要求・設計資産からのプロダクトライン アーキテクチャ抽出

熊木 健太郎† 鷲崎 弘宜† 深澤 良彰†

ソフトウェアプロダクトラインにおけるコアアセット開発は，既存のソフトウェア設計資産から構築されることが求められている．しかしながら，既存のソフトウェア設計資産は必ずしも再利用を考慮して設計されてはおらず，共通な部分構造を効率よく再利用することは難しい．そこで我々は，既存の要求とアーキテクチャを，語の類似性を用いた共通性・可変性分析することで追跡可能にし，リバースエンジニアリングによってフィーチャモデルとプロダクトラインアーキテクチャを自動抽出する手法を提案する．提案手法を用いることで，効率的なアーキテクチャ再利用を支援することができる．

Extraction of Product Line Architecture from Requirement and Software Assets

Kentaro Kumaki,† Hironori Washizaki† and Yoshiaki Fukazawa†

The core asset development at the software product line is needed to be built with existing software assets. However, the existing software assets are not always designed to reuse, it's difficult to reuse the common part of structure effectively. We propose the method of connecting architecture with requirement by analyzing commonality/variability with the similarity of words and making feature model and product line architecture automatically by reverse engineering. It enables to support effective reuse of architecture.

1. はじめに

プロダクトライン型ソフトウェア開発は，既存の検証済みソフトウェア資産(コアアセット)の再利用によって，開発工数削減，品質保証，リソースの有効活用が期待できる開発手法である[1][2]. Proactive なプロダクトライン開発のアプローチ[9]では，対象ドメインの開発に必要なコアアセットの予測は困難であるため，その構築に先行投資をするのはリスクが高い．これに対し Reactive なプロダクトライン開発のアプローチ[9]では，実績のある既存製品の設計資産を用いてコアアセットを構築することでリスクを低減できるため，広く用いられている[3][6].

Reactive なプロダクトライン開発において，プロダクトライン化させることを考慮しないで開発された既存製品を用いてコアアセット開発を行う場合がある．この場合，今後の製品開発において再利用が可能である設計資産が含まれているにもかかわらず，再利用が難しい設計資産と混在しているため，それらを効率的に特定し，コアアセットを構築することは難しい．特にソフトウェア構造(以下，アーキテクチャ)は，複数の機能要求と非機能要求が混ざり合って構成されており複雑なため，再利用可能な部分構造の特定は困難である．そこで我々は，既存の複数の要求を記述した文章(以下，要求)とアーキテクチャの組み合わせを対象に語の類似性を用いた共通性・可変性分析を行って追跡可能にし，リバースエンジニアリングによってフィーチャモデルとプロダクトラインアーキテクチャを自抽出する手法を提案する．提案手法によって，アーキテクチャの効率的な再利用を支援することができる.

†早稲田大学理工学術院 基幹理工学研究科
Graduate school of Fundamental Science and Engineering, Faculty of Science and Engineering, Waseda University.

2. プロダクトライン型開発の課題

2.1 ソフトウェアプロダクトライン

ソフトウェア開発の生産性の向上，品質の安定，開発期間の短縮には，品質の保証されたソフトウェア部品を蓄積し，これらの再利用によりソフトウェアを開発することが有効である[1][2]. ソフトウェアプロダクトラインでは，ソフトウェア製品系列を対象にコアアセットと呼ばれるソフトウェア設計資産(要求・アーキテクチャ・コンポーネント等)を整備し，それを再利用することでソフトウェア製品を開発する．プロダクトライン型開発は，あらかじめ想定したソフトウェア製品系列を対象に再利用性の高い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できる[2].

