March 31, 2009
RT5300
Engineering Technology; Human-Computer Interaction 5 pages

Research Report

A method to update UML model elements by editing the analysis results on a tabular view

Atsushi Fukuda, Takashi Nerome, Hirofumi Matsuzawa, Naoshi Tabuchi

IBM Research - Tokyo
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

モデル分析結果の更新値から入力モデルを編集する手法の提案

福田 敦史　根路銘 崇　松澤 裕史

日本アイ・ビー・エム株式会社

E-mail:　{ atfukuda, nerome, matuzawa} @jp.ibm.com,

キーワード　UML，モデル更新，モデル分析結果，OCL，推薦度

Abstract

In this report, we propose an efficient method of model edit by updating result of target model analysis.　In recent development style such as them of I/T system or automobile, an modeling approach is getting popular from perspective of quality with design visibility and formalization of design.　However, it is getting difficult to edit it because of increasing the number of model element including design essence of development target.

In our approach, we utilize not only OCL technology to conform to model structure by table based editing environment, but also new method of a recommendation analysis.

1. はじめに

近年，IT システム開発や自動車開発において，設計の可視化や形式化を目的として，UML モデリングを利用した設計が重要になってきている．一般的に，設計の進捗に依存して UML モデルのモデル要素数が膨大になってくるがゆえに，設計者にとってモデル要素を理解した編集が困難になりやすいという課題がある．

この課題に対して，モデリングの容易性を高めるために，閲覧したいモデル要素を抽出して表形式で一覧化する技術や，関連要素を辿る技術とその技術を採用したツールがある．

OCL(Object Constraint Language)[1]は，標準化団体の OMG が提唱するモデル制約記述言語であるが，モデルの抽出および分析にも適用可能な言語である．

Eclipse BIRT[2]は，OCL 式に基づきモデルを分析し，表形式で表示する機能を提供している．

しかし，OCL や Eclipse BIRT のような既存技術では，表示される結果を確認することしか出来ず，モデルを直接編集することは出来ない．また，モデリングの容易性を更に高めるためには，表形式を用いたモデルの構造に基づく編集が必要だと考えられる．

本稿では、OCL(Object Constraint Language)式などの集約演算と算術演算によるモデルの表形式での分析及び抽出結果を、ユーザーが求める値に変更することを目的として、分析結果を直接更新可能な環境を提供する技術および手法について述べる．また，更新結果をもとにした入力となるモデル要素の編集を，モデル構造の制約をもとにした推薦箇所の選択により実施する方法について提案する．

2. 本研究で用いる OCL 表記

OCL では，一般的にモデル要素群から必要なモデル要素を抽出するための OCL 抽出式と，その結果抽出された要素群を分析するための OCL 分析式が表現可能である．

OCL 抽出式は、モデル要素の起点から関連要素を抽出する表現となる．例として，self.oclIsType(Class)の表現は，対象モデル要素が Class という Type を保有するかどうかを BOOL 型で返す．一方，分析式は、図 1 に示す BNF をもとに表記される．

AnalysisExpr ::= ***SetExpr*** "->" ***SizeExpr***
SetExpr = ***OwnerExpr*** "." ***FeatureReferenceExpr***
OwnerExpr = "self" | "self" "." [***FeatureReferenceExpr***]+
FeatureReferenceExpr ::= ***FeatureName*** "." ["->" ***FunctionExpr***]*
FunctionExpr ::= ***SetFunction*** "(" ***SetExpr***")" | ***FilterFunction*** "(" ***Predicate*** ")"
SetFunction ::= "union" | "intersection" | "minus"
FilterFunction ::= "select" | "reject"
Predicate ::= "oclIsType" "(" ***TypeName*** ")"
SizeExpr ::= "size()" | "count" "(" ***FeatureName*** ")"
FeatureName ::= メタモデル上の feature の名前 ("ownedPort" など)
TypeName ::= メタモデル上の型の名前 ("InputPort" など)

図 1. OCL 分析式の BNF

また、図 2 の OCL 式の例は，対象となるモデル要素が保有するモデル要素を分析し，その数を抽出することを意味する．

```
self.packagedElement.select(oclIsType(Class))
                    .ownedElement -> size()
```

