近似的モデリングアーキテクチャに関する考察

岸知二[†1]　川嶋優樹[†2]　野田夏子[†3]

ソフトウェアモデリングは、プロダクトライン開発、モデルベースド開発、形式検証などに広く活用されるが、モデルの規模が大きくなると共に、より厳密で整合性のとれたモデル群の構築が求められるため、モデル構築や維持のコストが大きくなる傾向がある。我々は、モデル群の構造をモデリングアーキテクチャとして捉え、それをどういうコストバランスで作るかの判断する手法について検討している。ここではまずモデリングアーキテクチャに関するメトリクスを用いて構造上の特徴を推定し、その特性に合わせたコストでのモデル構築をするために必要に応じて近似的なモデリングを行う。本稿では可変性管理を対象に手法の全体像を説明する。

A Study on Approximate Modeling Architecture

TOMOJI KISHI[†1]　YUKI KAWASHIMA[†2]
NATSUKO NODA[†3]

Advanced use of software modeling such as those in product-line development, model-based development, and formal verification require large, complicated, precise and consistent models. As a result, the cost of model development and maintenance are becoming too expensive. We are currently examining a method that supports deciding cost tradeoff for software modeling based on the “modeling architecture”, the software structure that affects software modeling. In our method, we firstly estimate characteristics of the modeling architecture utilizing metrics, and apply approximate modeling if necessary. In this paper, we overview the method in the context of variability management.

1. はじめに

プロダクトライン開発 3)、モデルベースド開発 2)、形式検証など、ソフトウェア開発へのソフトウェアモデリング 13)の活用が広がっているが、それに伴いモデルが大規模化、複雑化、厳密化し、モデルの構築や整合化コストが大きくなっている 7)8)。モデリングに無制限にコストをかけられないため、限られたリソースを有効に活用することが求められる。

そうした背景を踏まえ、我々は近似的モデリング手法の検討を進めている 9)10)。この手法は本来の意図通りのモデリングが高コストと判断される際に、一定の近似化を行うことでモデル構築や整合化のコストを減少させることを狙っている。一方、近似化することによるペナルティも発生するため、その適用について何らかの指針が必要となる。

本稿では、モデリングに影響を与えるモデル群の構造をモデリングアーキテクチャとして捉え、その特性を把握することで近似的モデリングの適用方法を判断するための手法を提案する。具体的にはプロダクトライン開発における製品導出を題材に、モデリングアーキテクチャに関するメトリクス等から特性を把握し、その傾向に応じてモデルを近似化すべきか、どう近似化すべきかの判断を支援する手法を提案する。なお本稿では、定性的な問題分析と手法の提案を行う。評価は今後の課題である。

2 章では我々の検討してきた近似的モデリングの概要を述べる。3 章では本稿で扱う問題を述べる。4 章では製品導出と近似モデルを用いた製品導出について説明する。5 章で問題の分析と考察を行い、それを踏まえて 6 章で提案手法を述べる。さらに 7 章では関連する議論を行い、8 章で本稿を締めくくる

2. 近似的モデリング

我々は近似的モデリングについて、基本的な定義や考え方、また近似的モデリングの適用を考える際の概念的な枠組みについて提案してきた 9)10)。以下、本稿に関わる内容について、簡単に紹介する。

ソフトウェア開発のために何らかの形式性に基づいて作られる記述をソフトウェアモデル（以下モデル）、モデルの構築やモデル間の整合化の作業をモデリングと呼ぶ。近似的モデリングは、当初意図したモデリングが高コストになると判断される際に、次善の策として、一定のペナルティを承知でモデリングコスト削減のために適用することを意図している。

近似モデルは、意図するモデルを近似的に表現したモデルであり、近似モデルを利用したモデリングを近似的モデリングと呼ぶ。近似方法には記述の抽象度を上げる概略化と、記述する対象範囲を狭める局所化との 2 種類がある。いずれもモデルの構成要素が減少するので直接的なモデリングコストは減少する。一方表現されるモデルの詳細度が低下したり対象範囲が狭まったりするため、活用においては後工程で情報を補うなどのペナルティが発生する。

†1 早稲田大学
Waseda university.
†2 (元)早稲田大学
(Ex.) Waseda university
†3 芝浦工業大学
Shibaura Institute of Technology

