社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SS2005-48 (2005-10)

UMLモデルに対するXPathとXMI-differenceを用いた不整合検出と解消

佐々木　亨†　　岡野　浩三†　　楠本　真二†

† 大阪大学大学院情報科学研究科
〒 560-8531 大阪府豊中市待兼山 1-3
E-mail: †{toru-ssk,okano,kusumoto}@ist.osaka-u.ac.jp

あらまし　本稿では，UML モデルの要素間の不整合を検出しそれらを解消するツールを提案する．具体的には，クラス図とシーケンス図の要素が満たすべき整合性ルールと，そのルールに基づき不整合が検出された場合に実行すべき解消動作を，XPath と XMI-difference を用いて記述する手法を提案する．また，その記述に基づいて，不整合の検出，解消を自動的に行なうツールについて述べる．

キーワード　UML，整合性，不整合解消，XMI-difference，XPath

Detection and Repair of Inconsistency in UML Models using XPath and XMI-difference

Toru SASAKI†, Kozo OKANO†, and Shinji KUSUMOTO†

† Graduate School of Information Science and Technology, Osaka University
Machikane-yama 1-3, Toyonaka City, Osaka, 560-8531 Japan
E-mail: †{toru-ssk,okano,kusumoto}@ist.osaka-u.ac.jp

Abstract　In this paper, we propose a method for detection and repair of inconsistency in UML models. The method utilizes conditions that elements in class diagrams and sequence diagrams should meet and actions which enable when inconsistency elements are detected. These rules and actions are described in a language based on XPath and XMI-difference. Finally, we give a tool detecting inconsistency elements and automatically executing the repair actions based on our proposed approach.

Key words　UML, Consisntecy, Repair Action, XMI-difference, XPath

1. はじめに

近年，ソフトウェア開発において，UML(Unified Modeling Language) [1] が幅広く利用されている．UML は対象システムを複数の作業者が様々な視点から捉え，複数のダイアグラムとして記述するため，また，繰り返しの中で詳細化，修正されるため，それらのダイアグラム間で不整合が生じる可能性がある．また，ソフトウェアの巨大化に伴い，ダイアグラムが大規模化，複雑化しているため，その内部においても不整合が生じることがある．こうした問題を解決するために，これまでに数多くの UML の整合性管理に関する研究が行なわれてきている．これらは以下の 2 種類に大別できる．

（1） 開発プロセスの各段階に存在するモデル間で不整合が生じないように，モデル間の refinement 制約を記述し，それをチェックするというもの [2] [3]

（2） 同一モデル内の，ダイアグラム間の不整合をチェックするためにダイアグラムの要素間の制約を定義するもの [4] [5]

我々は，(2) のダイアグラム間の不整合チェックに着目した．既存の研究では，要素間の制約を定義するだけ，もしくは，制約を満たしたかどうかをフィードバックするのみで，不整合要素の検出，解消までを十分にサポートするものはほとんどなかった．しかし，検出されたすべての不整合を人手で解消することは，膨大な手間がかかり効率が悪い．そこで，不整合の解消を自動的に行なう，または，解消を手助けする情報を開発者に与えるといった支援が必要となる．

不整合の解消支援を行なう既存のアプローチとしては，佐藤らによるもの [6] や Nentwich らによるもの [7] がある．[6] は，クラス図の各コンポーネントを論理プログラミングのルールとして表し，その上で矛盾が導かれるかどうかをチェックし，矛盾があれば，その極小矛盾集合を表示するというものであるが，クラス図のみしか扱っていない．また，[7] は，xlinkit というツールにより，ユーザが与えた制約式に違反するような UML

モデルの不整合要素を検出し，その要素情報とマッピングルールを用いて解消動作を導出，実行するものである．

しかし，不整合を検出するために用いられる，各要素が満たすべき整合性ルールは，ユーザによってその都度与えなければならないので，解消動作実行後に新たに生じ得る不整合を予測して，ユーザに提示したり，複数ステップに及ぶ解消動作を考えることができない．また，多くの不整合の原因となっているような要素を優先的に解消するといった，解消動作の優先度を考えていない．そこで我々は，

