情報処理学会論文誌　Vol. 50　No. 11　2654–2664 (Nov. 2009)

製品リリース履歴における論理的結合集合に基づいた横断フィーチャ分析法

吉村　健太郎[†1]　成沢　文雄[†1]　菊野　亨[†2]

本論文では，既存システムにソフトウェア・プロダクトラインを導入するための，論理的結合集合に基づく横断フィーチャ分析法を提案する．大規模な組み込みシステムには可変フィーチャ数を削減する横断フィーチャの導入が有効だが，分析期間と定量的評価が課題となる．提案手法では製品リリース履歴におけるソフトウェア部品間の論理的結合集合に基づいて横断フィーチャ候補を抽出する．また，精度と再現度を評価用メトリックスとして定義する本手法により，横断フィーチャ候補を自動的かつ定量的に決定できる．本手法を車両制御システムに適用したところ，横断フィーチャ候補を精度 97%，再現度 31%で自動抽出した．

A Method to Analyze Cross-cutting Features Based on Logical Coupling Sets of Product Release History

Kentaro Yoshimura,[†1] Fumio Narisawa[†1]
and Tohru Kikuno[†2]

This paper describes a method to analyze cross-cutting features based on logical coupling of legacy product release history for migrating into software product line engineering. Crosscutting features help developer of large embedded systems to reduce the number of variable feature. However, time for analyzing and quantitative evaluations are problems to be solved. The proposed method mines candidates for crosscutting features from the product release history, based on the logical coupling of software components. Also, the method applies the precision and the recall as metrics and determines the candidates quantitatively and automatically. We apply the proposed method to an embedded software, automotive control system. The method extracted the candidates with evaluation metrics, 97% of precision and 31% of recall.

1. はじめに

自動車，産業用機器，家電製品等，広い分野で組み込みソフトウェアが用いられている．組み込みソフトウェアは，特定のビジネス領域における単一製品として開発されることが多く，その後ビジネスの成功とともに改良，派生が加えられ，多くの製品バリエーションで構成される製品系列（プロダクトライン）として開発されるようになる．たとえば自動車用エンジン制御システムは，エンジン仕様の違い，自動車メーカごとの独自要求仕様，さらに地域ごとに異なる法規制に対応するために製品バリエーションが非常に多い．その一方，ソフトウェア規模・複雑度の増大，開発コスト・期間の削減，信頼性の向上等の要求によって，バリエーションを個別に開発することはきわめて困難であり，組み込みソフトウェアの再利用性向上が重要な課題になっている．

近年，製品群を横断したソフトウェア再利用のための手法として，ソフトウェア・プロダクトライン（Software Product Line，以下，SPL）が注目されている[1)–6)]．SPL 型開発の特徴の 1 つは，フィーチャの概念の導入である．Kang らの定義[7)] によれば，フィーチャとはユーザ視点でのシステムの特性である．製品間の共通部分である共通フィーチャに加え，製品仕様に基づいて可変フィーチャを選択することによって，製品バリエーションの個別大量生産が可能になる．SPL 型開発の成功事例として，製品バリエーション数が多いドメインである携帯電話[8)] や自動車制御システム[9),10)] 等が報告されている．

その一方で，SPL の効率的な運用には可変フィーチャ数の削減が必要不可欠である．たとえば，2 通りの選択を持つ可変フィーチャを 20 個持つシステムですら，組合せ数は 100 万通りを超えてしまう．自動車制御システムでは，可変フィーチャ数が製品全体で数千になる例も報告されており[9)]，製品開発時に選択しなければならない可変フィーチャの組合せ数が膨大になる．このため，派生製品の開発において製品仕様を満足する可変フィーチャを設定するための工数が増大するという課題がある．SPL 型開発を効率的に運用するためには，可変フィーチャ数の削減が重要である．

SPL 導入後のシステムを対象として，横断フィーチャを用いた可変フィーチャ数の削減手法がすでに提案されている[11),12)]．横断フィーチャとは，複数の詳細な可変フィーチャを

†1 株式会社日立製作所日立研究所
Hitachi Research Laboratory, Hitachi, Ltd.
†2 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

© 2009 Information Processing Society of Japan

製品	可変フィーチャ	
	f_i（距離センサ）	f_j（ブレーキ制御）
A	○	○
B	×	×
C	○	○
D	○	○
E	×	×
パターン1	○	○
パターン2	×	×

○：要
×：不要

横断フィーチャ X_k（車間距離制御）
○
×

図 1　横断フィーチャ
Fig. 1　Cross-cutting feature.

横断して影響を及ぼす，抽象度の高いフィーチャである．**図 1** に車両制御システムの例を示す．可変フィーチャとして「距離センサ f_i」，「ブレーキ制御 f_j」が定義されている．ここで，f_i，f_j は独立なフィーチャとして個別に設定可能である．しかし，製品 A から E までの開発履歴を確認すると，f_i，f_j の両方が要，または両方が不要として設定されていることが分かる．横断フィーチャとして「車間距離制御 X_k」を定義することによって，共通に影響を受けていた複数の可変フィーチャ f_i，f_j をまとめられる．そのため，製品開発時の選択肢を削減することが可能となり，派生製品の開発を効率化できるという利点がある．

図 2 に横断フィーチャに基づく SPL の概要をまとめる．従来のソフトウェア再利用ではソフトウェアを部品化することによって再利用してきた．しかし，実際の製品開発における製品仕様とソフトウェア部品仕様とのギャップが大きいため，ソフトウェアの再利用性が低いという課題があった．これに対し SPL では，製品間における仕様の可変部分に対応する可変フィーチャを導入し，ソフトウェア部品と対応付けることによって再利用性を向上させている．さらに，可変フィーチャ数が大規模になる製品分野に対して横断フィーチャを導入することによって，製品開発時の選択肢を削減して開発を効率化できる．しかしながら，先行手法は SPL 導入後の可変フィーチャの設定実績に基づいているため，既存システムから SPL 型開発への移行には利用できない．ところが，自動車制御等のセーフティ・クリティカル・システムでは信頼性がきわめて重視されるため，実績のある既存システムに基づいて設計を最適化したいという要求が強い．そのため，開発済みの製品バリエーションに基づいて，横断フィーチャを分析することが望ましい．