一般にソフトウェア開発では複数のモデルを利用する。近似的モデリングの適用に際しては、近似化によるコスト減と、ペナルティによるコスト増とのトレードオフを考えながら、複数のモデルのどれをどう近似化することが全体として有用かを検討する必要がある。そこでモデルマップという図法を使って開発に関わるモデル群を俯瞰してモデリングの戦略を検討するための概念的な枠組みを提案した。

3. 問題

これまでの基本的、概念的な検討を踏まえ、本稿では具体的なモデリングの利用局面を対象に、近似的モデリングの有効性や活用法について検討する。具体的には、プロダクトライン開発における製品導出を対象とし、以下を本稿での問題とする。

- 近似的モデリングが有効な状況があるかどうか。あるとすればそれはどのような状況か : 製品導出に関わるモデル群がどのような状況にあるときに近似的モデリングが有効と考えられるか、その状況を明らかにする。
- 有効な状況があるならば、その状況をどう把握するか : モデリングを行っている際に、現在のモデル群に近似的モデリングの適用することが有効なのかどうかをどうやって把握するのか、状況把握の方法を明らかにする。
- 状況が把握できた際に、どのような近似方法が有効かをどう判断するか : 把握された状況において、概略化と局所化の 2 種類の近似モデルをどう適用することが有効そうか、判断方法を明らかにする。

本研究は大規模なモデリングをどう進めることがよさそうか、モデリングの戦略付け、方針決めを支援する手法を提案することを意図している。従って厳密かつ詳細に状況を解析することなく、比較的簡易な方法で状況を把握しモデリングの指針を得ることを狙っている。

4. 製品導出と近似的製品導出

本章では、まず製品導出ならびに近似モデルを利用した製品導出について説明する。

4.1 製品導出

本稿で対象とする製品導出が、どのようなモデルを利用してどのように行うことを想定しているかを説明する。

4.1.1 フィーチャモデル

フィーチャモデル(FM)はプロダクトラインが持つ共通あるいは可変なフィーチャ群を階層的に表すモデルである 5)。本稿では図 1 に示す記法を用いる。図 2 はこの記法を用いたフィーチャモデルの記述例である。なおルートから必須の関係だけで辿れるフィーチャを共通フィーチャ、辿れないフィーチャを可変フィーチャと呼び、共通フィーチャにハッチングをつけている。

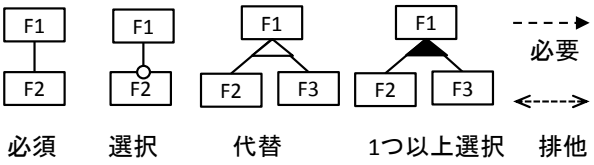

図 1　フィーチャモデルの記法

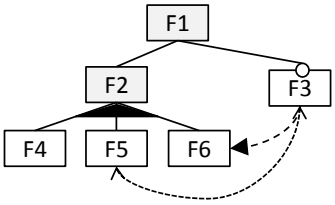

図 2　フィーチャモデルの例

4.1.2 アーキテクチャモデル

本稿でのアーキテクチャモデル(AM)は可変性の観点からのアーキテクチャの記述である。可変性を持たせたアーキテクチャ記述に関してはいくつかの提案があるが 4)12)14)、ここでは可変性の側面に注目してフィーチャモデルと類似な記法で表す。

図 3 は AM の記述例である。角丸四角はコンポーネントを示し、可変点を持つ場合にそこに当てはまり得るバリアント（製品毎に変わり得る可変なコンポーネント）を選択や代替として示す。またあるバリアントを利用する際に必要なバリアントや、同時に使えないバリアントの関係を必要や排他として示す。また共通コンポーネントにはハッチングをつけている。

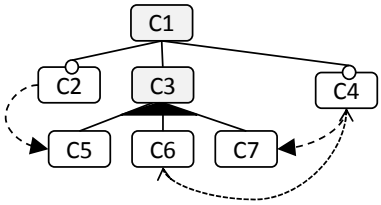

図 3　アーキテクチャモデルの例

4.1.3 トレーサビリティリンク

FM 中の可変フィーチャと AM 中のバリアント間にトレーサビリティリンク(TL)を定義できる。TL はフィーチャの実現に必要なバリアント群を示し、フィーチャ単位に対応するバリアント群を図 4 のように記述する。

```
F3→{C4}
F4→{C6}
F5→{C5, C6}
F6→{C6, C7}
```

