Vol.2010-SE-170 No.11
2010/11/12

フィーチャを用いたプロダクトラインのバリエーション管理

山口信人[†]　　加藤滋郎[†]

新規製品のテストは過去の製品バリエーションとの差分に着目して行われるが，そのためには，フィーチャの実績のある組合せを管理して，新しい組合せなどが容易に特定できる仕組みが必要である．本稿では，ソフトウェアプロダクトラインの製品実績をフィーチャで管理する方法を提案するとともに，実績を利用することで，従来の差分開発と比べて生産性が向上することを示す．

Variation Management for Software Product Lines with Feature Model

Nobuhito Yamaguchi[†] and Shigeo Kato[†]

Tests for the new product in the product line are made by focusing on differences from the past products. To identify differences, it is necessary to figure out feature combination of the past products, which makes easy to identify occurrence of new feature combinations. In this paper, we propose a method for handling the feature combination of the past products in the product line, and illustrate how the efficiency of the product development is improved in comparison with traditional derivative developments.

1. はじめに

自動車システムにおけるソフトウェアへの依存度は，年々強くなる一方である．多様なユーザニーズに合わせて，自動車メーカーは多くのバリエーションを用意している．また，グローバル化が進み，世界各国のニーズ，法規に合わせたバリエーションも必要である．自動車システムにおける組込みソフトウェア開発は，この発展に適応していかなければならない．

このような課題に対する最も有力な戦略の 1 つに，ソフトウェアプロダクトラインエンジニアリング(SPLE)がある[1]．SPLEでは，個々の製品開発のエンジニアリングを扱うのではなく，共通性を持つ一連の製品シリーズを定義して，再利用可能な共通のコア資産から個々の製品バリエーションを開発する手法である．コア資産を再利用することで，品質も向上することが報告されている[2].

我々は，モデル駆動開発(MDA)をSPLEに応用し，ドメイン知識を形式化したコンポーネントモデルとして定義してコア資産とし，製品開発においてそれを再利用することに取り組んできた[3]．この取り組みで，我々はハンドコーディング無しでコンポーネントの組合せだけで製品開発ができるようになった．しかし，コンポーネントを中心とした再利用では，製品開発において，コンポーネントの設定が煩雑になり，その設定を別の製品開発で再利用するのが困難になる．これは，コンポーネントが製品の特徴(フィーチャ)と直接対応しておらず, 製品間の 1 つのフィーチャの違いに対して，複数のコンポーネントの設定が影響を受けることに起因している．

こうした問題を解決する手法として，フィーチャ指向開発がある[4]．フィーチャとは製品の特徴をユーザ視点やシステムの視点で捉えたもので，製品のバリエーションを表現するのにコンポーネントより適している．例えば，ハイブリッドシステムと通常のエンジンシステムでは 1 つのフィーチャの違いでしかないが，現実には多くのコンポーネントに影響を与える．本稿で，我々は，これまでコンポーネントの組合せで表現されていた製品のソフトウェア構成を，フィーチャの組合せで表現することで，より効率良くコア資産や過去の製品の設定を再利用する方法を提案する．

2 章では，従来のコンポーネントベースの再利用における課題を整理する．3 章では，フィーチャを使って製品の実績を蓄積して，次の製品開発に役立てるためのバリエーション管理の方法について提案する．また我々は，過去に実際に開発した製品実績に，この管理方法を適用してみて，どのように製品実績が蓄積されていくかを調べた．このことについて，4 章で述べる．5 章では，この結果について考察する．

[†](株)デンソー ボデー機器事業部技術企画室
DENSO CORPORATION
BODY ELECTRONICS COMPONENTS PRODUCT DIVISION TECHNOLOGY PLANNING CENTER

2. 課題

コンポーネントをベースとした再利用の課題には，大きく 2 つの課題がある．1 つは，コンポーネントの設定に関するもので，もう 1 つは評価・テストに関するものである．