- UML モデルの要素が満たすべき整合性ルール
- 不整合が検出された場合に実行する解消動作

の記述法を提案し，既存研究 [5] で知られている整合性ルールと，それぞれのルールにおいて不整合要素が検出された場合に実行すべき解消動作集合を記述することとした．整合性ルールとそれを違反した際に実行すべき不整合解消動作をあらかじめ与えておくことによって，(1) ユーザがチェックの度にルールを入力する必要がない．(2) その開発現場にマッチした整合性ルール，解消動作を与えることができる．(3) 解消動作によって作りこまれる可能性のある，新たな不整合を検出することができ，これまで実現されていなかった，複数の解消動作の組み合わせによる不整合解消や，不整合を作り込む可能性のある解消動作であることをユーザに提示するなど，効果的な不整合解消の支援が実現できるようになる．

本稿では，整合性ルール，解消動作の記述法を提案し，記述した整合性ルールに基づき，UML モデルの不整合要素を検出し，対応する解消動作に従って自動的に UML モデルを編集するツールの説明を行なう．以降，2. で提案する手法とツールおよび対象となるモデルについて述べ，3. では整合性ルール記述，4. において解消動作記述を提案する．5. で提案するツールを例題に適用し，最後に 6. でまとめと今後の課題について述べる．

2. 提案する手法の概要

2.1 手法とツールの概要

提案する手法の概要を図 1 に示す．ツールにはあらかじめ，満たすべき全ての整合性ルールとその解消動作のセットを，以降で提案する表記法を用いて，書換え規則の形で記述し与えておく．図 1 を用いて動作を説明する．なお，文章中の番号は，図中の番号と対応している．

まず，(1) ユーザは，モデリングツール等で UML 文書を作成する．(2) ユーザは，UML 文書をツールへの入力として与える．(3) ツールは，与えられた UML 文書と，あらかじめ登録されている全ての整合性ルールを参照しながら，整合性をチェックし，不整合が検出された場合には，書換え規則として記述されている解消動作から，その場に応じた解消動作集合を導出する．この動作を繰り返すことで，複数ステップに及ぶ解消動作も同時に導出することができる．(4) ツールは，ユーザに優先度を示して解消動作を提示する．(5) ユーザは，提示された解消動作の中から，適した解消動作を選択する．(6) ツールは，ユーザが選択した解消動作を実行し，自動的に，または，必要に応じてはユーザと対話的に UML 文書を編集する．

もし，適した解消動作がない場合には，ユーザが新たに解消動作を追加することもできる．また，ユーザが利用目的に合わせて，UML モデルに適用してチェックしたい整合性ルールの選択をすること，外部から新たに整合性ルール・解消動作集合を入力することなども想定している．

2.2 対象とするモデル

本研究で扱うダイアグラムは一般のモデリングツールで作成される UML ダイアグラムである．ダイアグラムは，どういう形でどこに配置するかという描画，レイアウト情報に関する表記モデルと，それ以外の論理モデルから成るが，我々はこの論理モデルを扱う．

また，論理モデルは，UML メタモデルとして定義され，モデリングツールでは，ツール固有のバイナリ形式か，XMI [8] 準拠の XML ファイルとして出力されるが，XMI バージョン 1.1 に準拠した XML ファイルを対象とする．なお，我々はモデリングツールとして，Enterprise Architect(EA) [9] を利用し，これを用いて XMI 準拠の XML ファイルを作成した．

3. 整合性ルール記述

本章では，整合性ルール記述について説明する．これまで，要素間で満たすべき整合性ルールを記述するには，OCL(Object Constraint Language) [10] を利用するのが一般的であった．OCL は UML を策定している OMG(Object Management Group) の標準でもあり，整合性ルールを高階論理で記述できるため，整合性ルールの記述のみを考えるのであれば，有用と考えられる．しかし，本研究では，不整合要素の解消動作もターゲットとしているため，解消動作の表記には適さない OCL は利用しない．そこで我々は，整合性のルール記述のために [7] で定義されているルール文法を用いた．

整合性ルールは [5] で定義されているものを用いた．この中では，クラス図，シーケンス図，ステートチャート図に関する整合性ルールを OCL で定義しており，全部で 120 個の整合性ルールから成る．我々は今回，クラス図とシーケンス図に関する 72 個の整合性ルールを 11 のグループに分類し，各グループについて代表的な 1 個または 2 個を扱うこととした．今回扱わない整合性ルールに関しても，各グループにおいては類似した形で記述ができると考えられる．[5] でも述べられているが，これらで全ての整合性ルールを扱っているとは証明できないため，2. で述べたように，ユーザが新たにルールを追加できることが必要となる．