図 4　トレーサビリティリンクの例

4.1.4 製品導出

本稿では、フィーチャの集合 FS0 が与えられたとき、それらを持った製品を実現するために必要なコンポーネントの集合を決定する作業を製品導出と呼ぶ。製品導出は以下のステップから構成される。

1. フィーチャ構成導出 : FS0 を含み、FM の制約を満たしたフィーチャの集合 FS1 を決定する。この手続きには、後述するセレクティビティ駆動製品導出(SDPD)を用いることを想定している。FS1 は複数存在することもあるがその場合はその中の 1 つを導出する。また FS1 が存在しない場合は、製品導出は失敗する。
2. TL 追跡 : FS1 中の可変フィーチャと TL で結ばれたバリアントの集合 VS0 を決定する。FS1 中の可変フィーチャに対する TL が定義されていなければそのフィーチャ構成は実現不可能なのでステップ 1 に戻り他のフィーチャ構成の導出を試みる。
3. バリアント構成導出 : アーキテクチャ設計上での判断があれば、製品に含めるバリアントの集合を VS0 に付け加える。その上で、VS0 と AM に対して SDPD を適用し、VS0 を含み AM の制約を満たしたバリアントの集合 VS1 を決定する。VS1 が複数ある場合はその中の 1 つをバリアント構成として導出する。存在しない場合は、FS0 は実現不可なので、ステップ 1 に戻り他のフィーチャ構成の導出を試みる。

例えば。FS0={F4}のとき、FM の制約を満たす集合としては{F4}、{F4, F5}、{F4, F6}, {F4, F5, F6}, {F3, F4, F6}などが考えられる。FS1={F4}を選ぶと、VS0={C6}となる。ここで設計判断として C2 も必要と考えれば VS0={C2, C6}となり、{C2, C5, C6}あるいは{C2, C5, C6, C7}が AM の制約を満たすのでどちらかを VS1 とする。これに共通コンポーネントを加えると製品に必要なコンポーネントが特定される。

SDPD は与えられたフィーチャの集合を含み FM の制約を満たすフィーチャの集合をひとつ特定するための手法である 1)11)。与えられたフィーチャの集合から、フィーチャをひとつ選び、それが必要とするフィーチャ群を特定するとともに、それを選んだことによって排他関係や代替関係のために選べなくなるフィーチャ群を FM から除去するという手順を繰り返して、最終的なフィーチャの集合を求める。本稿では AM も FM と同じ構造を持っているので、バリアント集合を求める際にも SDPD を適用する。

製品導出のステップ 2 やステップ 3 からステップ 1 に戻った場合には、実現不可と分かったフィーチャ構成でないことを手順の中で確認することで、同じフィーチャ構成の導出を避けることができる。

4.2 フィーチャモデルの近似

本稿では、FM 側の近似化のみに焦点をあてる。具体的には以下の 2 つの近似化を考える。

- 概略化：直感的にはフィーチャ階層の下部を省略する近似化である。可変フィーチャのリストを与え、そのメンバの子孫フィーチャをすべて除去することで得られる。子孫が省略されたメンバには、ステレオタイプ《概略》をつける。この際、子孫フィーチャが参加する必要、排他は、祖先フィーチャ（リスト中のフィーチャ）に付け替える。
- 局所化：直感的にはいくつかの可変フィーチャを必須化することで選べなくなったフィーチャ群を除去する近似化である。可変フィーチャのリストを与え、それらを順次選択して不要なフィーチャや関係を削除する。またリストのメンバにはステレオタイプ《局所》をつける。

図 5 は 図 2 のフィーチャモデルを近似化した例である。左はリスト(F2)を与えて概略化した例であり、右はリスト(F5)を与えて局所化した例である。概略化された FM においてステレオタイプ《概略》のついたフィーチャに接続される必要や排他は、そのフィーチャそのものではなく、その子孫フィーチャとそういう関係があり得るという可能性を示すものであるので、図のように同一のフィーチャから必要と排他が結ばれることもありうる。一方局所化された FM は通常の FM と同様の記法、解釈となる。