図 2. OCL 式の例

3. 提案手法

本稿において,我々は表形式で表示されたモデル分析結果に対し，直接編集を行い，これをモデルに反映されるための手法を提案する．

はじめに，次のようなシナリオを想定していただきたい（その概要を図 3 に示す）．

1) 設計者が UML モデルを用いた設計の実施中に，OCL 分析式を用いてモデルの分析を行うとする．

2) 設計者は，分析結果を元に，表中に示された分析結果の値を直接編集する．

図 4 では Class D の集約値を 6 から 7 に変更する．

3) モデリングシステム（モデルエディタ）は、どの型のモデル要素（インスタンス）を追加すべきかをガイドするため，推薦度に基づく優先順位付けされたモデル要素のリストを設計者に対して提示する．

4) 設計者は，表示された優先順位に基づき，適切な要素を選択およびその属性値を入力する．

5) モデルが自動的に更新される．

すなわち，分析表に対して、修正が行われる時、どのように編集すべきかが定義されたモデル要素が、推薦度に基づく優先順位付けされて提示することにより、設計者は適切な要素を選択し、モデリングシステムは選択されたモデル要素の編集方法の定義に従い、モデルを追加することで，モデルの分析結果から直接モデルの編集を行うことが可能となる．

4 章において，モデル分析式を用いたモデル要素の推薦度計算方式について述べる．

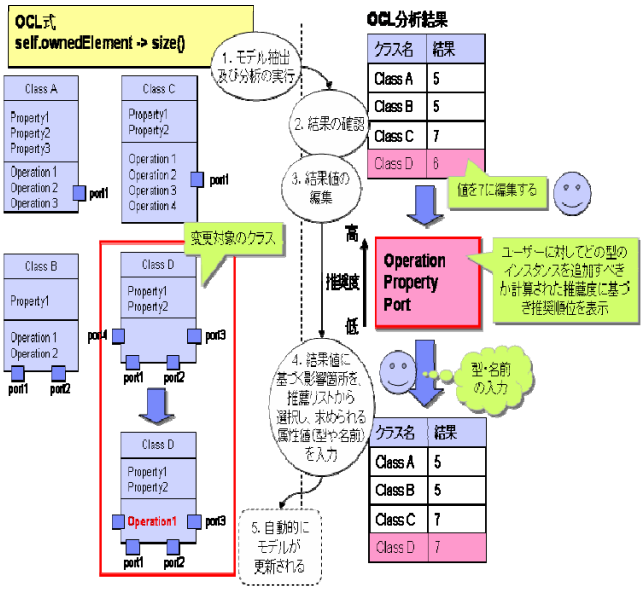

図 3. 適用シナリオの全体像

4. モデル分析式を用いた推薦度計算

OCL で分析した結果を編集するイベントを受けた際，ユーザーに対して，どの型としてインスタンスを追加 (もしくはどの型のインスタンスを削除) すべきかを優先順位付けをして表示する．この優先順位付けの根拠となる値が推薦度である．推薦度は図 4.に示す推薦度計算ルールに従って計算される．

属性名	説明	選択オプション
イベント	この計算ルールが適用されるイベントの種類	▪追加 ▪削除
差異決定方法	推薦度の計算方法	▪平均からの差分 ▪偏差値 ▪推薦度計算関数
ソート形式	推薦度の低い順に表示するか、高い順に表示するかを定義	▪昇順 ▪降順

図 4. 推薦度計算ルール

図 4.の推薦度計算ルールには，イベント，差異決定方法，ソート形式の 3 つがある．推薦度計算ルールには要素追加・削除の際の推薦順位を決定する際，ルールを定義することを可能として XML などを用いた定義ファイルとして記述されることとする．このファイルをもとに,実際のモデル要素の推薦度が計算される．

4.1. 推薦度テーブル

推薦度テーブルは，推薦度を計算する際の基となる Feature の型ごとの数や，計算された推薦度の結果を格納するためのテーブルである．

推薦度テーブルのスキーマを図 5.に示す．オーナー要素名，OCL 式の OwnerExpr で取得される要素の名前が設定される．Feature の型名は，これは UML のメタモデルにおける Feature を表しており，OCL 式の SetExpr にて取得される要素の型名が設定される．