設定に関する課題は，さらに設定の再利用の課題と，設定の粒度の課題に分けられる．製品開発におけるコンポーネントの設定は，コンポーネントの追加，削除，コンポーネント間の接続，プロパティ設定によって行われる．このとき，同じ機能を実現するのにも，製品毎に異なる設定が可能な場合がある．例えば，ある機能が不要な場合，その機能に関するコンポーネントを全て削除することも可能であるが，その機能の動作に必要な入力を全て無効化することもできる．このように製品開発では，コンポーネントの設定にある程度の自由度がある．したがって，製品間で同じフィーチャ構成にもかかわらず，コンポーネントの設定が異なり，設定の比較，再利用が困難になる．これが，設定の再利用の課題である．もう 1 つの設定の粒度の課題は，製品のバリエーション間の違いの粒度，すなわち，フィーチャの粒度に対して，コンポーネントの粒度が小さいことである．我々のコンポーネントでは横断的な関心事を捉えることができないため，横断的なフィーチャに対して複数のコンポーネントが影響を受けてしまう．そのため，コンポーネントの設定が煩雑になる．

評価・テストに関する課題には，コンポーネントの事前の評価に関する課題と，製品開発における組合せの評価の課題がある．コンポーネントは再利用を前提に開発されるので，事前に使われ方を想定して評価しておかなければならない．しかし，現実的には,全てのコンポーネントの組合せに対して事前に評価することは困難であるし，膨大な工数がかかる．したがって，製品開発でも，コンポーネントの組合せの評価を実施しなければならない．このとき，過去に実績のある組合せを特定することができれば評価工数を低減できるが，前述の通り，コンポーネントは粒度が小さく，組合せを特定するのが困難である．したがって，製品開発で過去の評価実績を再利用することが困難で，評価工数を下げることができない．

3. 提案

前述の課題は，結局のところ，コンポーネントの単位では製品のバリエーションに対して情報量が多すぎて扱いにくいことが原因である．そこで，まず我々は，コンポーネントの設定情報をフィーチャの選択情報へ変換するためのマッピングを定義する(図 1，図 2)．フィーチャは，機能構成，デバイス構成，仕向け地等で製品を特徴付けるものに着目し，図 1 のようにツリー構造の表で表現する．その各フィーチャに対して，どのコンポーネントが必要になるかを示したマッピング表が図 2 である．このフィーチャとコンポーネントのマッピングは，コンポーネントの再利用の手順を示すマニュアルとして活用する．これによって，コンポーネントの設定の再利用が可能になり，同じフィーチャ構成のときに異なるコンポーネントの設定になることを防げる．ここでは詳しく述べないが，このフィーチャモデルを形式的なモデルとして表現すれば，フィーチャの選択によって自動的にコンポーネントの設定をすることも可能になる．

この表によって，これまでコンポーネントの設定によって表現されていた製品のソフト構成を，フィーチャの選択によって表現できるようになった．次に我々は，このフィーチャの選択情報を使って実績を管理することを提案する．従来の製品開発では，ベースとなる製品から設定を引継ぎ，差分となる箇所に着目して開発していたため，ベースとなった製品以外の設定や評価内容を再利用できなかった．製品の実績を蓄積し，管理することで，従来の 2 製品間の差分ではなく，プロダクトライン全体の実績との差分から新規点を抽出できるようになる．図 3 に従来の差分開発と実績管理された開発の模式図を示す.右側の実績管理された開発の太枠は，プロダクトラインのスコープ全体を示す．スコープが変化するのは，新規フィーチャが追加された場合である．このように過去の評価実績を再利用し，着目すべき新規点をあきらかにして評価工数を低減するのが狙いである．

