行った。しかし、トレーサビリティ表現はRASのサブセットでも十分可能であり、また、RASを用いたプロダクトライン開発資産表現では、フィーチャと開発資産間の関連がRASに組み込まれているため、関連付けを行うまでRASが不完全となり、RASモデルに対する種々の検証が行うことが

できない。

本稿ではRASを用いた資産表現を、開発資産と開発資産間の関連の観点で分割し、フィーチャとともにプロダクトライン開発資産を表現するFRMモデルを提案した。また、FRMモデルを用いた開発支援ツールの開発、およびECU開発でのケーススタディを行い、フィーチャに基づいたプロダクトライン開発資産の管理の実現、ならびにFRMモデルの有用性を確かめた。

今後の課題としては、開発支援ツールがまだ開発途上のため、その完成度を高めること、FRMモデルにおいて、リレーションの作成に大きな手間がかかったため、その労力を軽減するためのモデルもしくはツールの改良、FRMモデルを用いたプロダクトライン開発導入のためのガイドラインの考案、モデルを利用した開発の定量的な評価が挙げられる。

参考文献

- 1) Paul Clements, Linda Northrop, *Software Product Lines*, Addison-Wesley, 2002.
- 2) Paul Clements and Charles Krueger, *Point/Counter Point*, *IEEE Software*, Vol.19, No.4, pp.28-31, July 2002.
- 3) pure-systems GmbH., *Technical white paper variant management with pure::variants*, <http://www.pure-systems.com/>, 2003-2009.
- 4) BigLever Software Inc., *BigLever Software Gears*, <http://www.biglever.com/solution/product.html>, 2008-2009.
- 5) Object Management Group, *Reusable Asset Specification*, <http://www.omg.org/cgi-bin/apps/doc?formal/05-11-02.pdf>, 2005.
- 6) K. Kang et al., *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, tech. report CMU/SEI-90-TR-21, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1990.
- 7) K. Kang, M. Kim, J. Lee and B. Kim, *Feature-Oriented Re-engineering of Legacy Systems into Product Line Assets-A case study*, *Proc. International Software Product Line Conference (SPLC)*, pp.45-56, 2005.
- 8) I. John, *Capturing Product Line Information from Legacy User Documentation*, *Software Product Lines*, Springer-Verlag Berlin Heidelberg, pp.127-159, 2006.
- 9) 服部 勇祐, 平川 剛, 芦原 秀一, 中西 恒夫, 北須賀 輝明, 田頭 茂明, 福田 晃: RASを用いたソフトウェアプロダクトライン開発資産表現, 情報処理学会 組込みシステムシンポジウム 2008 (ESS2008), pp.71-77, 2008年10月.