2.2 フィーチャモデル

フィーチャモデルは，製品系列に属する製品間の共通・可変な特性をツリー構造で表現したもので，コアアセットへの索引として用いられる[7]. 親フィーチャ

と子フィーチャは，mandatory，optional，alternative のいずれかのフィーチャ関係を持つ．フィーチャモデルの例を図 1 に示す．

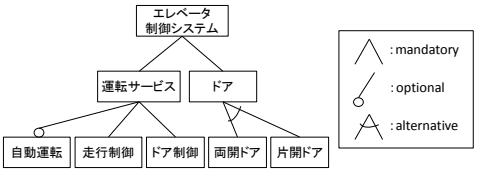

図 1. フィーチャモデルの例

製品系列の共通・可変な特性を表すことができるため，我々はフィーチャモデルとして対象ドメインの特性を表現する．なお，提案手法におけるフィーチャは機能的または非機能的特性を表す．ステレオタイプを持ち，これによってアーキテクチャ部分構造の追跡を実現する. また, フィーチャ関係の種類は考慮しない.

2.3 プロダクトラインアーキテクチャ

プロダクトラインアーキテクチャは，製品系列全体の構造的特徴を記述したものであり，製品系列全体のバリエーションに対応している．バリエーションの表現として，Clauss らはステレオタイプやタグ付き値など，UML の拡張機能を使った記述を提案している[8]. プロダクトラインアーキテクチャの例を図 2 に示す.

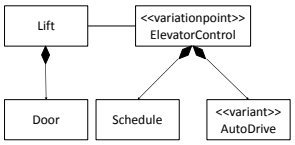

図 2. プロダクトラインアーキテクチャの例

提案手法では，ステレオタイプを用いてアーキテクチャの部分構造を特定する．そのため，プロダクトラインアーキテクチャの記述は Clauss らの手法を拡張し，ステレオタイプによるバリエーション表現法を用いる.

2.4 プロダクトライン開発における問題

Reactive なプロダクトライン開発において，プロダクトライン化させることを考慮しないで開発された既存製品を用いてコアアセット開発を行う場合がある．再利用を前提としない場当たり的な製品開発からプロダクトライン型開発に移行させる場合などである．この場合，既存製品の設計資産が再利用を目的として開発されていないため，再利用が可能な資産と再利用が難しい資産が混在してしまっている．そのため，今後の製品開発において再利用が可能である資産が含まれているにもかかわらず，それらを特定し，効率的にコアアセットを開発することは難しい．特にアーキテクチャは，複数の機能要求と非機能要求が混在しているため，再利用可能な部分構造の特定は困難である.

3. プロダクトラインアーキテクチャの自抽出方法

3.1 概要

2.4.節に挙げた問題を解決するために，我々は，既存製品の要求とアーキテクチャに対し，語の類似度を用いた共通性・可変性分析することで関係性を特定し，リバースエンジニアリングによってフィーチャモデルとプロダクトラインアーキテクチャを自動抽出する手法を提案する．実績ある過去の設計資産からプロダクトラインアーキテクチャを抽出することで，一定水準の妥当性あるプロダクトラインアーキテクチャの抽出が期待できる．自動で抽出することで，プロダクトラインアーキテクチャ設計にかかるコストを削減し，属人性を排除することができる．特性とアーキテクチャの関係を明確にし，フィーチャモデルとプロダクトラインアーキテクチャを追跡可能にすることで，プロダクト開発における効率的な再利用を支援することができる. 全体像を図 3 に, 提案手法の概要を以下に示す.

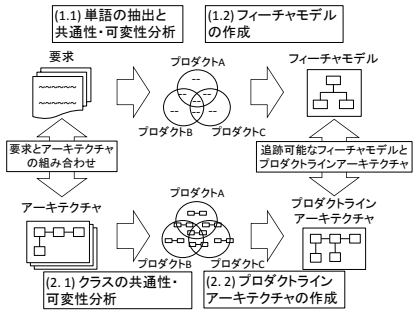

図 3. 提案手法の全体像

(1.1) 単語の抽出と共通性・可変性分析 : 要求から単語を抽出し，共通性・可変性分析を行ってフィーチャを特定する.

