制約指向に基づいたUMLモデルの不整合検出・解消手法の提案

佐々木　亨[†*]　　岡野　浩三[†]　　楠本　真二[†]

Consistency Management of UML Model Based on Constraint Rules and Repair Actions

Toru SASAKI[†*], Kozo OKANO[†], and Shinji KUSUMOTO[†]

あらまし　UML（Unified Modeling Language）では複数種類のダイアグラムを用いて，システムを複数の異なる視点から表現できるが，設計者が記述した異なる種類のダイアグラム間で不整合が生じることがある．本研究では，記述されたUMLモデルの各構成要素間が満たすべき静的な整合性をチェックし，不整合の検出とその不整合の解消動作を導出，実行するための手法を提案する．本提案手法では制約指向のアプローチに基づき，XMI（XML Metadata Interchange）形式で保存されたUMLダイアグラムの構成要素が満たすべき制約式と，それらの違反時に実行すべき解消動作を，あらかじめ規則として記述し与えておく．これらの記述方法として，XPathとXMI.difference表記を用いた記述文法を与える．これらの記述に基づき，整合性検査，不整合要素の検出と解消動作の導出，実行を繰り返すことで，不整合解消動作の候補をユーザに提示することができる．提案手法を実装したツールをDrinkMakerの例題に適用した結果，本手法がUMLダイアグラムの修正支援として有効であることが確認できた．

キーワード　UML，制約指向，静的整合性，XPath，XMI.difference

1. まえがき

UML（Unified Modeling Language）は対象システムを複数の作業者が様々な視点からとらえ，複数のダイアグラムとして記述できるという特徴があるが，各ダイアグラムが完全に独立しているわけではなく，設計者が記述したダイアグラム間で不整合が生じる可能性がある．また，開発工程における繰返しの中でダイアグラムが詳細化，修正されるため，同一のダイアグラムにおいても，各工程間で不整合が生じる可能性がある．こうした問題を解決するため，UMLの整合性管理に関する研究が大別して二つのアプローチで多く行われてきている[1]．(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制約を定義し，それに従い開発を進めることを促し，制約を満たしているかどうかを調べる[2], [3]．(2) 同一モデル内の，ダイアグラム間（ダイアグラム内の構成要素間も含む）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し，それを調べる[4]〜[11]．

本研究では，上記(2)のダイアグラム間の不整合検査に着目した．既存の研究では，要素間の制約を定義するだけ，若しくは，制約を満たすかどうかをフィードバックするのみで，不整合要素の検出から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には膨大な手間がかかり効率が悪い．また，既存のモデリングツールにおいて，これらの不整合を起こさない機能を有しているものが存在するが，そのためには，通常のモデリング作業に加えて，ダイアグラムの構成要素間に依存関係を定義するなどの処理が必要となり，手間がかかる．こうした理由から，不整合検出だけではなく，自動的な不整合解消，または解消の支援情報の開発者への提供といった機能が必要となる．

不整合の解消支援を行う既存のアプローチとしては，論理プログラミングを使ったもの[8]や，xlinkit[9], [10]と呼ばれるツールを用いたもの[11]がある．

† 大阪大学大学院情報科学研究科コンピュータサイエンス専攻，豊中市
Graduate School of Information Science and Technology, Osaka University, Toyonaka-shi, 560–8531 Japan
* 現在，日本オラクル株式会社

文献 [8] では，クラス図の各コンポーネントを論理プログラミングの規則として表し，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，クラス図のみしか対象にすることができない．また，文献 [11] は，xlinkit により，ユーザが与えた制約式に違反する 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規則を用いて解消動作を導出し，選択した解消動作を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る，各要素が満たすべき整合性規則は，ユーザによって整合性をチェックする都度入力しなければならないため，解消動作を実行してダイアグラムの構成要素に修正を加えることで，他の構成要素との間で新たに発生する可能性のある不整合に関しては，検出することができない．そのため，複数ステップに及ぶ解消動作を扱うことができない．

本研究では，(A)「UML モデルの要素が満たすべき整合性規則（制約式）」と (B)「不整合が検出された場合に実行する解消動作」の記述方法を提案する．これら二つからなる規則をあらかじめ用意することで，整合性規則に違反した際に実行すべき不整合解消動作を容易に求めることができる．そのため，(1) ユーザがチェックのたびに規則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規則，解消動作を与えることができる．(3) 解消動作によって生じる可能性のある，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合せによる不整合解消や，不整合を生じる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．