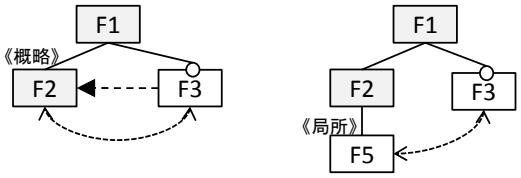

図 5　概略化と局所化

概略化と局所化は排他的なものではなく、局所化して概略化したり、概略化して局所化したりするなどの組合せも考えられる。

4.3 近似化された FM による製品導出

近似化された FM を利用した製品導出の方法を以下に説明する。

概略化 FM に対する製品導出は以下のようになされる。まず、削除されたフィーチャの TL を《概略》フィーチャに付け替える。図 5 左のような概略化を行った場合、図 4 の TL では F4, F5, F6 がすべて F2 に置き換えられる。製品導出はフィーチャの集合 FS0 に対して、以下のように行われる。なお概略化されたフィーチャは指定できないので、例えば FS0={F2}といった粗い指定しかできない。

1. FS0 を含み FM の制約を満たしたフィーチャの集合 FS1 を決定する。この際、《概略》フィーチャの参加する必要と排他は無視する。これとは別途に FS1 中の《概略》フィーチャと必要で結ばれているフィーチャ

の集合 FS1r と、排他で結ばれているフィーチャの集合 FS1e を得る。FS1 が存在しない場合は、製品導出は失敗する。

2. 3つのフィーチャの集合それぞれから TL を辿りバリアントの集合 VS0, VS0r, VS0e を得る。FS1 中の可変フィーチャに対する TL が定義されていなければステップ 1 に戻り他のフィーチャ構成の導出を試みる。
3. VS0r と VS0e を参考に設計上必要となるバリアントを VS0 に加え、VS0 を含み AM の制約を満たしたバリアントの集合 VS1 を決定する。VS1 が存在しない場合には、ステップ 1 に戻り他のフィーチャ構成の導出を試みる。

FM が概略化されたので FM 上では粗いフィーチャ構成導出しかできなくなるため、AM 上でバリアント構成を導出する際により多くのバリアントを指定して最終的なバリアント構成を得ていることになる。直感的には、近似化しなければステップ 1 で行っていた作業の一部を、ステップ 3 に移行させていることになる。ただしフィーチャの指定をバリアントの指定に置き換える作業は意味的な人手作業であり、一般には近似化によって総コストが増大する。

局所化 FM に対する製品導出では、削除されたフィーチャの参加する TL は削除する。図 5 右のような局所化を行った場合、図 4 の TL では F4 と F6 からの項目が削除される。製品導出のステップは 4.1.4 と同様だが、削除されたフィーチャは指定できなくなるので指定できるフィーチャ集合は少なくなり、結果として導出可能な製品の数が減る。

5. 問題分析

製品導出を目的とした FM や AM のモデリングにおいて、製品導出コストに照らして FM に対する近似的モデリングの適用が有利と考えられるのはどのような状況か、モデリングアーキテクチャの観点から検討する。

5.1 製品導出のモデリングアーキテクチャ

図 6 は製品導出に関わるモデルをモデルマップ 8)として表現したものである。

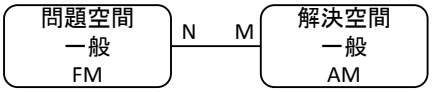

図 6 モデルマップ

角丸四角はモデルを表し、その中にモデルの開発における位置づけ、一般的モデルか例示モデルか、モデルの種類を記述する。モデリングにおいてモデル間の概念の対応を取り扱う際には線を引き、概念間の多重度を示す。FM と AM のフィーチャとバリアントの間には一般に多対多の関係がある（クロスカッティングしている）ことが示されている。なおモデリングのコストは、モデルやモデル間の関係の規模や複雑度と相関を持つと考えられる。