一方で，ソフトウェア保守の分野では，論理的結合集合に基づいたアーキテクチャ分析手法[13] や開発支援手法[14] が成果をあげている．論理的結合集合とは，CVS 等のソフトウェアリポジトリ上で同時に変更されるファイル，関数，変数等のプログラム要素の同時変更関係である．しかし，従来手法は時系列に沿って直線的に進化するシステムを対象としており，SPL 型開発で対象とする開発分岐をともなう製品の分析には十分でない．

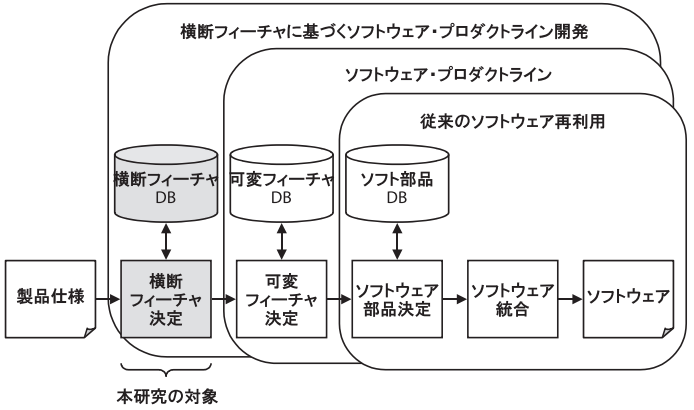

図 2　本研究の位置付け
Fig. 2　Scope of the proposed method.

そこで我々の研究の目的は，既存システムへの SPL 型開発導入のための横断フィーチャ分析法を確立することである．**図 3** に提案するアプローチの概要を示す．既存製品から SPL への移行にあたり，まずはじめに，(a) 製品リリース履歴の前処理を行う．次に，(b) 分析閾値に沿ってソフトウェア部品の論理的結合集合を抽出する．さらに，(c) 分析結果の評価を行う．(d) 決定した論理的結合集合に基づいて横断フィーチャを定義し，(e) 再利用資産として用いる．

上記目的を達成するための第 1 の課題は，バリエーションを有する既存製品の分析に適した，製品リリース履歴の前処理である．関連研究もリポジトリ上に蓄積されたデータを処理したうえで変更履歴の前処理を行っているが，多品種同時開発による分岐を考慮していない．提案手法では，開発履歴の分岐を考慮したうえで，製品リリースタイミング間での変更箇所を抽出して製品リリース履歴を前処理する．これにより，バリエーション開発が行われている製品分野における分析が可能になる．

第 2 の課題は，横断フィーチャ候補自動分析のための分析閾値の開発である．大量の製品リリース履歴を効率的に分析するためには，処理を自動化するための分析閾値を定義する必

© 2009 Information Processing Society of Japan

要がある．また，対象とする製品によって変更履歴の性質が異なるため，対象ドメインごとに調整が可能な分析閾値であることが望ましい．そこで我々は，最小同時変更頻度と最小同時変更信頼度をパラメータとした横断フィーチャ候補分析閾値を開発した．これにより自動分析が可能になるとともに，次に述べる評価メトリクスに基づいたパラメータ調整が可能になる．

第 3 の課題は，抽出した横断フィーチャ候補に対する評価メトリクスの開発である．分析閾値を用いて横断フィーチャ候補が抽出できるが，分析結果が SPL 導入支援の観点からどの程度有効かを定量的に評価する必要がある．そこで我々は，情報検索技術で実績のある「精度」と「再現度」の評価指標を SPL 向けに拡張した．精度は，横断フィーチャによる変更がどれだけ製品リリース履歴に合致するかを示す．再現度は，製品リリース履歴における変更箇所がどれだけ横断フィーチャによってカバーされるかを示す．これにより，分析した横断フィーチャ候補を定量的に評価することができる．

以上により，SPL の導入に向けて，レガシーソフトウェアに基づいて自動的にかつ定量的に横断フィーチャ候補を決定できる．

なお，提案手法の適用対象は同一のソフトウェアアーキテクチャ上で改良型開発が行われたソフトウェアの変更履歴であるという制限を持つ．van der Linden ら[15] によれば，製品系列の開発は新規開発，改良型開発を経たうえで，SPL 型開発へと移行することが多い．そのため，SPL 導入支援が必要な既存システムでは改良型開発が行われていることが多く，本手法の適用範囲は十分広いと考えている．

以下本論文では，2 章で本論文の提案する横断フィーチャ分析法について述べる．そして 3 章で，本手法の自動車エンジン制御用組み込みソフトウェアへの適用実験を行い，有用性を確認する．また 4 章で関連する研究との比較を行う．

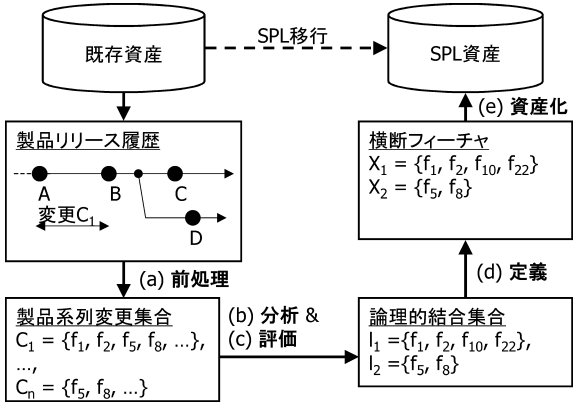

図 3　既存システムへの SPL 導入
Fig. 3　Migrating into SPL from legacy software.