以降，**2.** で手法の概要，対象とするモデル，記述手法とツールの提案を行う．**3.** で，例題適用実験の結果と考察を述べる．最後に **4.** でまとめる．

2. 提 案 手 法

2.1 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，満たすべきすべての整合性規則，それを違反した場合に実行すべき解消動作の集合を，以降で提案する表記法を用いて，各整合規則ごと次のように与える．

整合性規則 ⇒ 解消動作 A｜解消動作 B｜解消動作 C

なお，複数の解消動作 A，B，C は整合性規則に違反した場合の解消動作候補が，複数あることを意味する．

図 1 を用いて提案手法の概要を説明する．なお，文中の番号は，図中の番号と対応する．

（1） ユーザは，モデリングツールで UML モデル

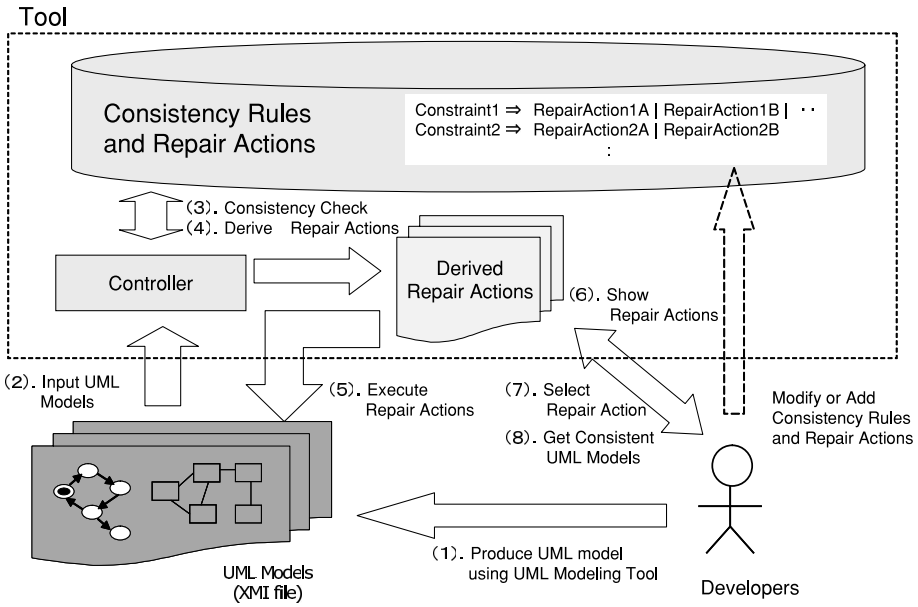

図 1　提案する手法の概要図
Fig. 1　Outline of the proposed method.

を作成する．

（2） ユーザは，XMI 形式で保存した UML モデルをツールへの入力として与える．

（3） ツールは，与えられた XMI ファイルを DOM トリーに展開し，それらノードに対して，あらかじめ登録されているすべての整合性規則をチェックする．

（4） 不整合が検出された場合には，対応する解消動作から，その場に応じた解消動作集合を導出する．

（5） 解消動作を内部で実行する．

（6） ツールは，ユーザに解消動作を提示する．

（7） 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．

（8） 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，図 1 の破線の矢印で示されるように，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，チェックしたい整合性規則の選択し，その規則についてのみ UML モデルに適用することができる．

なお，上記の（2）から（5）を繰り返し行うことで，ある要素に対する解消動作の実行によって新たに他の要素において発生する不整合を検出することができる．更に，その不整合を解消する動作を導出し，この繰り返しによって，複数の解消動作の組合せによる不整合の解消を実現することが可能となる．

2.2 対象とするモデル

本研究で対象とする UML モデルは一般の UML モデリングツールで作成され，XMI 準拠の XML 形式で表された UML モデルである．XMI のバージョンは 1.1 とし，モデリングツールには Enterprise Architect（EA）を用いた．

2.3 整合性規則記述

2.3.1 記 述 文 法

本節では，整合性規則記述について説明する．要素間で満たすべき整合性規則の記述には，OCL（Object Constraint Language）[14] を利用するのが一般的である．OCL は UML を策定している OMG（Object Management Group）の標準であり，制約式を高階論理で記述できる．しかし，OCL により，UML モデルの構成要素が満たすべき整合性規則をチェックする場合，真偽値以外のフィードバックを返すことが困難であるとされている [9]．

そこで，文献 [9] をもとに規則記述文法を定めた．

表 1 整合性規則記述文法
Table 1 Syntax of consistency rules.