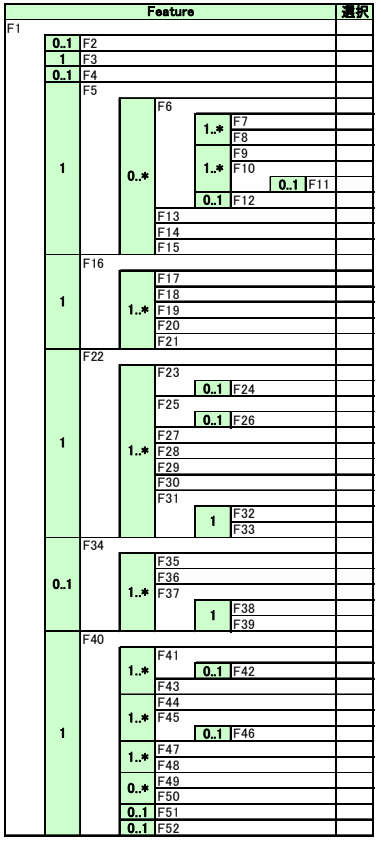


図 1 フィーチャモデル

次に，この実績管理の具体的な方法について述べる．製品の実績は，フィーチャの組合せによる総当り表(図 4)を使って表現する．表の行・列の先頭には，各フィーチャの名前が現れる.そして,その製品で選択されているフィーチャが黄色で示される．表の各セルには行・列のフィーチャの選択情報に基づき数字を記入する．0 は行・列のフィーチャがどちらも選択されていない状態，1 は列のフィーチャのみ選択されている状態，2 は行のフィーチャのみ選択されている状態，3 は行・列のフィーチャがどちらも選択されている状態を表す．このような表を各製品ごとに作成し，実績を蓄積し，新規のフィーチャの組合せのセルを桃色で塗りつぶす．図 4 の左側は，最初の製品の実績で，全ての組合せが新規点である．右側の図は，2 製品目で，セルの値が 1

Vol.2010-SE-170 No.11
2010/11/12

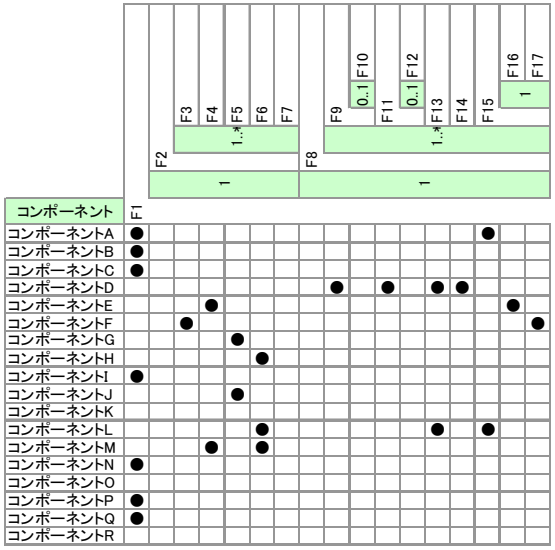


図 2 フィーチャ - コンポーネントのマッピング

製品目と同じ部分が白抜きになる．これを 3 製品目以降も繰り返すことで，桃色のセルが少なくなっていく．

このように 2 つのフィーチャ間の組合せを総当り表で表現することで，製品の新規点を可視化する．この場合，図 3 のスコープは全フィーチャの任意の 2 つの組合せ全体として定義される．このフィーチャの組合せ実績を実績リポジトリに蓄積し，実績リポジトリに蓄積されたフィーチャの組合せ実績と新規製品のフィーチャの組合せを比較すれば，容易に新規点を特定できる．

4. 実験

本章では，3 章で提案した実績管理法を実際に適用すると，どのように製品実績が把握されるかを実験的に確認する．この実験の対象として，2008 年から 2009 年にかけて実際に開発された製品から無作為に 10 製品を選んだ．この製品は，実際には 300 以上のフィーチャで構成されるが，今回は，その中の 50 程度のフィーチャに絞って実験を行った．機密管理上，名称は伏せてあるが，図 1 および図 2 が，そのフィーチャと対応するコンポーネントの抜粋である．