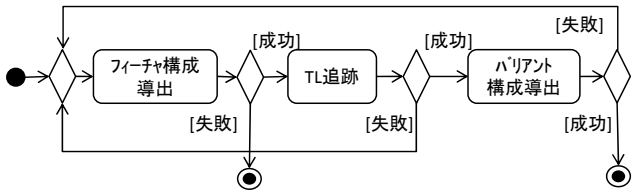

図 7 製品導出プロセス

図 7 は製品導出のプロセスをアクティビティ図で示したものである。この中でフィーチャ構成導出とバリアント構成導出は、モデルの規模が大きくなると組み合わせ的に高コストとなる。従って何度もこの作業を繰り返すことは不利であり、それが近似的モデリングのペナルティとなる。

5.2 近似的モデリングが有効と考えられる状況

上記のモデリングアーキテクチャを踏まえ、近似的モデリングが有効と考えられる局面について考察する。

5.2.1 モデルの構築コストの削減

FM を近似化すればモデリングのコストは減るが、そのペナルティとして、導出されるフィーチャ構成が粗くなり、バリアント構成導出作業が高コストになる。厳密なコストバランスはケースバイケースであるが、一般には上流で行うべき判断を下流に持ち越すことはコスト的に不利である。

しかしながら、FM が過度の情報量を持っている場合は、近似化してもそのペナルティは少ない。これは FM から導出可能な構成の数が、AM から導出される構成の数に比べて大きいときに発生する。例えば、AM 中のコンポーネントが多くの機能を持つ粗粒度なものであるような状況である。この場合は、FM 側で細粒度なフィーチャを定義し、きめ細かなフィーチャの構成を導出しても、結局同じバリアントの構成にマッピングされたり、あるいは対応するバリアントの構成が存在しなかったりという状況が起りやすくなる。つまり AM に比べて FM が詳細すぎる状況である。

図 8 はこの状況を模式的に例示したものである。一つのバリアントと複数のフィーチャが TL でつながっているため、FM 側での複数のフィーチャ構成が、AM 側の少数のバリアント構成で実現されている。

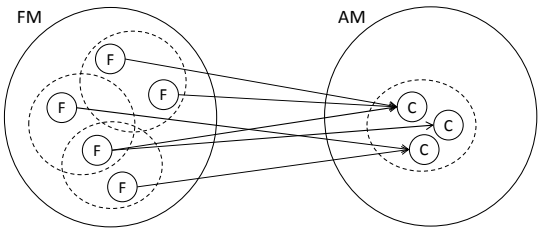

図 8 AM が粗粒度な状況

こうした状況をマクロに判断する指標としては、FM と

AM からの導出可能構成数の比率や、より簡易にはモデル要素数の比率などが考えられる。この状況では、FM を概略化して FM 側を粗粒度にすることによって、過度にきめ細かなフィーチャを選ぶという無駄を省くことができる。

5.2.2 TL 追跡失敗の軽減

フィーチャ構成が導出できても、それに含まれる可変フィーチャに対して TL が定義されていなければ、製品導出はできず他のフィーチャ構成の導出を試みる必要が発生する。これは単に TL が定義されていないという状況もあるが、より本質には AM 側が不十分な状況が該当する。つまり FM 側で想定しているだけのコンポーネントが未整備な場合である。この場合は、フィーチャ構成を導出しても TL 追跡が失敗するために再度フィーチャ構成を試みるということが繰り返される危険性がある。

図 9 はこの状況を模式的に例示したものである。AM 中に十分なバリアントが整備されていないため、多くのフィーチャが実現できない。

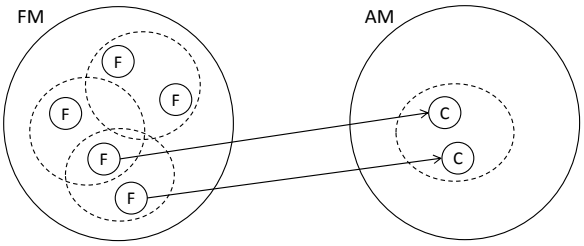

図 9　AM が未整備な状況

こうした状況をマクロに捉えるための指標としては、FM 中で TL リンクの定義されている可変フィーチャ数の比率などが考えられる。この比率が低い場合は、FM を局所化して対応するバリアントがない部分を削除することで無駄を省くことができる。