今回，OCL を要素数の分析を対象としているため扱うため，要素数および要素数に対する推薦度をスキーマに含めている．

オーナー要素名	Featureの型名	要素数	推薦度(追加処理)	推薦度(削除処理)

図 5. 推薦度テーブルのスキーマ

推薦度テーブルの表示例を図 6.に示す．最初の行では，オーナー要素が Class A である Feature の型が Property であり，入力モデルに適用した要素数が 2 である．推薦度については，次節で紹介する．

オーナー要素名	Featureの型名	要素数	推薦度(追加処理)	推薦度(削除処理)
Class A	Property	2	54.4	-1
Class A	Operation	1	44.6	-3
Class B	Property	3	63.4	1
Class B	Operation	4	66.1	3

図 6. 推薦度テーブルの表示例

4.2. 推薦度算出手法

推薦度計算ルールにおける差異決定方法に関する推薦度算出手法について説明する．図 3.のモデル情報の例を題材に，平均からの差異を選択した場合の例を示す．

4.2.1. 平均からの差分を差異決定方法に用いた推薦度算出

推薦度計算ルールとして，イベントを”追加”，再決定方法を”平均からの差分”，ソート形式を”降順”としてセットする．図 7 に示すように，差異決定方法は，各クラスが持つ平均からの差異となっている．Property の例では，全体のクラスの Property 保持平均個数が 2 であり，対象となる Class D でも 2 であるので，推薦度は 0 と計算される．Operation に関しては，同様の計算で-3 と計算され，Port に関しては同様に 2.67 となる．また，ソート形式が降順であるため，一番平均からの差分が大きい Operation が優先順位 1 となる．

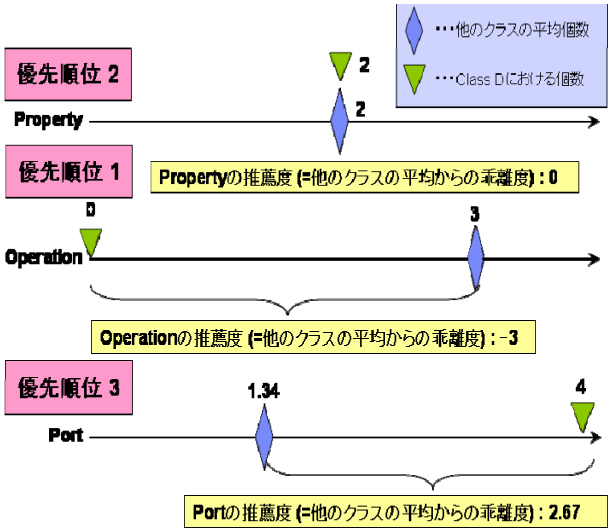

図 7. 平均からの差分を差異決定方法に用いた場合の推薦度算出 (追加処理)

また，イベントを”削除”とし，ソート形式を”昇順”とした場合は，優先順位が逆で計算される．

4.2.2. 偏差値を差異決定方法に用いた推薦度算出

推薦度として偏差値を利用する場合を説明する．推薦度計算ルールとして，イベントを”追加”，再決定方法を”偏差値”，ソート形式を”降順”としてセットする．

平均からの差分が大きい場合でも、分散が大きければ、そもそも各要素ごとにばらつきが発生しやすい Class となる．その場合、追加および削除の動機付けとしては弱いとし、分散による補正が可能な偏差値を用いる。偏差値の計算式を図 8.に示す．

$$10(\text{個々の値} - \text{平均}) / \sqrt{\text{分散}} + 50$$

図 8. 偏差値の計算式

図 9 では，Port と Property を持つ，Class V, X, Y, Z において，OCL 式 self.ownedElement -> size()の分析を適用し，その結果より Class Z に対して要素の追加を行う場合の推薦度算出例を示す．図 8 の計算を適用し，Port および Property の偏差値はそれぞれ，26.9 および 42.3 となる．ソート形式は”降順”で指定されているため， Port の方が優先順位 1 となる．