3.1 整合性ルール記述文法

整合性ルールを記述するためのルール文法を表 1 に示す．なお，“XPath” と記述してある部分は，XPath [11] の表記法を用いる部分である．XPath については，次節で説明する．

表 1 の整合性ルール記述文法のように，全整合性ルール記述は，“∀” 記号から始めることができると考えている．実際，今回扱う整合性ルールについて確認をしたが，そのすべての整合性ルールが “∀” 記号から始まることを確認できた．すなわち，各構成要素について，ある規則が満たされるべきである，といった形でのみ制約式が記述されるという制限を置いている．

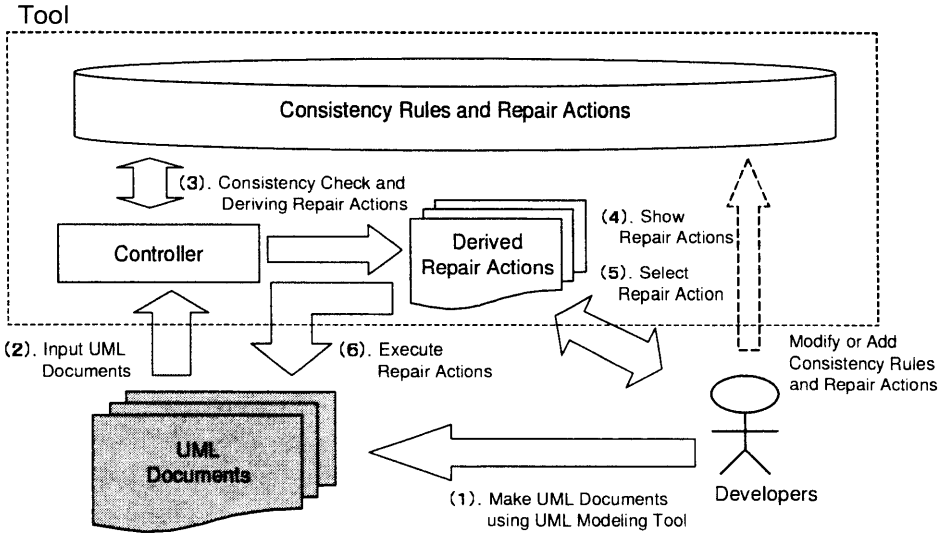

図 1　提案する手法の概要図

表 1　整合性ルール記述文法

Rule	::=	∀ var ∈ XPath (Formula)
Fomula	::=	∀ var ∈ XPath (Formula) |
		∃ var ∈ XPath (Formula) |
		Formula and Formula |
		Formula or Formula |
		Formula implies Formula |
		not Formula |
		XPath = XPath

3.2 XPath

XPath は，XML 文書の中の特定の位置を指し示すための言語の仕様であり，XML ドキュメントの階層構造をナビゲートする際に用いられる．我々は XMI に準拠した XML ファイルとして表されたモデルのロケーションパスにこれを用いる．基本的な記述方法は UNIX のファイルシステムに似ていて，文書木構造の頂点となるルートノードを「/」で表し，以下「/」で区切って要素をたどっていく．例えば，a 要素の中の b という値を参照するには「/a/b」と記述する．要素 element の属性 attribute の値が「value」であるものを選択するときには，「//element[@attribute="value"]」のように記述する．ノードのデータ型やノードの種類，名前空間 (XML namespace) の扱いなどについても規定があり，これらを使用して条件式や演算などを含んだ複雑な位置指定を行なうこともできる．また，一般的な表現をより簡潔に表現することができる構文的省略も多く存在する．これにより，個々の要素をすべて表現しなくても，省略した形式で一連の要素や特定の条件に一致する要素を表すことができる．省略文法を使ったロケーションパスの例を表 2 に挙げる．

3.3 記 述 例

以上で説明した整合性ルール記述文法を用いて，図 2 で示されるように，「シーケンス図中のどのメッセージも，クラス図中に対応するオペレーションを持つ」ことを表す整合性ルールを、

表 2　XPath の省略文法を使った記述例

表記	意味
text()	文脈ノードの子テキストノードすべてを選択する
@name	文脈ノードの name 属性を選択する
@*	文脈ノードの属性すべてを選択する
para[n]	文脈ノードの n 番目の子 para を選択する
para[last()]]	文脈ノードの最後の子 para を選択する
/para	文脈ノードの孫 para すべてを選択する
.	文脈ノードを選択する
.//para	文脈ノードの子孫 para 要素を選択する
..	文脈ノードの親を選択する