(1.2) フィーチャモデルの作成 : (1.1) で作成したフィーチャから，フィーチャモデルを作成する.

(2.1) 語の類似性によるクラスの共通性・可変性分析 : モデルに含まれるクラスの共通性・可変性分析を行い，類似したクラスを抽出する.

(2.2) プロダクトラインアーキテクチャの作成 : (2.1) で抽出した類似クラスをもとに，プロダクトラインアーキテクチャを作成する.

日本語で記述された要求を記述した文章と，英語で

UML クラス図として記述されたアーキテクチャの組み合わせを複数入力して，フィーチャモデルとプロダクトラインアーキテクチャが出力される．要求とアーキテクチャの組み合わせの集合に対して共通性・可変性分析を行うことで，要求とアーキテクチャの関係性を特定する．それらから，リバースエンジニアリングによって追跡可能なフィーチャモデルとプロダクトラインアーキテクチャを自動抽出する．

共通性・可変性分析によるフィーチャとアーキテクチャの関係性を特定する手順を図 4 に示す．あるプロダクトの組み合わせに共通して見られる特性は，同一のプロダクトの組み合わせに共通して見られる部分構造と関係していると考えられる．そこで提案手法では，アーキテクチャの部分構造はクラスの集合であると考え，同一のプロダクトの組み合わせに見られる特性とクラスの集合には関連性があると仮定し，それらを関連付ける．例えば図 4 において，フィーチャとアーキテクチャの共通性・可変性分析結果として，プロダクト Y とプロダクト Z に共通でプロダクト X にはないフィーチャ「自動走行」と，クラス「AutoDrive」がある．このとき，これら 2 つには関係性がある．

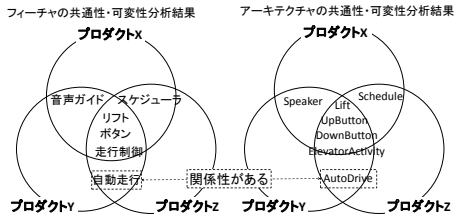

図 4. フィーチャとアーキテクチャの追跡

フィーチャとクラスの集合の関係性を明らかにするためには，抽出源となったプロダクトを明確にする必要がある．そこで，フィーチャとアーキテクチャの両方に抽出源となったプロダクト名をステレオタイプとして記述する．例えば，図 4 におけるフィーチャ「自動走行」とクラス「AutoDrive」は，共にプロダクト Y とプロダクト Z から抽出されているので，プロダクト名 Y，Z をステレオタイプとして付加する．同一のステレオタイプを持つクラスを追跡することによって，対象のフィーチャと関係性のあるクラス群を追跡することができる．なお，提案手法においては末端のフィーチャからのみ追跡するため，末端のフィーチャのみステレオタイプを持つ．

フィーチャモデルの作成とアーキテクチャの共通性・可変性分析を行うために，類似性を求める必要がある．そこで，概念辞書である日本語 WordNet[12]と WordNet[13]をもとに構築した WordNet::Similarity[14]の pathLength 法を用いて，フィーチャ名とクラス名を対象に単語間の類似度を測定する．WordNet は，各単語に上位語となる単語がツリー構造として体系づけられている．pathLength 法では，2 つの単語の上位単語をたどり共通する単語を求め，2 単語がたどった全単語数の逆数を単語間類似度と定義している．単語間類似度は 0 以上 1.0 以下の値で定義され，値が大きいほど 2 つの単語間の類似度は高くなる．測定対象となる語が単語でない場合や，品詞が異なるなど，単語間類似度の計測が不可能な場合は 0 となり，単語同士が完全に一致した場合は 1 となる．