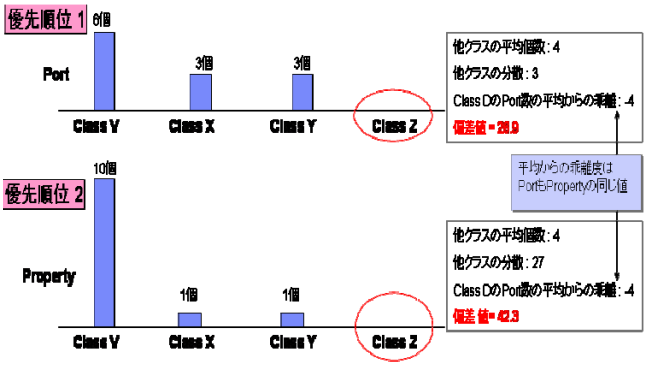

図 9. 偏差値を差異決定方法に用いた推薦度算出

4.2.3. 推薦度計算関数を差異決定方法に用いた場合の推薦度の計算

推薦度計算関数による指標は，OCL 式と偏差値 (or 平均からの差分) などを表す関数を組み合わせた指標を推薦度計算に適用可能である．OCL 式では，図 10 に示す関数 Deviation()に OCL 式を適用した例を示す．

Deviation(self.ownedPort.size() / self.ownedElement.select(oclIsType(Interface)).size())

図 10. 推薦度計算関数の例

Deviation 関数の引数に記述している OCL 式は，あるクラスの持つポートの数とインターフェースの数の比率を計算するものである．この指標を用いることで，あるクラスのポートの数とインターフェースの数の比率が他のクラスと比べてどれくらいかけ離れているかを計算することができる．これにより，例として「インターフェースの数に比べてポートの数が極端に少ない」という好ましくない要素を検知して，ユーザーにポートの追加を促すことが可能となる．