2. 提 案 手 法

2.1 概　　要

本論文では，既存製品リリース履歴におけるソフトウェア部品間の論理的結合集合を用いた横断フィーチャ分析法を提案する．**図 4** に提案手法の全体構成を示す．

まずはじめに，(a) 製品リリース履歴の前処理を行う．本研究では製品間でのソフトウェア可変性を分析対象とするため，変更履歴の分析にはリリースされた製品間でのソフトウェア更新情報を用いる．また，バリエーション開発において生じることが多い開発の分岐を分析対象に含む．この前処理には既存製品の構成情報を入力とし，製品リリース間でのソフトウェア部品の変更情報（修正・追加・削除）が出力となる．この変更情報を，(b) での分析に用いる学習用データと，(c) での評価に用いる評価用データとに分割する．

次に，(b) 分析閾値に沿ってソフトウェア部品の論理的結合集合を抽出する．本研究の目的は横断フィーチャ候補の分析のため，ある論理的結合集合を横断フィーチャとして採用した際の効果が高くなるような分析閾値を用いる必要がある．そこで，ソフトウェア部品が同時に変更された頻度と，ソフトウェア部品が同時に変更された信頼度とを分析閾値として採用する．なお，同時変更頻度と同時変更信頼度の閾値はこの段階では固定しない．同時変更頻度と同時変更信頼度を段階的に変化させて論理的結合集合を抽出し，その結果を次の (c) で評価，対象システムに適した閾値を決定する．

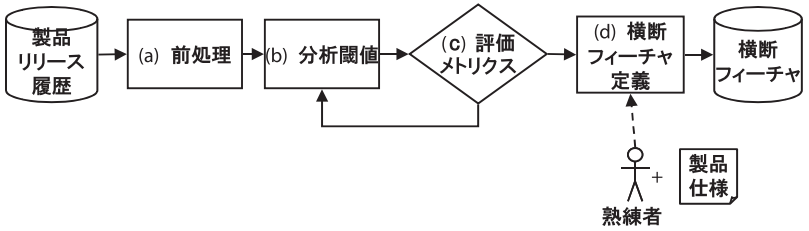

図 4　提案プロセスの概要
Fig. 4　An overview of the proposed method.

© 2009 Information Processing Society of Japan

	製品	A	B	C	D	E	F	G
システム	ベース製品	-	A	B	B	C	E	D
ソフトウェア部品	f_1	1.0	1.2	1.3	2.0	1.3	1.3	2.1
	f_2	1.0	1.1	1.1	1.1	1.2	1.2	1.1
	f_3	1.0	1.1	1.1	1.2	1.2	1.3	2.0
	f_4	1.0	1.0	-	-	-	-	-
	f_5	1.0	1.2	1.2	1.3	1.3	1.3	2.0
	f_6	-	-	1.1	-	1.1	1.1	1.1

図 5　製品リリース履歴
Fig. 5　Product release history.

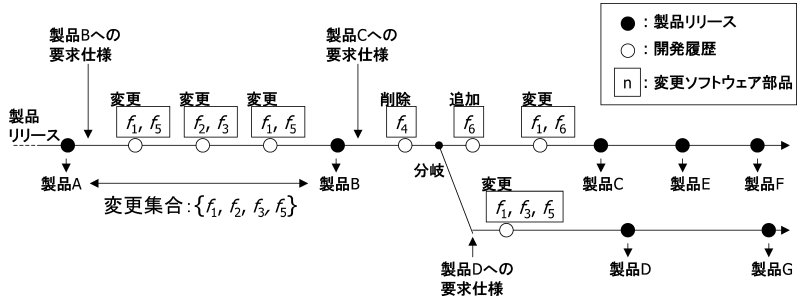

図 6　製品開発の流れ
Fig. 6　Product development flow.

続いて，(c) 抽出結果の評価を行う．ここでは，抽出した論理的結合集合が製品リリース履歴における変更をどの程度表現できているかを評価する．評価には，情報検索分野で実績のある精度と再現度のメトリクスをもとに，SPL 向けに拡張して用いる．精度は論理的結合集合どおりに変更が生じている割合を示す．再現度は製品リリース履歴における変更において，論理的結合集合が占める割合を示す．さらに，分析閾値を変更して抽出および評価を繰り返す．メトリクスに基づいて抽出結果を評価し，次の (d) で分析対象とする論理的結合集合を決定する．

最後に，(d) 決定した論理的結合集合に対して横断フィーチャを定義する．論理的結合集合を構成するソフトウェア部品と，論理的結合集合が生じた製品リリース履歴を用いて製品仕様との対応付けを行う．この工程は，熟練者が仕様書との対応をとりながら行う．

2.2　製品リリース履歴の前処理

本論文で提案する手法の入力となる，製品リリース履歴の一例を**図 5** に示す．製品リリース履歴は，新規製品名，ベース製品名，そして新規製品を構成するソフトウェア部品名およびその ID（バージョン情報等）で構成される．提案手法では，製品リリース履歴に基づいて，ベース製品から新製品への差分情報を抽出し，製品を構成するソフトウェア部品の変更集合として表現する．

製品を構成するソフトウェア部品を f_i，ソフトウェア部品全体の集合を F とする．

$$F = \{f_1, f_2, \cdots, f_n\} \tag{1}$$

製品リリース履歴 i において変更されたソフトウェア部品の集合を変更集合 C_i とする．C_i は，F のべき集合（Power(F)）に属する．

$$C_i = \{f_j, f_k, \cdots, f_l\}, C_i \in \mathrm{Power}(F) \tag{2}$$

・例 1　**図 6** の例では，製品 A から製品 B にかけての変更集合 C_1 は下記のようになる．

$C_1 = \{f_1, f_2, f_3, f_5\}$

図 6 に製品開発の流れの概要を示す．一般に，新製品の開発は複数回のソフトウェア変更で実現される．特に，ソフトウェア部品を新規に追加する場合は，バグ修正のため製品リリースまでに複数回更新されることが多い．ただし，提案手法は SPL 導入に向けた製品間可変性の分析である．そのため，1 回変更されたソフトウェア部品も，複数回変更されたソフトウェア部品も同等に扱うこととする．

また，新製品を並行開発するため，1 つのベース製品から複数の新製品が開発されることがある．これを製品リリース履歴の分岐と呼ぶ．図 6 の例では，製品 B から製品 C を開発している途中で，製品 D の開発へと分岐している．このような場合は，製品リリース履歴の分岐直前にリリースした製品と分岐後にリリースした製品との比較で変更集合を生成する．

なお，提案手法は SPL 導入前の製品間可変性分析を主な目的としている．そのため，分析例では個々のソフトウェア部品の機能を可変フィーチャとして定義し，可変フィーチャとソフトウェア部品とを 1 対 1 に対応させている．すでに部分的に SPL を導入し，1 つの可変フィーチャに複数のソフトウェア部品が対応している場合には，ある可変フィーチャに含まれているソフトウェア部品のいずれかが変更された場合に，その可変フィーチャは変更されたものとして扱い，変更集合を抽出する．