3.2 プロダクトラインアーキテクチャの抽出方法

(1) フィーチャモデルの構築

(1.1) 単語の抽出と共通性・可変性分析

はじめに，各設計資産の要求から特性を抽出する．プロダクトの機能的，非機能的特性は，要求の中で単語として記述されていると考えられる．そこで形態素解析ツール Sen[11]を用いて，要求記述から単語を抽出する．抽出した単語群には「こと」「する」などのストップワードが含まれている場合がある．ストップワードが対象ドメインの機能や特性を表わしているとは考えられないので，フィルタリングする．さらに，単語群の追跡のために，抽出源となったプロダクト名をステレオタイプとして付加し，抽出源を明確にする．

続いて，抽出した単語を対象に共通性・可変性分析を行う．ここで，単語の中には同一の特性を表す類義語が含まれていることが多い．そのため，日本語 WordNet を用いて同一の特性の特定と共通化を行う．特性の共通化の際，それらの持つステレオタイプが異なる場合がある．すなわち抽出源が異なる場合である．このとき，それらの抽出源から単語を抽出したことを明確にするために，各ステレオタイプは継承する．

共通性・可変性分析を終えた単語群を対象ドメインのフィーチャとする．フィーチャの作成例を図 5 に示す．図 5 では，プロダクト X，Y，Z から抽出したフィーチャ「リフト」，「エレベータ」，「エレベータ」が類義語であるので同一化され，すべてから抽出源をステレオタイプとして継承している．

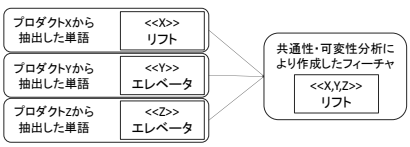

図 5. フィーチャ作成の例

(1.2) フィーチャモデルの作成

(1.1)で作成したフィーチャからフィーチャモデルを作成する．(1.1)で作成したフィーチャは単語であるため対象ドメインの特性としては最も具象度が高い．そこで，それらを抽象化したフィーチャでグループ化してツリー構造にする．フィーチャである単語の最も抽象度の高い分類は品詞によるものと考えられるので，品詞によってグループ化する．次に抽象度の高い分類は単語の類似性によるものと考えられるので，品詞によって分けられたグループごとに，日本語 WordNet を用いた WordNet::Similarity の pathLength 法によってフィーチャの単語間類似度を測定し，単語間類似度が閾値以上ならばグループ化する．フィーチャのグループ化では，グループごとのフィーチャ群を抽象化した親フィーチャを作成する．親フィーチャは日本語 WordNet で求めたグループのフィーチャに共通の上位語とし，末端のフィーチャではないのでステレオタイプは持たない. 図 6 にフィーチャモデル作成例を示す. 図 6 では，(1.1)で抽出されたフィーチャが実線枠，フィーチャのグループ化によって生成された親フィーチャが点線枠で表現されている．「転じる」「仕事」が品詞，「生じる」が単語間類似度によるグループ化において親フィーチャとして生成されている．

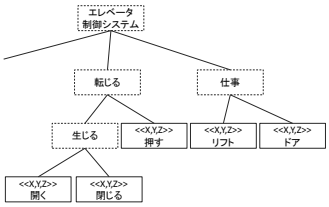

図 6. フィーチャモデル作成例(一部)

(2) プロダクトラインアーキテクチャの抽出

(2.1) 語の類似性によるクラスの共通性・可変性分析

各設計資産のアーキテクチャにおける類似なクラスを特定するために，クラス名を対象に類似度を測定する．クラス間類似度が閾値以上であれば類似したクラスと考え，共通化する．クラス間類似度は，単語の頭文字を大文字にする命名規則を利用して分解し，分解語のすべての組み合わせに対して単語間類似度を測定する．単語間類似度が最も高い組み合わせを特定し，それらを除いて同様の計算を単語数が少ないクラス名の単語数だけ繰り返して和を求める．和を単語数の多いクラス名の単語数で割り，得た数値をクラス間類似度と定義する．クラス間類似度は 0 以上 1.0 以下の値で定義され，値が大きいほどクラス間の類似度は高くなる．例として，クラス「ElevatorControl」と「ElevatorSelection」のクラス間類似度の測定手順を考える．まず，クラス名を分解してすべての組み合わせに対して単語間類似度を測定する(表 1)．表 1 より，「Elevator」と「Elevator」の単語間類似度が 1.0 で最大である．それらを取り除くと，類似度は「Control」と「Selection」の単語間類似度が 0.25 で最大になる．最後に特定した類似度の和を 2 で割り，クラス名間類似度=(1.0+0.25)/2≒0.63 となる．