5. 推薦度を取り込んだモデル分析結果とモデル編集のための処理の流れ

5.1. 事前処理

推薦度計算を行うための，事前処理に関する処理全体のイメージを図 11 に示し，主要な手順を以下に示す．

1) ユーザーによってモデル分析結果の値の変更が行われる．

2) モデル更新モジュールによって，処理イベントと選択された要素の通知が推薦度計算モジュールに通知される．

3) 推薦度計算モジュールが推薦度テーブルから選択要素が含まれる行を取得する．

4) 推薦度計算モジュールが取得した行を計算ルールに従い，ソートを行う．

5) 推薦度順位表示モジュールがユーザーに対して追加・削除を推薦する要素を表示する．

6) ユーザーは表示された要素に対して，追加・削除対象の要素を選択する．

7) モデル更新モジュールに追加/削除対象の要素が通知され，モデル要素の追加/削除が行われる．

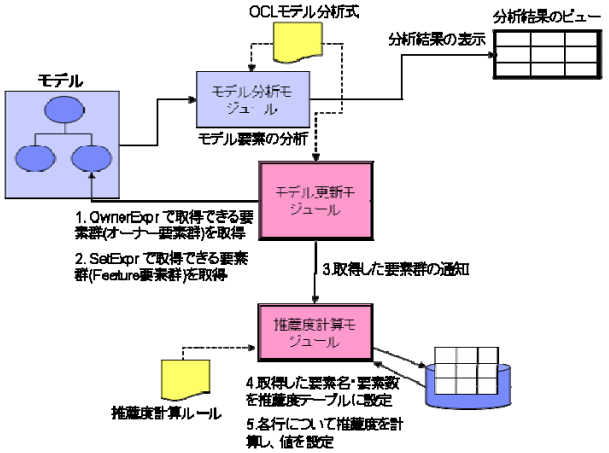

図 11. 事前処理に関する処理の全体

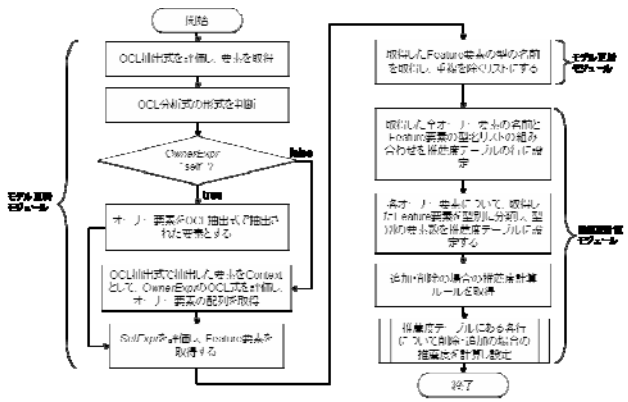

図 12. OCL 式を用いた事前処理のフローチャート

OCL 式解釈を含んだ事前処理のフローチャートを図 12 に示す．

5.2. 追加・削除イベント処理

追加・削除イベント処理に関する処理全体のイメージを図 13 に示し，主要な手順を以下に示す．

1) ユーザーによってモデル分析結果の値の変更が行われる．

2) モデル更新モジュールによって，処理イベントと選択された要素の通知が推薦度計算モジュールに通知される．

3) 推薦度計算モジュールが推薦度テーブルから選択要素が含まれる行を取得する．

4) 推薦度計算モジュールが取得した行を計算ルールに従い，ソートを行う．

5) 推薦度順位表示モジュールがユーザーに対して追加・削除を推薦する要素を表示する．

6) ユーザーは表示された要素に対して，追加・削除対象の要素を選択する．

7) モデル更新モジュールに追加/削除対象の要素が通知され，モデル要素の追加/削除が行われる．

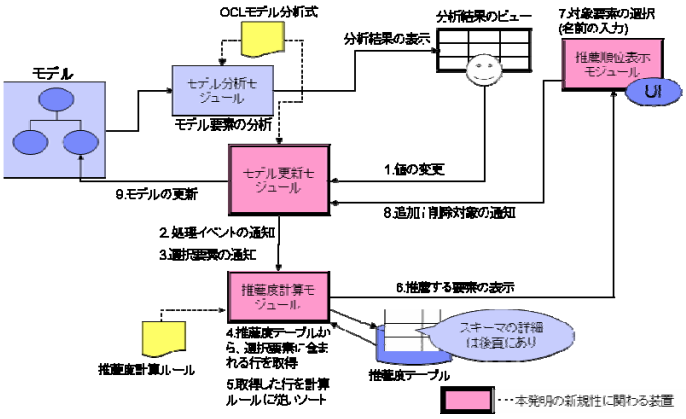

図 13. 追加・削除イベント処理に関する処理の全体

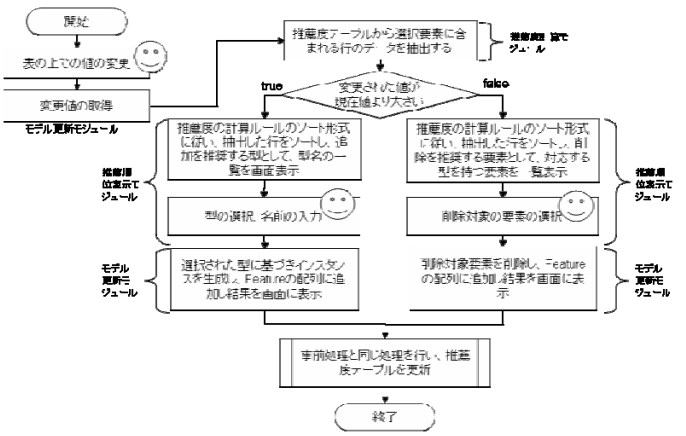

図 14. OCL 式を用いた事前処理のフローチャート

OCL 式解釈を含んだ追加・削除イベント処理に関するフローチャートを図 14 に示す.

6. 適用シミュレーション

実際に適用可能な例をもとにシミュレーションを実施した. まず, 用意する OCL 式を図 15 に示す.

抽出式 : self.oclIsType(Package)
分析式 :self.packagedElement.
select(oclIsType(Class)).ownedElement -> size()

図 15. シミュレーションに用いた OCL 式

適用する UML モデルのイメージを図 16 に示す. Package1 と Package2 にはそれぞれ数の異なる Property と Operation を保有する 4 つのクラスが存在する.