4.1 実験手順

以下の手順で実験を実施した．

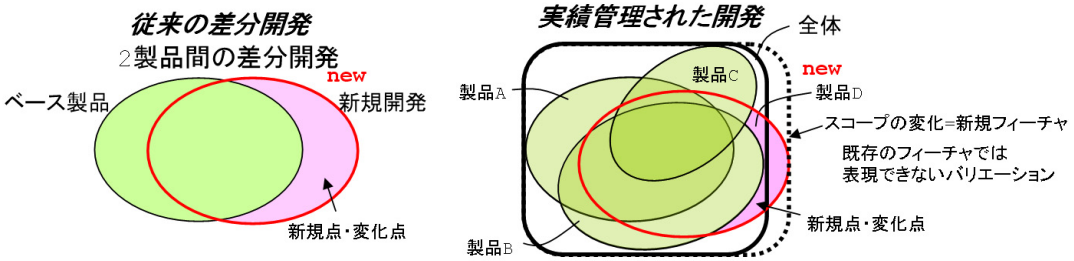


図 3 従来の差分開発と，実績管理された差分開発

(1) フィーチャとコンポーネントのマッピング定義

製品バリエーションを表現するためにフィーチャモデルを作成しなければならないが，この製品では，実績管理こそしていないが，コンポーネントの設定マニュアルとして，すでにフィーチャを特定し，コンポーネントとのマッピングを定義してあった．フィーチャは，機能構成・デバイス構成・仕向け地等で製品を特徴付けるものに着目して識別した．

(2) コンポーネント構成からフィーチャ構成への変換

[3]のコンポーネントベースの開発環境では，製品の構成モデルから，コンポーネント構成のレポートを作成できる．この機能を使って，構成レポートを出力し，図 2 の対応表を基に製品のコンポーネント構成をフィーチャ構成に変換する．

(3) フィーチャ総当り表の作成と製品実績との比較

3 章で提案した方法に基づき，フィーチャの総当り表の各セルに数字を記入する．1 製品目は，比較する製品実績がないため，0 以外のすべてのセルを新規点として，2 製品目は，1 製品目と各セルの数字を比較し，違いがあるセルだけを新規点とする．3 製品目は，1 製品目・2 製品目と比較する．同じ手順で 10 製品目まで総当り表を作成する．

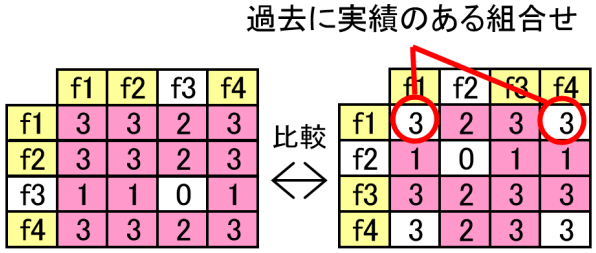


図 4 フィーチャの総当り表

4.2 結果

製品毎のフィーチャ構成の違いがどのように現れたか,結果を図 5，図 6 に示す．1 製品目から順に，9 製品目までを各図の(a)から(i)に示してある．10 製品目は，偶然にもその前の製品にまったく同じフィーチャ構成であったため新規点は見つからなかった．図は総当り表なので対称性がある．この図を見ると，実績が蓄積されていないうち(3 製品目まで)は，新規点は帯状に現れていることがわかる．それが，実績が増えるにつれてまばらになり(6 製品目)，9 製品目になると点状に現れることがわかる．
図 7に，製品展開数と新規点の関係のグラフを示す．横軸に製品数(開発順)，縦軸にその製品で現れた新規の組合せのセル数と，取り得る全組合せのうち，実績がないものの数を示している．このグラフを見ると，新規点の数は，全体としては 0 に向かって収束しているが，単調減少ではなく，ところどころで増加していることがわかる．例えば，2 製品目よりも 3 製品目の新規点の数が多くなっているが，これを詳しく調べてみると 3 製品目は 2 製品目までと比べて高機能な製品であることがわかった．同様に，7 製品目も新規点が増えているが，これまでにない特別仕様の車両の製品であった．

一方，図 7のグラフでは，実績のない組合せの残数は，0 ではないところに収束していることがわかる．これは，使われないフィーチャの組合せ，排他的なフィーチャの存在を示唆している．