	Elevator	Selection
Elevator	1.0	0.1
Control	0.2	0.25

表 1. クラス名間類似度の測定例

特定した類似クラスの組み合わせに対して，それらを抽象化した上位クラス名を定義する．上位クラス名は，クラス間類似度の計算に用いた単語の組み合わせに共通な上位語を組み合わせたものとする. 例として，クラス「ElevatorControl」と「ElevatorSelection」の上位クラス名を考える．クラス間類似度測定において特定した単語の組み合わせに対し，WordNet を用いて共通な上位語を求めると，「Elevator」と「Elevator」の共通上位語は「Elevator」，「Control」と「Selection」は「Activity」である．上位クラス名はこれらを組み合わせて「ElevatorActivity」となる．

(2.2) プロダクトラインアーキテクチャの作成

(2.1)で特定した類似なクラスとそれらの関連を合成することで，すべてのアーキテクチャを合成し，プロダクトラインアーキテクチャを作成する．

すべてのアーキテクチャのすべてのクラスと関連にステレオタイプとしてプロダクト名を付加し，抽出源を明確にし，共通な構造を単一化するために(2.1)で特定した類似なクラスをすべて合成する．このとき，関連とステレオタイプはそのまま継承し，名前は上位クラスへと変換する．クラス合成の例を図 7 に示す．図 7 では，クラス「ElevatorControl」と「ElevatorSelection」を合成し，上位クラスである「ElevatorActivity」を作成している．また，それぞれのクラスで保持していたステレオタイプと関連は上位クラスに継承されている．

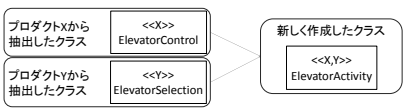

図 7. クラス合成の例

続いて，上位クラスが持つ関連について考える．ある上位クラスが他の上位クラスと直接関連を持ってい

る場合，それらは異なるステレオタイプを持った複数の関連を持つ場合がある．このとき，関連の種類を抽象化してそれらの関連の持つステレオタイプを継承することで合成する．関連の合成の例を図 8 に示す．図 8 では，プロダクトラインアーキテクチャ上の上位クラス「Lift」と「Door」，「Door」と「Sensor」間の関連が抽象化され，合成されている．

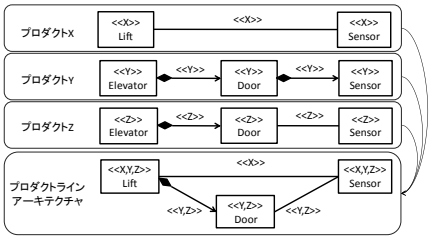

図 8. 関連の合成の例

4. 適用実験

4.1 実験概要

ET ロボコン 2009[15]に東海地区で出場した団体の中から，無作為に選んだ 4 団体が提出した要求とアーキテクチャの組み合わせ(本実験において，4 団体のプロダクト名を X，Y，Z，W と仮命名する)を用いてプロダクトラインアーキテクチャを抽出し，それからプロダクト開発で再利用できるクラスの割合を求める実験を行った．ET ロボコンとは，コースを自律走行するように設計されたロボットの走行性能を競うコンテストである．ロボットはコースに引かれた黒のラインを読み取り，それを光センサで感知して走行する．