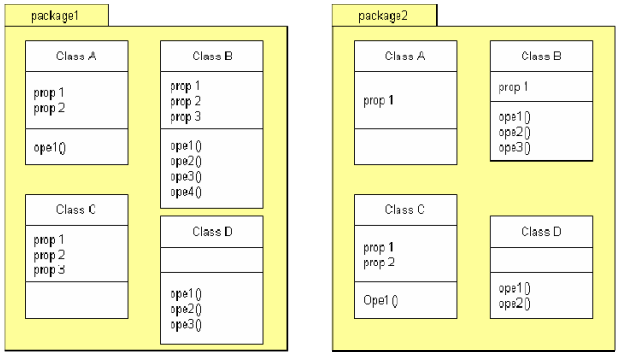

図 16. 適用する UML モデルのイメージ

図 15 の OCL 式を適用すると, Pacakge1 の要素数が 16 で, Package2 の要素数が 10 と表示される. 具体的には要素数は, Property と Operation の総数となる.

また, 推薦度計算ルールとして, イベント”追加”に対し, 差異決定方法とソート形式をそれぞれ, ”偏差値”, ”降順と指定する. また, イベント”削除”に対して, 同様に”偏差値”, “昇順”と指定する. この推薦度計算ルールを適用して計算された推薦度テーブルを図 17 に示す. 図 17 では, 要約して Package1 のみに着目することとする.

オーナー要素名	Featureの型名	要素数	推薦度(追加処理)	推薦度(削除処理)
package1.Class A	Property	2	54.472136 (4)	54.472136
package1.Class A	Operation	1	44.611841 (3)	44.611841
package1.Class B	Property	3	63.416408 (6)	63.413408
package1.Class B	Operation	4	66.104477 (7)	66.104477
package1.Class C	Property	3	63.416408 (6)	63.413408
package1.Class C	Operation	0	37.427629 (2)	37.427629
package1.Class D	Property	0	30.583592 (1)	30.583592
package1.Class D	Operation	3	58.98023 (5)	58.98026

図 17. 適用例における推薦度テーブル

なお, この推薦度テーブルの図では, 推薦度計算ルールに基づき偏差値の低い順に優先順位を付けた例を, 追加処理に対して示している.

この状態で, 設計者側の利用イメージを図 18 に示す. まず, Package1 に対する分析結果である 16 に対して設計者が 17 に変更したとする. その際に, 推薦度表示モジュールにより, 図 16 の推薦度テーブルをもとに計算した, 追加するモデル要素の Type を推薦度順に表示する. この表示により設計者は, 編集の継続として, 追加対象のモデル要素の Type を選択し, そのモデル要素の名前を入力する. これにより, 自動的にモデル構造の制約に沿ったモデル要素の追加が自動的に行われることが可能となる.

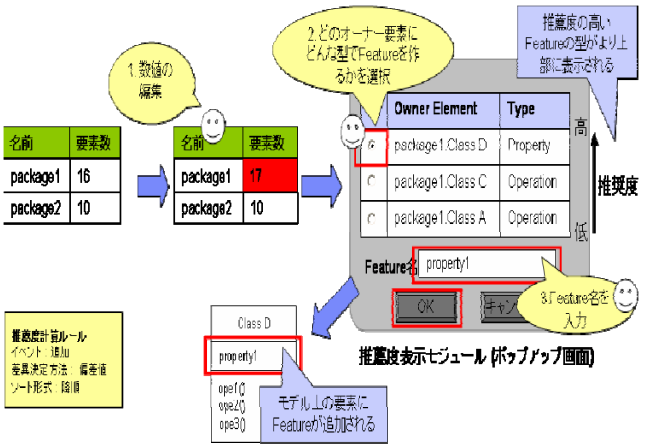

図 18. 推薦度を用いた分析結果の編集によるモデル変更イメージ

7. まとめ

従来技術では, モデル内に多量の要素が存在する中でモデル分析を行い, その結果基づき要素を修正する必要がある場合, 該当する要素を手作業で検索してから修正を行う必要があった. ゆえに, 作業の容易性が低くなり, 作業効率が低下する恐れがあった. これに対して, 本研究における提案による分析結果を直接編集できることにより, モデルの編集作業の効率性を高めることが可能となる.

今後は本研究で検討を行った技術を応用し, モデル駆動型開発におけるプロセスに適用することでその実用性を検証していきたい.

文　献

[1] OCL:
http://www.omg.org/technology/documents/modeling_spec_catalog.htm#OCL
[2] Eclipse BIRT :
http://www.eclipse.org/birt/phoenix/intro/index.php