・例 2　図 6 の例では，製品 B から製品 D にかけての変更集合 C_3 は下記のようになる．

$C_3 = \{f_1, f_3, f_4, f_5\}$

© 2009 Information Processing Society of Japan

以上のように抽出した変更集合 C_i を用いて，製品リリース履歴全体の変更集合である製品系列変更集合 C を以下のように定義する．C は，F のべき集合のべき集合（Power(Power(F))）に属する．

$$C = \{C_1, C_2, \cdots, C_n\}, \quad (3)$$
$$C \in \mathrm{Power}(\mathrm{Power}(F))$$

提案手法では，抽出した横断フィーチャを評価するために k-分割交差検定[16] を採用する．k-分割交差検定では，標本群をおおむね同一サイズの k 個の部分集合にランダムに分割する．そして，$(k-1)$ 個の部分集合で構成されるデータを学習データとして用い，残り 1 つの部分集合を評価用データとして学習結果の評価を行う手法である．交差検定として，k 個に分割された標本群それぞれが評価用データとなるように k 回検定を行う．

提案手法では，製品系列変更集合 C を k 個の部分集合に分割し，$(k-1)$ 個の部分集合で構成される抽出用変更集合 C_L と，残り 1 つの部分集合で構成される評価用変更集合 C_E とに分割する．

$$C_L \subset C \quad (4)$$
$$C_E = C - C_L \quad (5)$$

以降では，C_L，C_E の要素をそれぞれ $C_{Li} \in C_L$，$C_{Ej} \in C_E$ と表記する．

・例 3　以降の説明のために，図 6 の例における製品系列変更集合 C を次のようにおいて計算例を示していく．

$$C = \{C_1, C_2, C_3, C_4, C_5, C_6\}$$
$$C_1 = \{f_1, f_2, f_3, f_5\}$$
$$C_2 = \{f_1, f_4, f_6\}$$
$$C_3 = \{f_1, f_3, f_4, f_5\}$$
$$C_4 = \{f_2, f_3, f_5\}$$
$$C_5 = \{f_3\}$$
$$C_6 = \{f_1, f_3, f_5, f_6\}$$

計算例における k-分割交差検定では簡単のために $k = 2$ とし，抽出用変更集合 C_L と評価用変更集合 C_E は次のように分割する．

$$C_L = \{C_{L1}, C_{L2}, C_{L3}\},$$
$$C_{L1} = C_1, C_{L2} = C_2, C_{L3} = C_3$$
$$C_E = \{C_{E1}, C_{E2}, C_{E3}\},$$
$$C_{E1} = C_4, C_{E2} = C_5, C_{E3} = C_6$$

2.3　論理的結合集合の抽出

2.3.1　分析用メトリクス

抽出用変更集合 C_L から論理的結合集合を抽出するため，分析メトリクスを定義する．分析用メトリクスとして，同時変更頻度（frequency）と同時変更信頼度（confidence）とを用いる．

同時変更頻度 freq は，複数個のソフトウェア部品で構成される論理的結合集合候補 x が，抽出用変更集合 C_L において同時に変更された回数を示す．その定義式を以下に示す．

$$x = \{f_j, f_k, \cdots, f_l\}, x \in \mathrm{Power}(F) \quad (6)$$
$$\mathrm{freq}(x) = |\{C_{Li} \mid C_{Li} \in C_L, x \subseteq C_{Li}\}| \quad (7)$$

同時変更信頼度 conf は，論理的結合集合候補 x に含まれるソフトウェア部品が変更されたときに，x 全体が変更された割合を示す．その定義式を以下に示す．

$$\mathrm{conf}(x) = \frac{\mathrm{freq}(x)}{|\{C_{Li} \mid C_{Li} \in C_L, x \cap C_{Li} \neq \phi\}|} \quad (8)$$

同時変更頻度 freq と同時変更信頼度 conf とをあわせて分析用メトリクス λ とする．

$$\lambda(x) = [\mathrm{freq}(x), \mathrm{conf}(x)] \quad (9)$$

・例 4　論理的結合集合候補 x_{ex} を以下のように定義した場合の分析用メトリクス $\lambda(x_{ex})$ を求める．

$$x_{ex} = \{f_1, f_4\}$$

ソフトウェア部品 f_1 と f_4 の組は，変更集合 C_2，C_3 において 2 回変更されている．よって，同時変更頻度は “2” となる．

$$\mathrm{freq}(x_{ex}) = |\{C_2, C_3\}| = 2$$

また，変更集合 C_1，C_2，C_3 において，x_{ex} の少なくとも一部に変更が生じている．よって，同時変更信頼度は $2/3 = 0.66$ となる．

$$\mathrm{conf}(x_{ex}) = \frac{\mathrm{freq}(x_{ex})}{|\{C_1, C_2, C_3\}|} = \frac{2}{3} = 0.66$$

よって，この論理的結合集合候補 x_{ex} に対する分析用メトリクス λ は以下のようになる．

$$\lambda(x_{ex}) = [2, 0.66]$$

© 2009 Information Processing Society of Japan

2.3.2 分析閾値

次に分析閾値を用いて，論理的結合集合候補を抽出する．分析閾値として，最小同時変更頻度と最小同時変更信頼度を用いる．最小同時変更頻度は，製品リリース履歴において最低限生じた同時変更回数を示す．最小同時変更信頼度は，論理的結合集合と扱うための同時変更信頼度の最小値を示す．

$$\theta = [\mathrm{freq}_{min}, \mathrm{conf}_{min}] \tag{10}$$

そのうえで，分析用メトリクス λ が分析閾値以上となる論理的結合集合候補を抽出し，論理的結合集合 $L(\theta)$ とする．

$$\begin{aligned} L(\theta) &= \{x \mid \mathrm{freq}(x) \geq \mathrm{freq}_{min}, \mathrm{conf}(x) \geq \mathrm{conf}_{min}\} \\ &= \{l_1(\theta), l_2(\theta), \cdots, l_k(\theta)\}, \end{aligned} \tag{11}$$

$$L(\theta) \in \mathrm{Power}(\mathrm{Power}(F))$$

・例 5　分析閾値として最小同時変更頻度は 2，最小同時変更信頼度は 0.75 に設定した場合を考える．

$\theta_{ex} = [2, 0.75]$

最小同時変更頻度 2 を満たす論理的結合集合候補として，$\{f_1, f_4\}$，$\{f_1, f_3, f_5\}$，$\{f_3, f_5\}$ が存在する．