各プロダクトについて，対象プロダクトを除く 3 プロダクトよりプロダクトラインアーキテクチャを抽出する．対象プロダクトの要求をプロダクト開発における要求，アーキテクチャを正解アーキテクチャと仮定し，プロダクト開発における要求からプロダクトラインアーキテクチャを用いて再利用できるクラスの割合を求める．例として，設計資産の 1 つ(プロダクト W)を図 9，プロダクト Y を対象としたときにプロダクト X，Z，W から作成したフィーチャモデルとプロダクトラインアーキテクチャの一部を図 10，11 に示す．

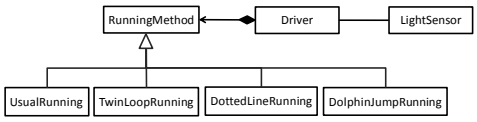

図 9. 入力したプロダクトアーキテクチャ(一部)

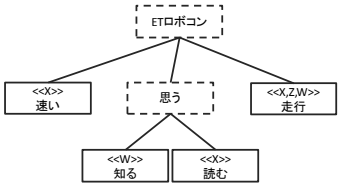

図 10. 作成されたフィーチャモデル(一部)

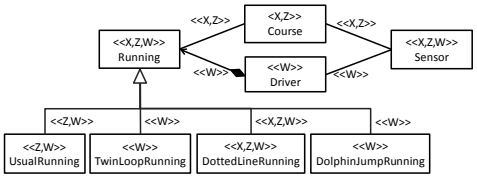

図 11. 作成されたプロダクトライン
アーキテクチャ(一部)

4.2 実験の評価と考察

対象プロダクトの要求から抽出したフィーチャのうち，提案手法により構築したフィーチャモデルの末端のフィーチャと一致したものが，提案手法によって再利用可能な特性である．そのようなフィーチャの数を，フィーチャモデルの末端フィーチャ数で割ったものをフィーチャ一致率として求めた．また，上記のフィーチャ群からステレオタイプを参照することで追跡できたクラス群のうち，対象プロダクトのアーキテクチャに含まれるものが実際に追跡できた(再利用できる)クラスである．そのようなクラスの数を，対象プロダクトのアーキテクチャに含まれるクラス数で割ったものをクラスの再利用率として求めた．各設計資産における一致率と再利用率を表 2 に示す．

設計資産	<<X>>	<<Y>>	<<Z>>	<<W>>
フィーチャー致率	0.286	0.133	0.314	0.375
クラスの再利用率	0.727	0.500	0.400	0.440

表 2. 各プロダクトの再利用率

本実験において，プロダクトラインアーキテクチャ抽出のために 3 つのプロダクトを用いた．その結果，各プロダクトにおけるフィーチャ一致率が 3 割前後となっている．これは，入力として用いたプロダクト数が少なかったため，プロダクトラインアーキテクチャの持つ特性のバリエーションが不足していたためであると考えられる．いっぽうで，各プロダクトにおけるクラスの再利用率は 5 割前後であり，フィーチャ一致率に比べて高い結果となった．このことから，プロダクトラインアーキテクチャを用いて追跡が可能な特性であれば，再利用可能な構造を広く特定できると考えられる．プロダクトラインアーキテクチャによって追

跡を行うことができる特性のバリエーションを増やすことで，更に多くの部分構造を追跡することが期待できる．

5. 関連研究

吉村らは，プロダクトライン導入のための共通性・可変性分析手法を提案している[6]．コードクローンによる解析によって，レガシーシステムの効率的な共通性・可変性を効率的に評価することが可能である．これに対し提案手法では，モデルレベルでの共通性・可変性分析を行い，さらに他の資産と対応付けて活用するための具体的な手法を提案している．

Kang らは，リエンジニアリングによる既存資産からのプロダクトライン構築手法を提案している[5]．既存の設計資産からリバースエンジニアリングによってアーキテクチャとフィーチャモデルを作成し，今後のニーズなどを想定して新規にフィーチャを追加してからプロダクトラインアーキテクチャを構築する．リエンジニアリングを繰り返して既存資産をコアアセット化するのに対し，提案手法では一度のプロセスでプロダクトラインアーキテクチャを抽出することができる．