5. 考察

本章では，実験の結果から，我々の提案する手法の有効性を検証する．本手法の狙いは，製品実績を活用して評価工数を低減することにある．従来の差分開発では，ベースとなる製品との差分に着目するので，新規点は実験の 2 製品目と同様なものになる(図 5の(b)). 従来の開発では, これと同程度の新規性が継続して発生してしまうが，実績が蓄積され数が増えるにつれて新規点の数は 0 に収束する．しかも，新規点は帯状ではなく，点状に現れるため，新規に評価するべき部分をピンポイントで特定できるようになる．今回は，一部のフィーチャに絞って実験したため，フィーチャの数が増えるとどうなるかはわからないが，本実験では 5, 6 製品程度の実績が蓄積されると評価ポイントを絞り込めることがわかった. このように, 製品実績を蓄積することは，製品での評価工数を大幅に低減することが可能であると考える．また，図 7のグラフでは，使われないフィーチャの組合せの存在が示唆されている．このことから，あらかじめ全ての組合せを評価することよりも，製品開発の都度，その新規点を評価する方が無駄な評価をしなくて済む分，評価効率が良いと考えられる．もちろん，必ず一緒に使われるフィーチャや排他的なフィーチャを統廃合や抽象化することで，使われない組合せを少なくすることができるが，事前にそれを行うには深い分析が必要で容易なことではない．むしろ実績を蓄積することで，その情報をフィーチャを洗練するために活用することができる．

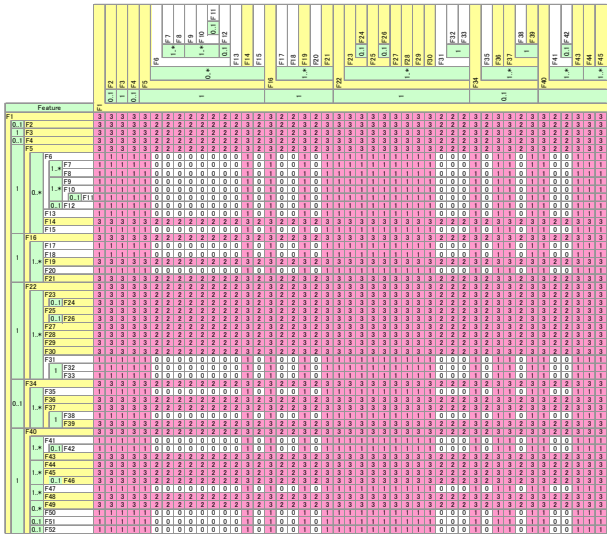
しかし，このように 2 つのフィーチャの組合せの実績があるからといって，評価を省略することが本当に可能であるかについては議論の余地がある．3 つ以上の組合せを調べれば，さらに多くの新規点が見つかるであろう．この点については，2 機能間の組合せのテストが有効であることが知られており([5][6])，2 つのフィーチャの組合せを管理していれば十分な網羅性が確保できると期待される．将来，フィーチャをモデル化して実績を蓄積すれば，3 つ以上のフィーチャの組合せについても特定することもできるであろう．