$\lambda(\{f_1, f_4\}) = [2, 0.66]$

$\lambda(\{f_3, f_5\}) = [2, 1.0]$

$\lambda(\{f_1, f_3, f_5\}) = [2, 0.66]$

さらに同時変更信頼度に基づいて，閾値 0.75 を超えた $\{f_3, f_5\}$ が抽出される．

$$\begin{aligned} L(\theta_{ex}) &= \{x \mid \mathrm{freq}(x) \geq 2, \mathrm{conf}(x) \geq 0.75\} \\ &= \{l_1(\theta_{ex})\} \\ &= \{\{f_3, f_5\}\} \end{aligned}$$

2.4 論理的結合集合の評価

抽出した論理的結合集合 $L(\theta)$ を，評価用メトリクスを用いて定量的に評価する．横断フィーチャとして採用した場合の精度（Precision）と再現度（Recall）を評価用メトリックスとして定義する．

精度 P は，論理的結合集合と同一のパターンで変更が生じた度合いを示す．ある製品リリースにおける精度 P_j は，評価用変更集合 $C_{Ej} \in C_E$ において，論理的結合集合 $l_i(\theta)$ に含まれるソフトウェア部品が変更されたときの $l_i(\theta)$ に含まれるソフトウェア部品の割合として求められる．

$$P_j(\theta) = \frac{|\{l_i(\theta) \in L(\theta) \mid C_{Ej} \in C_E, l_i(\theta) \subseteq C_{Ej}\}|}{|\{l_i(\theta) \in L(\theta) \mid C_{Ej} \in C_E, l_i(\theta) \cap C_{Ej} \neq \phi\}|} \tag{12}$$

製品系列変更集合 C における精度 P は，P_j の平均値とする．

$$P(\theta) = \frac{1}{|C_E|} \sum_j P_j(\theta) \tag{13}$$

再現度 R は，製品リリース履歴で生じた変更に対して，論理的結合集合で再現できた度合いを示す．ある製品リリースにおける再現度 R_j は，評価用変更集合 $C_{Ej} \in C_E$ において，どの程度が論理的結合集合 $l_i(\theta)$ どおりに変更されたかどうかで判断する．

$$R_j(\theta) = \frac{|\{l_i(\theta) \in L(\theta) \mid C_{Ej} \in C_E, l_i(\theta) \subseteq C_{Ej}\}|}{|C_{Ej}|} \tag{14}$$

製品系列変更集合 C における再現度 R は，R_j の平均値とする．

$$R(\theta) = \frac{1}{|C_E|} \sum_j R_j(\theta) \tag{15}$$

評価メトリクスの代表値として，精度と再現度との調和平均 H を用いる．

$$H(\theta) = \frac{2}{\frac{1}{P(\theta)} + \frac{1}{R(\theta)}} \tag{16}$$

・例 6　評価用変更集合 C_4 の精度 $P_4(\theta_{ex})$ は，論理的結合集合が 1 組（$\{f_3, f_5\}$）のため，次のようになる．

$$\begin{aligned} P_4(\theta_{ex}) &= \frac{|\{f_3, f_5\}|}{|\{f_3, f_5\}|} \\ &= \frac{2}{2} \\ &= 1.0 \end{aligned}$$

さらに，C_4，C_5，C_6 における精度の平均値から，$P(\theta_{ex})$ は次のように求められる．

$$\begin{aligned} P(\theta_{ex}) &= \frac{1}{|C_E|} \sum_j P_j(\theta_{ex}) \\ &= \frac{P_4(\theta_{ex}) + P_5(\theta_{ex}) + P_6(\theta_{ex})}{|\{C_4, C_5, C_6\}|} \\ &= \frac{1.0 + 0 + 1.0}{3} \\ &= 0.66 \end{aligned}$$

また，変更集合 C_4 における論理的結合集合 $L(\theta_{ex})$ の再現度 $R_4(\theta_{ex})$ は次のようになる．

© 2009 Information Processing Society of Japan

$$R_4(\theta_{ex}) = \frac{|\{f_3, f_5\}|}{|\{f_2, f_3, f_5\}|} = \frac{2}{3} = 0.66$$

さらに，C_4，C_5，C_6 における再現度の平均値から，$R(\theta_{ex})$ は次のように求められる．

$$R(\theta_{ex}) = \frac{1}{|C_E|}\sum_j R_j(\theta_{ex}) = \frac{R_4(\theta_{ex}) + R_5(\theta_{ex}) + R_6(\theta_{ex})}{|\{C_4, C_5, C_6\}|} = \frac{0.66 + 0 + 0.5}{3} = 0.39$$

以上の結果から，調和平均値は以下のとおり 0.49 となる．

$$H(\theta_{ex}) = \frac{2}{\frac{1}{P(\theta_{ex})} + \frac{1}{R(\theta_{ex})}} = 0.49$$

2.5　分析閾値のサンプリング

以上の手順を分析閾値を変化させて，繰返し論理的結合集合を分析する．対象とするシステムによって，論理的結合集合が生じる傾向は異なるため，最小同時変更頻度および最小同時変更信頼度を変更して論理的結合集合を評価する．

・例 7　図 7 に分析閾値のサンプリングの例を示す．この例では，最小同時変更頻度は 2，3，4，5 と変化させ，最小同時変更信頼度は 0.25，0.50，0.75，1.0 と変化させる．計 16 通りの閾値の組合せで論理的結合集合を分析する．