野本らは，構造および語の類似性に基づいたアナリシスパターンの自動抽出を提案している[4]．語の類似性を基に，複数のモデルの組み合わせに共通の構造をアナリシスパターンとして抽出する．これに対し提案手法では，対象ドメイン全体の共通・可変な構造を抽出することができる．

6. おわりに

Reactive なプロダクトライン型開発において，プロダクトライン化させることを考慮しないで開発された既存製品を用いてプロダクトラインアーキテクチャを構築することは難しい．これに対し我々は，フィーチャモデルから必要な部分構造を追跡できるプロダクトラインアーキテクチャを自動抽出し，その再利用性を検証した．提案手法を用いることで，プロダクトライン開発を前提としていないソフトウェア設計資産を用いても，人手を介することなく一定の再利用率のプロダクトラインアーキテクチャ抽出の支援が期待できる．

今後の展望として，提案手法におけるフィーチャモデルは，フィーチャ間の類似度を求めてグループ化しているだけにとどまり，対象ドメインの共通・可変な特性をツリー構造で表現しているとは言えない．そのようなフィーチャモデルを構築することで，フィーチャモデルの理解しやすさの向上を実現したい．クラスの類似性分析では，「属性」「操作」「ロール名」「多重度」は考慮せずクラス名のみを対象として解析している．これらを扱うことで，より正確に類似クラスを特定することができ，より厳密な追跡が可能なプロダクトラインアーキテクチャの抽出が可能になると考えられる．また，提案手法によって抽出したプロダクトラインアーキテクチャについて再利用性の評価は行ったが，妥当性や，手動抽出と比較した場合のコスト削減の程度に関する検討を行っていない．それらの評価により，提案手法の更なる有用性の検証を行いたい．

参考文献

1) J. Greenfield, K. Short S. Cook, and S. Kent, “Software Factories”, Wiley, August 16, 2004,

2) P. Clements and L. Northrop, “Software Product Lines: Practice and Patterns”, Addison-Wesley, 2001

3) 位野木万里, “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”，情報処理学会誌, Vol.50, No.4, 2009

4) 野本悠太郎, 久保淳人, 鷲崎弘宜, 深澤良彰, “構造および語の類似性に基づくアナリシスパターンの自動抽出”, 第16回 ソフトウェア工学の基礎ワークショップ FOSE 2009

5) K. Kang, M. Kim, J. Lee, and B. Kim, “Feature-oriented Re-engineering of Legacy Systems into Product Line Assets – a Case Study”, Proc. of 9th International Software Product Line Conference, 2005

6) 吉村健太郎, ”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, 2007.

7) K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson, “ Feature-Oriented Domain Analysis (FODA) Feasibility Study ”, Technical Report, CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pitts-burgh, PA, 1990

8) M. Clauss, “ Generic modeling using UML extensions for variability ”, In OOP- SLA 2001 Workshop on Domain Specific Visual Languages, 2001

9) C. Krueger, “Eliminating the Adoption Barrier”, IEEE Software, Vol.19, No.4, 2002

10) K. Schmid, and M. Verlage, “The Economic Impact of Product Line Adoption and Evolution”, IEEE Software, Vol.19, No.4, 2002

11) 形態素解析 Sen, http://www.mlab.im.dendai.ac.jp/ya-mada/ir/MorphologicalAnalyzer/Sen.html

12) 日本語 WordNet, http://nlpwww.nict.go.jp/wn-ja/

13) WordNet, http://wordnet.princeton.edu/

14) WordNet::Similarity, http://wn-similarity.sourceforge.net/

15) ET ソフトウェアデザインロボットコンテスト 2009, http://www.etrobo.jp/