XML で記述した例を図 3 に挙げる (便宜上行番号を振っているが，実際には行番号はない)．

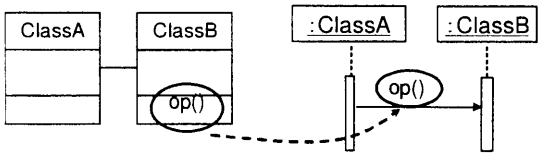

図 2　整合性ルール例

```
1: <rule>
2:   <forall var="x" in="/UML:Message">
3:     <exists var="y" in="/UML:Operation">
4:       <equal op1="$x[@name]"
5:              op2="$y[@name]" />
6:     </exists>
7:   </forall>
8: </rule>
```

図 3　整合性ルール記述例

記述したルールについて説明する．1 行目と 8 行目は，ルールの始まりと終わりを示すためのタグであり，これらの内側に

整合性ルールを記述する．2行目から7行目までの内容を，論理式で表すと次のようになる．

∀x∈“message”(∃y∈“operation”(“x/name”=“y/name”)))

このとき，“operation”(“x/name”=“y/name”) を満たさないような message 中の要素 x が，この整合性ルールに違犯する不整合要素として検出される．不整合要素に関する情報は，その要素の種類 (UML メタモデルでの分類)，要素名，そしてモデリングツールによって要素に対して一意に振られる ID を使って保持し，これらの情報が，次章で述べる解消動作記述に使われる．

4. 解消動作記述

本章では，解消動作記述について説明する．3. で扱ったクラス図とシーケンス図に関する整合性ルールそれぞれに対して，違犯した場合に実行すべき，考えられる解消動作を記述する．

我々は，解消動作記述に XMI-difference [8] を用いることとした．この XMI-difference は，本来，変更のあったモデル情報を転送する場合に，その差分だけを転送するために使われるもので，UML モデル情報の交換標準である XMI 仕様で定義されている．XMI-difference は，add，delete，replace という3つの動作を表す要素から成る．不整合の解消もモデル情報の変更であることを考えれば，この add，delete，replace を使えば，解消動作を記述できることは明らかである．もちろん，replace は add, delete を組み合わせて用いることで代用できるが，記述の読みやすさを考慮して，そのまま用いることとした．これら3つの要素中には属性として，参照したい位置を表す “ref” が用意されている．この位置の指定には，他の XML 文書へリンクする手段を提供する XLink や，XLink とともに使って特定の位置を指し示すために使われる XPointer を利用することができる．つまり，これらをまとめた仕様である XPath を使っていることになる．例えば，ある要素を追加する場合は，次のように記述すればよい．

```
<XMI.add href='XPath'>
    <追加する要素>
</XMI.add>
```

4.1 解消動作記述文法

解消動作を記述するための文法を表3に示す．上述したように，解消動作は Add, Delete, Replace から成り，これらで1つの解消動作を記述する．また，ある不整合に対する解消動作は複数存在するので，それらを “<Action>” “</Action>” タグの内側に記述する．Add, Delete, Replace 要素中では，XMI-difference では定義されていないが，参照したい位置を表す属性の他に，変数を扱えるようにした．変数は属性 “var” を使って指定することができ，“var1”,“var2” のように，接頭辞として “var” を持つような属性として，複数定義することもできる．要素 Add, Replace については，追加もしくは置換する要素 “Element” をその子要素として持つこととする．この要素 “Element” は，UML モデル要素を XMI に準拠した形式で記述するが，この “UML models” 中，解消動作を記述する時点では，まだ値の定まらないような属性がある．例えば，「シーケンス図中のメッセージ名をクラス図中で定義されているオペレーション名に変更する」というような解消動作の場合，「クラス図中で定義されているオペレーション名」というのは，実際のモデル情報からしか得ることはできない．このような場合には，UML モデル中で，XPath を利用して，特定の属性の位置や，特定の条件に一致する要素の属性等を指定して記述できる．また，ユーザと対話的にやりとりをすることでしか定めまらない値 (例えば，新しいオペレーションを追加する際のオペレーション名) も考えられるので，そのことを表す変数を用意した．これらについては，次節で具体例を使って説明する．