さらに，分析した精度と再現度の調和平均により，対象とするシステムにおける分析閾値を決定する．調和平均値が最大になる閾値を対象システム分析用の閾値 θ_{max} として採用し，抽出した論理的結合集合 $l_i(\theta_{max})$ を横断フィーチャ X_i として採用する．

$$X_i = l_i(\theta_{max}) \tag{17}$$

2.6　横断フィーチャの定義

最後に，分析・評価した論理的結合集合を横断フィーチャとして定義する．分析に用いる情報は，論理的結合集合を構成するソフトウェア部品の仕様と，論理的結合集合による変更が生じた製品リリース履歴での製品仕様である．この分析は，対象製品の知識を十分に持った熟練者が行う必要がある．

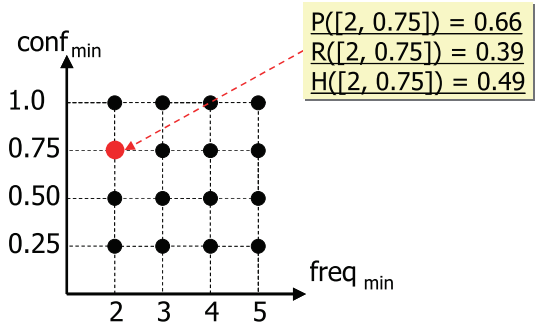

図 7　閾値のサンプリング
Fig. 7　Sampling of threshold.

まず，ソフトウェア部品仕様の分析を行う．論理的結合集合を構成している複数のソフトウェア部品に対し，共通の機能・非機能的要件を推定する．具体的な手順は 3.2 節で説明する．

次に，製品仕様の分析を行う．論理的結合集合による変更が生じた製品リリースにおいて，推定した機能・非機能的要件が変更されているかどうかを分析する．推定した要件が変更されていれば，その推定を横断フィーチャとして採用する．もし変更されていなければ，ソフトウェア部品仕様の分析を再び行う．

3. 適 用 実 験

3.1　実験の概要

提案手法の有効性を評価するため，組み込み制御ソフトウェアの製品リリース履歴に対して本手法を適用した．

実験対象の概要を**表 1** に示す．分析対象のソフトウェアは自動車用エンジン制御システムであり，ソフトウェア全体の規模は 50 万 LOC 以上である．実験では，特にエンジン気筒数等の機械的構成や，仕向地による排気規制等の非機能要件の影響を受けやすい，燃焼制御サブシステムの一部を選定して実験対象とした．対象となるサブシステムは 63 個のソフトウェア部品で構成されている．各ソフトウェア部品はたとえば補正係数等の，粒度の細かいフィーチャ f_1 から f_{63} に対応している．製品変更履歴には 37 製品が含まれ，搭載車種，気筒数等のエンジン形式，出荷地域等多くの製品仕様が少しずつ異なっている．

適用実験では，図 4 に示したプロセスに沿って実施した．まず，37 製品の製品リリース

© 2009 Information Processing Society of Japan

表 1　実験対象
Table 1　Experimental example.

製品系列	エンジン制御（一部）
製品数	37
ソフトウェア部品数	63

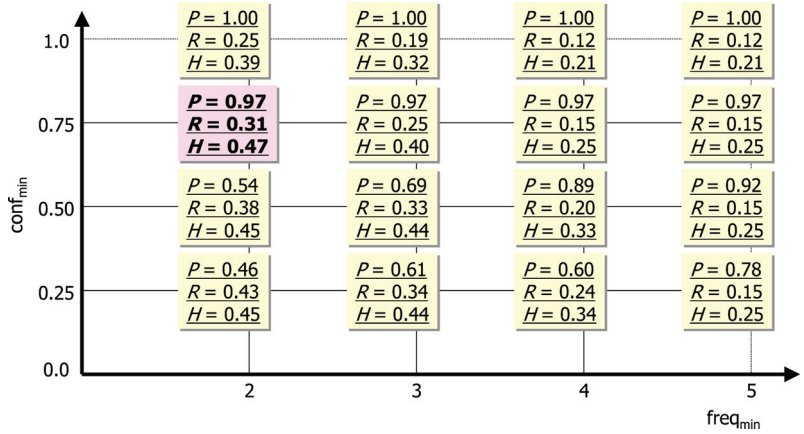

図 8　評価メトリクスの実験結果
Fig. 8　Evaluation metrics result.

履歴に対する前処理を行って，36 個の変更集合からなる製品系列変更集合 C を得た．交差検定法としては，k-分割交差検定の一種である leave-one-out 交差検定を採用した．製品系列変更集合 C を，35 個の変更集合からなる分析用変更集合 C_L と，1 個の変更集合からなる評価用分析集合 C_E に分割する．分析閾値に基づいて論理的結合集合を抽出し，評価用分析集合を用いて評価メトリクスを得る．この抽出および評価を，全変更集合が 1 回ずつ評価用分析集合となるように繰り返した．さらにこの一連の処理を，分析閾値を変更しながら繰り返し行い，製品リリース履歴全体での評価メトリクスを求めた．

3.2 実験結果

適用実験で得られた論理的結合集合の評価メトリクスを**図 8** に示す．分析閾値の設定値を変更しながら，複数回実験を実施した．たとえば，最小同時変更頻度 2，最小同時変更信頼度 0.75 における評価メトリクスは精度 0.97，再現度 0.31 となった．精度 0.97 は，論理的結合集合を構成するソフトウェア部品が変更されるときには，97%の割合で論理的結合集合どおりに変更されたことを意味する．再現度 0.31 は，製品リリース履歴で生じた変更のうち，論理的結合集合のパターンどおりの変更が 31%を占めていることを意味する．

表 2　横断フィーチャ定義結果
Table 2　Result of crosscutting feature definition.