5.2.3 バリアント構成導出失敗の軽減

TL 追跡が成功しても、対応するバリアント構成の導出が失敗するとフィーチャ構成導出を再度繰り返さなければならない。バリアント構成の導出が失敗するのは、FM 側単独では選択できるフィーチャやフィーチャ構成が、TL を辿って AM 側の制約を考えた場合には選択（実現）できないという状況などで発生する。我々はこういう状況を、FM と AM を結合することによって発生するデッドフィーチャや選べない組み合わせの問題 6)と捉えている。図 10 はこの状況を模式的に例示したものである。FM 側では導出されたフィーチャ構成に対応するバリアント構成が、設計上の排他関係などの理由で選べなくなっている。

こうした状況を把握するための指標としては、我々が過去に提案した結合により発生するデッドフィーチャや選べない組み合わせを推定するメトリクス 6)の利用が有効と考える。こうした状況では、FM 側で詳細なフィーチャ構成を検討するよりも、あえて FM を概略化してその判断を AM 側のバリアント構成の検討に持越し、そこで設計制約を考えながらバリアント構成を決定した方が有利になると考えられる。

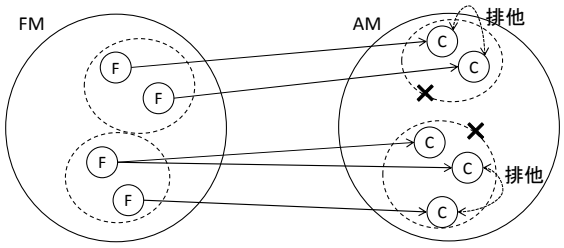

図 10　設計制約が強い状況

6. 提案手法

問題分析を踏まえ、近似的モデリングを適用するかどうかを判断するための手法を以下に提案する。

6.1 概要

本手法は、製品導出を目的に FM, AM, TL のモデリング作業を進めている状況において、FM に対して近似的モデリングを適用するべきかどうかを判断するための指針を与えることを目的とする。

そのために現時点のモデリングアーキテクチャの特性を、メトリクスを使ってマクロに把握し、その状況に応じて FM の近似化が有利そうかどうかを判断するという方法をとる。

6.2 モデリングアーキテクチャの特性把握

近似的モデリングが適した状況かどうかを把握するために、モデリングアーキテクチャに関する以下のメトリクスを利用する。

- 導出製品数：FM あるいは AM から導出可能な製品数。以下、それぞれ#P(FM), #P(AM)と記述する。
- TL 定義率：可変モデル要素（可変フィーチャあるいはバリアント）中で、TL が定義されている要素の割合。それぞれ%L(FM), %L(AM)と記述する。
- デッドフィーチャ率、選べない組み合わせの率：FM と AM を TL で結合することによって、FM 側単独では選択できるフィーチャやフィーチャ構成が選べなくなる率。それぞれ%DF, %DC と記述する。

なお上記の内、%DF と%DC を正確に求めるには計算量が大きく大規模なモデリングではコストが高いため、我々が提案したメトリクス 6)をそれに代替する。

6.3 近似化の判断方法

メトリクスに基づく近似化判断を表 1 に示す。#P(FM)/#P(AM)は FM と AM の導出可能構成数の比率を示す。この導出数比が 1 以下である場合は AM 側の方がより多くの導出可能構成を含んでいるので、FM 側の近似化は

不必要と考えられる。

この比率が 1 より大きい場合は、さらにリンク率%L(FM)を確認する。この値が 1 に近い時は、ほとんどの可変フィーチャに TL が定義されている状況を示す。#P(FM)/#P(AM)の値が大きいのにほとんどの可変フィーチャに TL が定義されているということは、ひとつのバリアントに複数のフィーチャが対応づけられている、つまりコンポーネントが粗粒度の状況（5.2.1）と考えられる。この場合は FM の概略化が有効と考えられる。一方#P(FM)/#P(AM)の値が大きく、TL の定義されている可変フィーチャが少ない場合はコンポーネントが未整備の状況（5.2.2）と考えられる。この場合は FM の局所化が有効と考えられる。

また、%DF や%DC が高い場合は設計制約が強く、モデルの結合によりデッドフィーチャや選べない組み合わせが多い状況（5.2.3）と考えられる。この場合は概略化が有効と考えられる。