表3 解消動作記述文法

Action	::=	(Add | Delete | Replace)*
Add	::=	Variable* Reference Element
Delete	::=	Variable* Reference
Replace	::=	Variable* Reference Element
Variable	::=	var = String in = XPath
Reference	::=	ref = XPath
Element	::=	(UMLModels | XPath)*

4.2 記 述 例

ここでは，上記の表3の文法を使った，解消動作記述の例を示す．先ほどの図3の例を考える．解消動作として考えられるのは，以下の3つの動作である．

（1） メッセージの受信オブジェクトが属するクラスに，メッセージのオペレーションを追加する (図4)

（2） シーケンス図中のメッセージを消す (図6)

（3） メッセージのオペレーション名を受信オブジェクトが属するクラスで定義されているオペレーションに置き換える (図8)

これらの解消動作を表した図と，それを記述文法にならって記述したものを以下の図4〜9に示す．なお，各解消動作記述は，本来は，“<Action>” “</Action>” タグにかこまれた1つの XML ファイルに書かれているが，今回はその中の3つの要素を取りだしたものを別々に載せている．

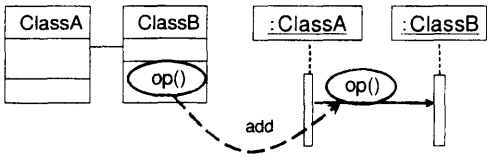

図4 解消動作例 (1)

これらの解消動作について説明する．まず，“$position” の表記は，XML ファイル中での不整合要素の位置を指す．ほとんどの解消動作はこの位置を基点として XPath を利用して，位置指定，属性値の取得 (例えば図5の2行目の “[@receiver]” で属性 receiver の値を取得している) をする．

次に，図5の7行目，8行目にある “%” 記号は，解消動作記述段階では記述できない不明な値を表しており，解消動作を実行する際に，ユーザとの対話によって定めるものとする．ま

```
1: <XMI.add
2:  var1="x" in1="$position[@receiver]"
3:  var2="y" in2="[@xmi.id='$x']/…/TaggedValue"
4:  var3="z" in3="y[@tag="classname"][@value]"
5:  ref="UML:Class[@name='$z']
              /Classifier.feature" >
6:    <UML:Operatin name="$position[@name]"
7:                  visibility="%" ownerScope="%"
8:                  isQuery="%" concurrency="%" />
9: </XMI.add>
```

図 5　解消動作記述例 (1)

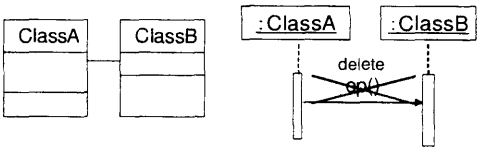

図 6　解消動作例 (2)

```
1: <XMI.delete href="$position" />
```

図 7　解消動作記述例 (2)

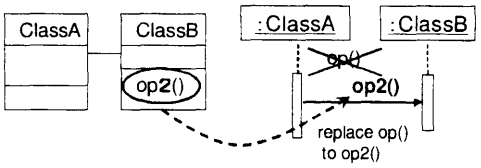

図 8　解消動作例 (3)

```
1: <XMI.replace
2:  var1="x" in1="$position/…/TaggedValue"
3:  var2="y" in2="$x[@tag='classname'][@value]"
4:  var3="z" in3="UML:Class[@name='$z']/
                    …/Operation"
5:  ref="$position" >
6:     <UML:Message name="$z[@name]"
7:                  xmi.id="unchage"
8:                  visibility="unchange"
9:                  sender="unchange"
10:                 receiver="unchange" />
11: </XMI.replace>
```

図 9　解消動作記述例 (3)

た，図 9 の 7〜9 行目にある “unchange” は，置換する前の値と変化しないことを表している．

5. 例題への適用

これまでに述べた手法を簡単な例題に適用した．なお，今回我々は描画・レイアウト情報に関する表記モデルは対象としておらず，本質的なデータに関する論理モデルのみを扱っている．そのため，解消動作によって XMI ファイル中の論理モデルに関する部分にのみ変更を加えるため，できた XMI ファイルを UML モデリングツールで読み込んだとしても，表記モデルが変更されていないため，クラスの追加・削除等，画面上に変化は見られない (現在存在するクラスへの属性値，オペレーション等の変更・追加・削除は行なわれる)．しかし，本稿では，表記モデルも変更されていることとして説明を行なう．