横断フィーチャ	可変フィーチャ
X_1	f_{21}，f_{31}
X_2	f_{25}，f_{26}，f_{27}
X_3	f_{28}，f_{29}，f_{30}
X_4（燃料性状）	f_{32}（点火時期補正），f_{39}（燃料補正）
X_5	f_{50}，f_{51}
X_6	f_{52}，f_{53}

次に，横断フィーチャの定義を行った．分析対象を選定するため，評価メトリクスの調和平均を用いた．調和平均は，分析閾値を最小同時変更頻度 2，最小同時変更信頼度 0.75 の結果が最も高かったため，このときの論理的結合集合を分析対象とした．

論理的結合集合に対して仕様分析を行い，横断フィーチャを定義した結果を**表 2** に示す．まず，論理的結合集合にあるソフトウェア部品の可変フィーチャを調査し，その後に可変フィーチャ間に共通する横断フィーチャを推定，確認した．

たとえば，点火時期補正 f_{32} と燃料補正 f_{39} という 2 つのソフトウェア部品に共通するフィーチャとして，燃料性状（レギュラー or ハイオク）という横断フィーチャX_4 を定義した．これは，製品が対応している燃料性状の違いにより，点火時期補正 f_{32} と燃料補正 f_{39} とに関わる制御アルゴリズムが変更されるためである．さらに，論理的結合集合が生じている製品リリース履歴において，定義した横断フィーチャ（燃料性状）が実際に変更されていたかどうかを調べ，推定の妥当性を確認した．

上記の仕様分析を抽出した論理的結合集合に対して実施し，燃料性状やバルブタイミング制御の有無等，6 つの横断フィーチャX_1，X_2，X_3，X_4，X_5，X_6 を定義した．

3.3 考　　察

まず，製品リリース履歴の分析について検討する．自動車用エンジン制御システムは，複数製品の並行開発が行われているため製品リリース履歴にも開発履歴の分岐が存在した．この課題に対し，提案手法は製品リリースタイミング間での変更箇所を抽出しており，バリエーション開発が行われている製品分野での既存システム分析が可能となった．

次に，論理的結合集合の自動抽出に関しては，提案した分析メトリクスおよび分析閾値を用いて自動抽出を実施した．これにより，大量の既存製品を有する製品分野においても，提

© 2009 Information Processing Society of Japan

案手法により横断フィーチャ候補の抽出が可能となる見通しを得た．

さらに，熟練開発者へのヒアリングによって，評価メトリクスに基づいた横断フィーチャ分析結果の妥当性を検証した．提案手法により定義された 6 つの横断フィーチャ中の 5 つについては，複数ソフトウェア部品の同時変更要因として熟練開発者の経験と一致した．また，1 つの横断フィーチャは熟練開発者の経験と一致しなかったが，詳細な検討の結果，対象製品の横断フィーチャとして新たに確認された．また，横断フィーチャを構成する複数のソフトウェア部品を統合化できるかを検討したが，各ソフトウェア部品は細粒度制御要件に対応しており，統合化は困難であると判断した．その一方で各ソフトウェア部品の変更要因は共通であり，提案手法の狙いどおり横断フィーチャが抽出できたと考えられる．

以上の考察より，本研究において課題とした点を，提案手法により解決できたと考える．

最後に，実験結果に基づいて本手法の有用性を議論する．本実験で用いた適用対象は 63 個のソフトウェア部品で構成されている．そのため，横断フィーチャ候補となる 2 つのソフトウェア部品の組合せは 1,953 通り，3 つのソフトウェア部品による組合せは 39,711 通り，合計 41,644 通り存在する．これらすべての組合せに対して熟練開発者による仕様分析を実施した場合，その分析期間は長期間に及ぶという問題がある．これに対して提案手法では，既存システムの製品リリース履歴における論理的結合集合に基づき，2 個または 3 個のソフトウェア部品で構成される 6 つの横断フィーチャ候補を自動的に抽出することに成功した．提案手法の適用により，熟練開発者による仕様分析が必要な横断フィーチャ候補を約 7,000 分の 1 に削減し，既存システムへの横断フィーチャの導入効率を大幅に向上させることができる見通しを得た．

今後の研究課題としては，既存製品数が少ない場合の適用範囲の検討と，横断フィーチャ間の依存関係の分析とがある．

提案手法を適用するためには，対象製品のソフトウェア構造が安定した状態で，ある程度の量の変更履歴が蓄積されていることが必要である．提案手法の適用範囲を明らかにするため，分析対象とする製品数を変化させ，分析結果の評価を行うことが必要である．

横断フィーチャ間の関係について検討したところ，ある横断フィーチャが変更された場合にのみ，別の横断フィーチャも変更されるケースがあった．ここから，横断フィーチャ間の依存関係があることが推測できる．今後，横断フィーチャ間の依存関係の分析に対応することが必要である．

4. 関連する研究

組み込みシステムを対象としたプロダクトラインの事例研究がある[8)–10)]．しかし，これらの研究では，機能要求，バリエーション管理およびアーキテクチャ設計を対象としており，実績ある既存システムを活用するという観点から見て十分ではない．また，Kang らはフィーチャ指向に基づいて，既存システムをプロダクトライン化する手法を提案している[17)]．単一のシステムのソースコードにおける関数間の依存関係から実装されているアーキテクチャを回復させ，SPL の観点からアーキテクチャを改良している．しかしながら複数のバリエーションを有する既存システムの分析には十分でない．

これに対して我々の研究は，既存製品リリース履歴におけるソフトウェア部品間の論理的結合集合に基づいて横断フィーチャを分析するという特徴がある．本手法は，過去の実績に基づいた候補を提示することが可能であり，特に信頼性を重視する組み込み制御システム向け SPL 導入支援手法として有効である．

変更履歴中の同時変更関係に基づくソフトウェア改良の研究がある．Gall ら[13)] は通信交換システムの製品リリース履歴における同時変更関係を分析し，モジュール分割の見直しを行う手法を提案している．この手法は，時間系列に沿って発展していく単一のシステムを対象としているため，製品開発履歴の分岐は考慮されておらず，バリエーションを持つ製品にはそのまま適用することができない．

我々の手法は，同時変更関係分析の先行研究を既存システムへの SPL 導入手法として拡張したものであり，バリエーション開発による製品リリース履歴の分岐を考慮しているという特徴がある．このため，複数の製品バリエーション間が並行して開発されることがある組み込みシステムにおいて特に有効である．

我々はすでに，既存製品間のコードクローンに基づいて，共通性と可変性を分析する手法を提案している[18)]．しかし，大規模システムにおいては可変フィーチャが数千に及ぶことがあり，製品開発時の可変フィーチャ設定が困難になるという課題があった．この課題を解決するため，既存製品の変更履歴における同時変更関係を分析して可変性と結びつける基本的な概念を提案したが，分析結果に対する定量的な評価が欠けているという課題があった[19)]．そこで論理的結合集合に基づく定量的な分析手法を提案したが，厳密な定義が不十分であるという課題があった[20)]．