表 1 メトリクスに基づく近似化判断

導出数比 #P(FM) / #P(AM)	リンク率 %L(FM)	AM 制約 %DF %DC	FM の近似に関する判断
大	高	高	概略化
		低	概略化
	低	高	局所化および概略化
		低	局所化
小	—	—	不要

7. 議論

本稿では製品導出を対象に、近似的モデリングの適用が有効と考えられる場合を検討し、それがモデリングアーキテクチャのどういう状況に相当するかを考察した。それを踏まえ、現時点でのモデリングアーキテクチャの特性をメトリクスで捉え、近似的モデリングが有効な状況であるか、有効な状況であればどういう近似化が適しているかを判断するための手法を提案した。本手法はミクロな状況を解析的に把握するのではなく、マクロな状況を把握してモデリングの戦略を決めるための利用を意図している。

本手法の特徴は、開発に関わるモデル群の構造上の特徴モデリングアーキテクチャとして捉え、現時点でのモデリングアーキテクチャの特性をメトリクスによって推定することによってモデリングの戦略を判断する点である。FM をどの程度の詳細度でどの程度の範囲を対象とすべきか、という判断は FM だけを眺めていてもできず、他のモデルとの関係の中で判断すべきであるという考え方に基づいている。

本稿では FM や AM を利用した製品導出という対象に焦点をあてた。従って他のモデルや他の利用目的に対しては、今回の手法をそのまま適用することはできない。しかしながら、モデルの表現力やモデルの記述範囲などに注目するという基本的な視点は、今回の対象に限らず有効性があると考えている。

8. おわりに

本稿は、製品導出の方法を定義し、それに対する近似的モデリングの有効性について定性的に検討したものである。本手法の評価は今後の課題である。また実際に適用するためにはどういうメトリクス値であればどの程度の近似化が適しているのかといった、より具体的かつ定量的な指針が必要である。こうした側面についての検討も今後の課題である。

参考文献

1) Sheng Chen, Martin Erwig, “Optimizing the Product Derivation Process”, In Proceedings of 15th International Software Product Line Conference (SPLC2011), pp.35-44, 2011.
2) http://www.omg.org/mda/.
3) http://www.splc.net/.
4) Hassan Gomaa, Designing Software Product Lines with UML, Addison-Wesley Professional, 2004.
5) K. Kang, S. Cohen, J. Hess, W. Novak, A.Peterson, “Feature-Oriented Domain Analysis (FODA) Feasibility Study”, In Proceedings of Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.
6) 川嶋優樹，岸知二: 可変性モデル間の製品バリエーション不均衡に関するメトリクスの提案, 情報処理学会 第 76 回全国大会, 2014(1). pp435-436, 2014.
7) Tomoji Kishi and Kyo-Chul Kang: Scalable Modeling Techniques for Software Product Lines (SCALE 2009), proceedings of SPLC2009, p299, 2009.
8) 岸知二: ソフトウェアモデル間のスケーラブルな整合化戦略について, ソフトウェア工学の基礎 XVIII 日本ソフトウェア科学会 FOSE 2011, pp.97-102, 2011.
9) 岸知二, 近似的モデリング技法についての考察, 情報処理学会 ソフトウェア工学研究会 研究報告, Vol.2012-SE-177, no.34, pp.1-7, 2012.
10) 岸知二: 近似的モデリングメカニズムについての考察, 情報処理学会 ソフトウェア工学研究会 研究報告, Vol.2013-SE-181, pp.1-7, 2013.
11) 永野寛丸，岸知二: ソフトウェアプロダクトラインにおける非機能特性を考慮した製品導出支援手法の提案, 情報処理学会 ソフトウェア工学研究会 研究報告, Vol.2013-SE-182, no.13, pp.1-8, 2013.
12) Haugen, Øystein, Wąsowski Andrzej and Czarnecki, Krzysztof: CVL: common variability language, Proceedings of the 16th SPLC - Volume 2,pp.266-267, 2012.
13) OMG: Unified Modeling Language Specification, Version 2.4, 2010.
14) Norbert Siegmund, Martin Kuhlemann, Marko Rosenmüller, Christian Kästner, Gunter Saake, “Integrated Product Line Model for Semi-Automated Product Derivation Using Non-Functional Properties”, In Proceedings of the International Workshop of Variability Modelling of Software-Intensive Systems (VaMoS), pp.25-31, 2008