5.1 例題概要

適用する例題の UML クラス図とシーケンス図を図 10 に示す．本例題は，Automatic Teller Machine(ATM) の例である．

具体的には，このシーケンス図は，ユーザが現金を引き出すときの ATM の動作を表している．この例題には，シーケンス図中の B で示されるメッセージのオペレーションが，クラス図中で定義されていない (本来はクラス図中の A あたりで定義さるべき)．つまり，整合性ルールの 1 つ,「シーケンス図中のどのメッセージも，クラス図中に対応するオペレーションを持つ」に関する不整合が埋めこまれている．

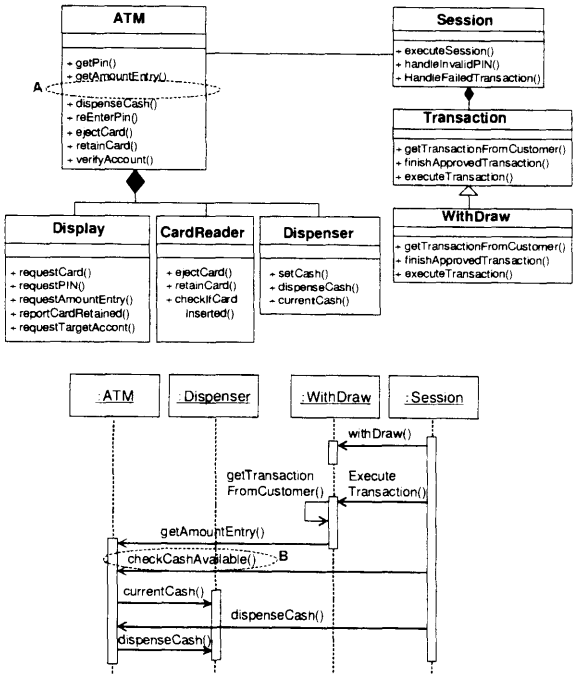

図 10　例題のクラス図とシーケンス図

5.2 不整合検出

上記の例題に対して，あらかじめクラス図とシーケンス図に関する 12 のルールとその解消動作を記述してある，提案するツールによって，不整合検出と解消動作導出を行なった．その結果，次のような不整合要素が検出された．

```
===== There are Inconsistency Elements =====
Element No[1]
 kind   : UML:Message
 name   : checkCashAvailable
 id     : EAID_E16769F5_38EA_4f4d_9A21_C9EF4776F2F3
```

これは，UML の Message 中の，名前が”checkCashAvailable” であるメッセージに関する不整合があることをユーザに提示している．なお，この “id” は，モデリングツールによって要素毎に振られる一意な ID である．

5.3 解消動作実行

次に，上記の不整合要素に対して導出された解消動作3つ(4.2で挙げた解消動作3つ)の中から，「メッセージの受信オブジェクトが属するクラスに，メッセージのオペレーションを追加する」という解消動作を選択し，実行した．その結果，図11に示す通り，クラスATMのオペレーションに，新たに“checkCashAvailable()”が追加された(図中のA')．

その後，このモデルの整合性をツールでチェックしたところ，以下の通り，不整合要素は検出されなかった．つまり，正しく解消できたことが確認できた．

```
===== There are no Inconsistency Elements =====
```

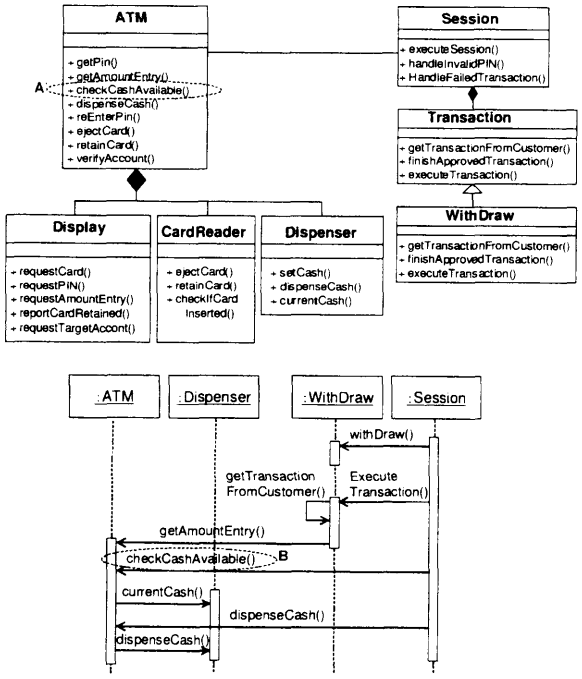

図11　不整合解消後のクラス図とシーケンス図

今回，導出された解消動作3つのうち，「メッセージの受信オブジェクトが属するクラスに，メッセージのオペレーションを追加する」という動作を選択したが，残りの2つを選択した場合にも，正しく動作し，不整合が解消されることが確認できた．