6. まとめと今後の予定

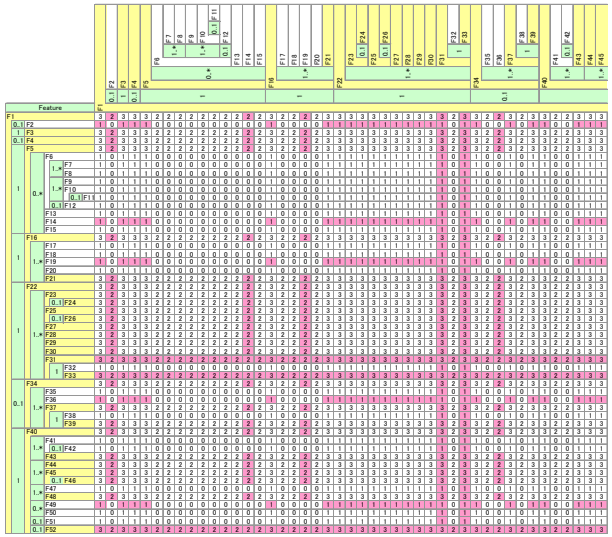
本論文では，フィーチャの組合せによって製品実績を管理する方法について述べた．これによって，製品の新規性を可視化し，評価すべきポイントを特定できるようになると考える．しかし，この方法にもまだ検討すべき課題は多い．その 1 つはフィーチャの保守の問題である．製品仕様の変更で既存フィーチャに影響があった場合，既存のフィーチャに変更を加えてしまうと過去の実績が使えなくなる．実績を活用するためには，いかなる仕様の変更に対しても，フィーチャの追加，すなわち，スコープの変化として扱っていかなければならない．このようにフィーチャを維持・管理するための運用方法を決めていく必要がある．同様に，フィーチャとコンポーネントのマッピングの保守の問題も残っている．

今後は，こうした課題を検討するとともに，フィーチャを形式的なモデルで定義して，フィーチャの選択情報からコンポーネントの設定情報への変換を開発環境でサポートする予定である．フィーチャとコンポーネントの対応関係を完全に形式的に定義することができれば，コンポーネントの使い方やソフトウェア・アーキテクチャといった関心事を製品開発から分離し，製品開発ではフィーチャ(製品の特徴)に注力できるようになる．