これに対し本論文では，抽出した論理的結合集合が SPL における可変性の概念に合致しているかどうか評価するメトリクスを形式的に定義して導入した．これにより，分析結果の

© 2009 Information Processing Society of Japan

定量的評価が可能になるとともに，SPL 導入効果を最適化するための分析パラメータ調整が可能になる．

5. ま と め

本論文では，既存システムへのソフトウェア・プロダクトライン導入のための横断フィーチャ分析手法として，製品リリース履歴における論理的結合集合を用いる手法を提案した．また実際の組み込み制御ソフトウェアを用いて適用実験を行った．その結果，横断フィーチャ候補の自動抽出が可能であることが分かった．

今後の取組みとしては，適用対象の製品数を増減させることによる提案手法の適用範囲の検証および，分析結果に基づく SPL 導入を行うことが必要である．

参 考 文 献

1) Bayer, J., et al.: A methodology to Develop Software Product Line, *Proc. 5th ACM SIGSOFT Symposium on Software Reusability* (*SSR'99*) (1999).
2) Clements, P. and Northrop, L.M.: *Software Product Lines: Practices and Patterns*, Addison-Wesley (2001).
3) Pohl, K., Böckle, G. and van der Linden, F.: *Software Product Line Engineering: Foundations, Principles And Techniques*, Springer (2005). 林　好一，吉村健太郎，今関　剛（訳）：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，エスアイビー・アクセス (2009).
4) 渡辺晴美：組み込みソフトウェア開発技術：4. プロダクトライン開発技術，情報処理，Vol.45, No.7 (2004).
5) Lee, J. and Muthig, D.: Feature-Oriented Variability Management in Product Line Engineering, *Comm. ACM*, Vol.49, No.12, pp.55–59 (2006).
6) 吉村健太郎：製品間を横断したソフトウェア共通化技術—ソフトウェアプロダクトラインの最新動向，情報処理，Vol.48, No.2, pp.171–176 (2007).
7) Kang, K., et al.: Feature-Oriented Domain Analysis (FODA) Feasibility Study, No.CMU/SEI-90-TR-21, ADA 235785, Software Engineering Institute, Carnegie Mellon University (1990).
8) Ommering, R.V. and Bosch, J.: Widening the scope of Software Product Lines – From Variation to Composition, *Proc. International Software Product Line Conference* (*SPLC*) (2000).
9) Steger, M., et al.: PLA at Bosch Gasoline Systems: Experiences and Practices, *Proc. Intl. Software Product Line Conference* (*SPLC*) (2004).
10) Thiel, S. and Hein, A.: Modeling and Using Product Line Variability in Automotive Systems, *IEEE Software*, Vol.19, No.4 (2002).
11) Loesch, F. and Ploedereder, E.: Optimization of Variability in Software Product Lines, *Proc. 11th International Software Product Line Conference*, pp.151–162 (2007).
12) Conejero, J.M. and Hernandez, J.: Analysis od Crosscutting Features in Software Product Lines, *Early Aspects at ICSE: Aspect-Oriented Requirements Engineering and Architecture Design* (*EA 2008*) (2008).
13) Gall, H., Hajek, K. and Jazayeri, M.: Detection of Logical Coupling Based on Product Release History, *ICSM '98: Proc. International Conference on Software Maintenance*, p.190 (1998).
14) Zimmermann, T., Weisgerber, P., Diehl, S. and Zeller, A.: Mining Version Histories to Guide Software Changes, *ICSE '04: Proc. 26th International Conference on Software Engineering*, pp.563–572 (2004).
15) van der Linden, F., et al.: Software Product Family Evaluation, *5th International Workshop on Software Product Family Engineering* (*PFE 2003*) (2004).
16) Kohavi, R.: A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, *Proc. International Joint Conference on Artificial Intelligence* (*IJCAI'95*) (1995).
17) 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. International Software Product Line Conference* (*SPLC*) (2005).
18) Yoshimura, K., Ganesan, D. and Muthig, D.: Defining a strategy to introduce a software product line using existing embedded systems, *EMSOFT '06: Proc. 6th ACM & IEEE International Conference on Embedded Software*, pp.63–72 (2006).
19) Yoshimura, K., Narisawa, F., Hashimoto, K. and Kikuno, T.: FAVE: Factor analysis based approach for detecting product line variability from change history, *MSR '08: Proc. 2008 International Workshop on Mining Software Repositories*, pp.11–18 (2008).
20) 吉村健太郎，成沢文雄，橋本幸司，菊野　亨：製品リリース履歴における論理的結合関係に基づいた横断フィーチャ分析法，組込みシステムシンポジウム 2008 論文集，pp.51–59 (2008).

(平成 21 年 1 月 30 日受付)
(平成 21 年 9 月 11 日採録)

© 2009 Information Processing Society of Japan

吉村健太郎（正会員）

1976 年生．1999 年早稲田大学理工学部機械工学科卒業．2001 年同大学大学院理工学研究科機械工学専攻修士課程修了．同年（株）日立製作所日立研究所入社．2005〜2007 年仏国 Hitachi Europe．現在，（株）日立製作所日立研究所研究員．2009 年大阪大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．主に組み込みソフトウェア開発方法論に関する研究に従事．IEEE，日本機械学会各会員．

成沢　文雄（正会員）

1970 年生．1994 年東北大学工学部情報工学科卒業．1996 年同大学大学院情報科学研究科情報基礎科学専攻修士課程修了．同年（株）日立製作所日立研究所入社．ソフトウェア工学，組み込みシステムの研究開発に従事．ACM 会員．

菊野　　亨（フェロー）

1975 年大阪大学大学院博士課程修了．工学博士．同年広島大学工学部講師．同大学助教授を経て，1987 年大阪大学基礎工学部情報工学科助教授．1990 年同大学教授．現在，大阪大学大学院情報科学研究科教授．2008 年大阪大学留学生センター・センター長．主にフォールトトレラントシステム，ソフトウェア開発プロセスの定量的評価に関する研究に従事．電子情報通信学会，情報処理学会各フェロー．ACM，IEEE 各会員．日本信頼性学会会長．

© 2009 Information Processing Society of Japan