6. まとめ

本稿では，UMLモデルの要素間の不整合を検出し，それらを解消するツールに関して述べた．整合性ルールと，不整合が検出された場合に実行する解消動作を，XPathとXMI-differenceを用いた記述法を用いて記述し，その記述に基づいて，不整合の検出，解消を自動的に行なうツールを示した．

今後の課題として，以下のことが挙げられる．

(1) 解消動作の優先度付け手法の考案と実装

多くの不整合の原因となっている要素に関する解消動作を優先的に実行することが効率的であるが，本ツールではまだ実現できていない．そのため今後，要素間の依存グラフ等を用いて，優先的に解消すべき要素を見つけ，それをユーザに提示する機能を追加したい．

(2) 複数ステップの解消動作の実現

ある不整合の解消動作を実行した後に，新たに別の不整合が発生する可能性がある．そのため，それらを予測してその不整合に適用する解消動作を求める，ということを繰り返し行なって，複数ステップにおよぶ解消動作をユーザに提示，実行する機能が望まれる．

(3) ユーザに対するグラフィカルな提示

解消動作をユーザに提示する際に，4.で挙げたような記述ではなく，一目でわかるようなグラフィカルな提示方法が必要である．例えば，[4]らによるGraphical Consistency Conditions(GCC)のような提示法が考えられる．

(4) 整合性ルールの追加・改良

我々が記述した整合性ルールで全てとは言えないので，新しく見つかる度に，追加していきたい．

(5) 表記モデルへの対応

現在扱っているのは，論理モデルだけである．つまり，描画情報などの表記モデルを扱えていない．このため，解消動作を実行したあとのXMIファイルをモデリングツールで読みこんでも，表示が繁栄されないため，その結果を一目で確認することはできない．このため，今後，モデリングツールでの表記モデルも扱うことが重要と考える．

(6) 大規模な例題への適用実験

大規模で，実際の開発現場で作成されたUMLモデルへ適用して，有効性を確認することが必要である．

文　献

[1] OMG. UML 2.0 Infrastructure Specification, 2003. http://www.omg.org/docs/ptc/03-09-15.pdf

[2] W. Shen, Y. Lu, and W. Liong Low, “Extending the UML Metamodel to Support Software Refinement,” Workshop Consistency Problems in UML, pp.35-42, 2003.

[3] B. Hnatkowska, L. Kuzuniarz, Z. Huzar, and L. Tuzinkiewicz, “Refinement relationship between collaborations,” Workshop Consistency Problems in UML, pp.51-57, 2003.

[4] J. Hausmann, R. Heckel, and S. Sauer, “Extended Model Relations with Graphical Consistency Conditions,” Workshop Consistency Problems in UML, pp.61-74, 2002.

[5] L. C. Briand, Y. Labiche, and L. O'Sullivan, “Impact Analysis and Change Management of UML Models,” Proceedings of the International Conference on Software Maintenance, pp.256-265, 2003.

[6] 佐藤 健，兼岩 憲，“UMLのクラス図における論理プログラミングを用いた無矛盾検査について”，ソフトウェア科学会第22回大会論文集，2005.

[7] C. Nentwich, W. Emmerich, and A. Finkelstein, “Consistency Management with Repair Actions,” Proceedings of the 25th International Conference on Software Engineering, pp.455-464, 2003.

[8] XML Metadata Interchange, http://www.omg.org/technology/documents/formal/xmi.htm

[9] Enterprise Architect, http://www.sparxsystems.jp/

[10] OMG. UML 2.0 OCL Specification, 2003. http://www.omg.org/docs/ptc/03-10-14.pdf

[11] XML Path Language(XPath) 2.0, http://www.w3.org/TR/2005/WD-xpath20-20050404/