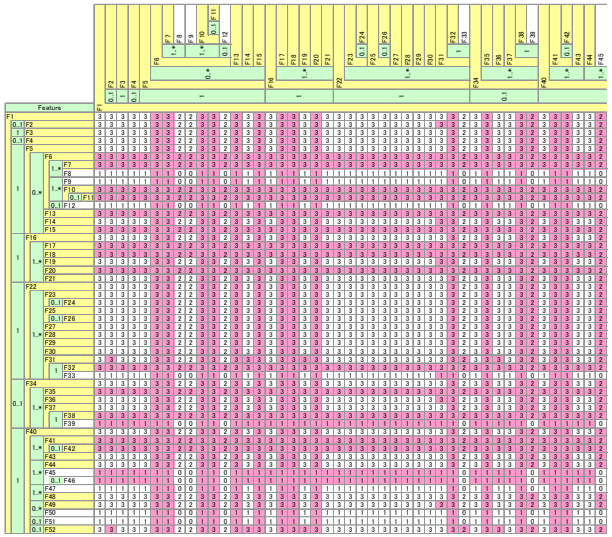
Vol.2010-SE-170 No.11
2010/11/12



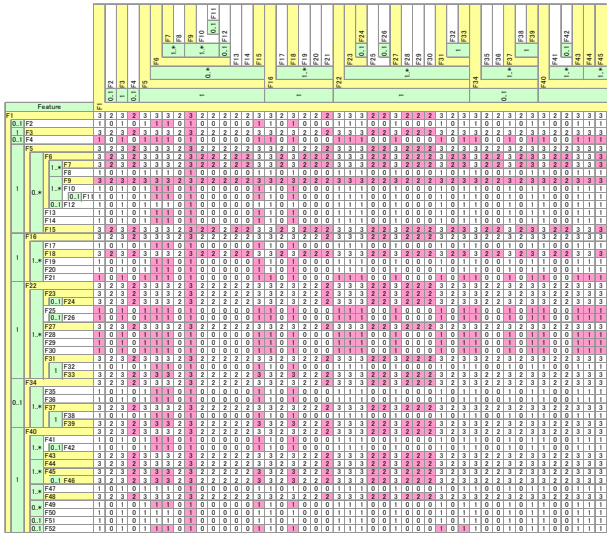
(a) 1 製品目



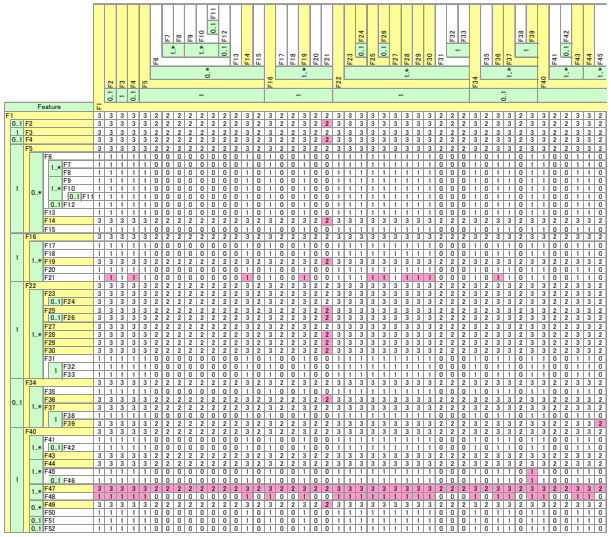
(b) 2 製品目



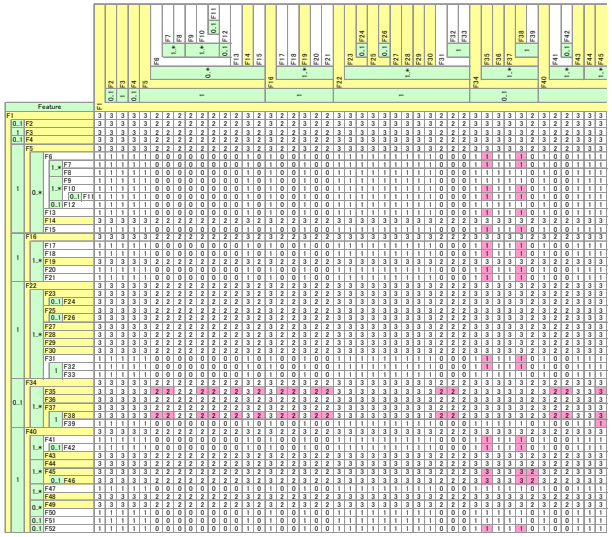
(c) 3 製品目



(d) 4 製品目

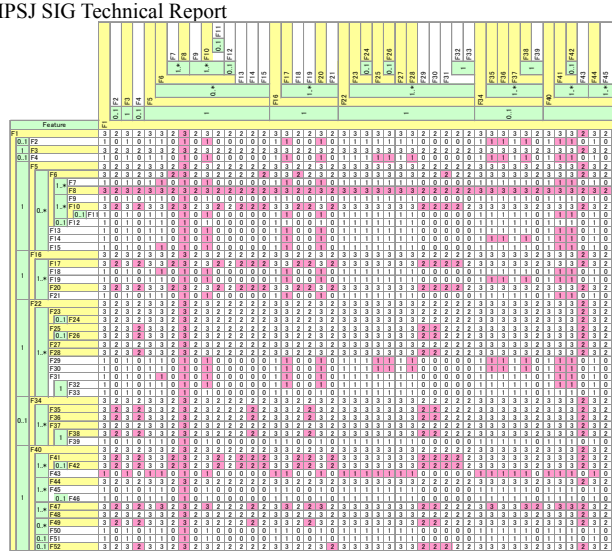


(e) 5 製品目

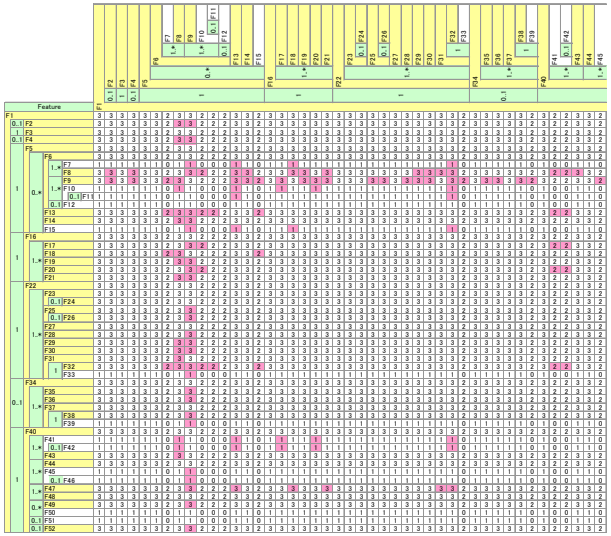


(f) 6 製品目

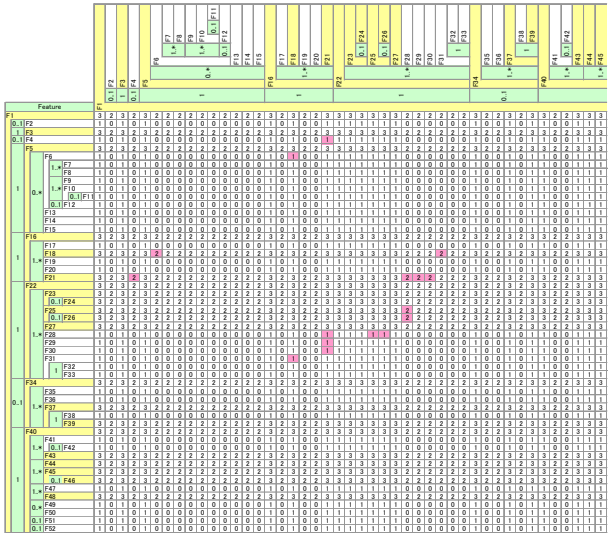
図 5　製品のフィーチャ実績の総当り表



(g) 7 製品目



(h) 8 製品目



(i) 9 製品目

図 6　製品のフィーチャ実績の総当り表(つづき)

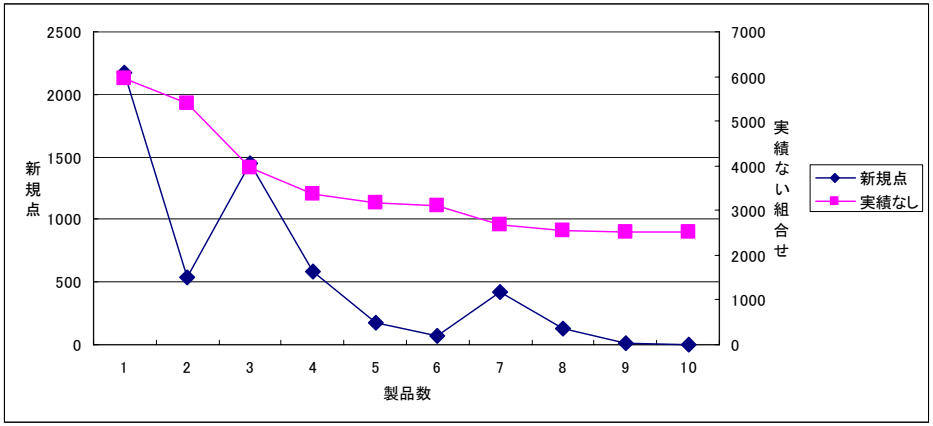


図 7　製品展開数と新規点の関係

参考文献

1)　Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns, Addison-Wesley, 2001. 前田卓雄訳: ソフトウェアプロダクトライン，日刊工業新聞社 (2003).

2)　Cohen, S.: Product Line State of the Practice Report, Technical Note CMU/SEI-2002-TN-017, SEI, Carnegie Mellon University (2002).

3)　加藤 滋郎，後藤 祥文: 組込みシステムのプロダクトラインにおける MDA，情報処理学会シンポジウム論文集，Vol.2007，No.8, pp.54-63 (2007).

4)　Kang, K., Kim S., Lee, J., Kim, K., Shin, E., Huh, M.: FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering Volume 5, pp.143-168 (1998).

5)　Cohen, D.M., Dalal, S.R., Fredman, M.L., Patton , G.C.: The AETG System: An approach to Testing Based on Combinatorial Design, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL.23, No.7 (1997).

6)　吉澤 正孝，秋山浩一，仙石太郎: ソフトウェアテスト HAYST 法入門，日科技連出版社 (2007).