Rule	::=	∀ var ∈ XPath (Formula)
Fomula	::=	∀ var ∈ XPath (Formula) |
		∃ var ∈ XPath (Formula) |
		Formula and Formula |
		Formula or Formula |
		Formula implies Formula |
		not Formula |
		equals(XPath, XPath)
		isDescendant (XPath, XPath)
		isAssociated (XPath, XPath)
XPath	::=	absolutePath | relativePath

定義した規則記述文法を表 1 に示す．“XPath” は，XPath [13] のロケーション指定の表記法を用いる．また，整合性規則を一般的な形で記述できるように，ある要素が別の要素の子孫であること，二つの要素間にパスが存在することを，それぞれ表す述語 “isDescendant” と “isAssociated” を用意した．

表 1 の整合性規則記述文法では，整合性規則記述（Rule）はすべて，全称記号から始まっている．これは後述する整合性規則のすべてが，「各構成要素について，ある規則が満たされるべきである」という形の制約式として記述できると判断したからである．

2.3.2 整合性規則と記述例

具体的な整合性規則としては，文献 [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規則約 120 個を OCL で定義している．本研究ではそれらの中から，クラス図とシーケンス図に関する約 50 個の整合性規則を扱う．一般に，UML ダイアグラムの構成要素が満たすべき整合性規則が，本研究で扱う整合性規則で被覆できると証明することは困難 [5] なため，ユーザが必要に応じて追加取捨選択できるようにした．

文献 [5] のルールを整理し，カテゴリー別に表 2 にまとめる．

提案表記方法で記述するのが難しいと思われる事前事後条件インバリアント，抽象，インタフェースなど，ある程度意味にかかわる不整合のルールに関してはツールに組み込んでない．

採用したカテゴリー，例えば，クラス特性のカテゴリーのルールは次の二つである．「leaf 特性をもつ（子クラスをもたない）クラスは拡張できない」「root 特性をもつ（最上位の親クラスである）クラスは他のクラスを拡張しない」また，オペレーション・属性の可視性のカテゴリーのルールには以下のものが含まれ

表 2　文献 [5] の整合性ルール
Table 2　Rules in [5].

カテゴリー	個数
オペレーション・属性の可視性	4
型	6
オブジェクト，オペレーション，属性とクラスの関係	8
抽象	8
事前，事後条件，インバリアント	8
名前	4
属性の変更可能性	4
クラスの関連，多重度	12
クラス特性	2
インタフェース	10
オペレーション	7

```
 1: <rule>
 2:   <forall var="x" in="UML:Message">
 3:     <exists var="y" in="UML:Operation">
 4:       <and>
 5:         <equal op1="$x[@name]"
 6:                op2="$y[@name]" />
 7:         <isDescendant op1="$x[@receiver]"
 8:               op2="$y/../..[@xmi.id]"/>
 9:       </and>
10:     </exists>
11:   </forall>
12: </rule>
```

図 2　整合性規則記述例
Fig. 2　Example of consistency rules.

る．「他のクラスに属するオペレーションに呼び出されるプライベートなオペレーションはない（シーケンス図中）」

一方，採用しなかったカテゴリー，例えば，事前事後条件インバリアントのカテゴリーには次のルールが含まれる．「各事前条件は，クラスインバリアントを破っていてはならない」

「シーケンス図中のすべてのメッセージは，その受信オブジェクトの属するクラスで定義されていなければならない」という整合性規則を，提案した記述文法に従った XML で記述した例を図 2 に示す（実際には行番号はない）．

図 2 について説明する．1 行目と 12 行目は，規則の始まりと終わりを示すためのタグであり，これらの内側に具体的な整合性規則を記述する．2 行目から 11 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”）and
　　（“x/receiver”=”y/../../xmi.id”）））

論理積の左項は，メッセージ名とオペレーション（操作）名が等しいことを，右項は，メッセージの受信オブジェクトが属するクラスの xmi.id と，オペレーション（操作）の二つ上位の要素（このオペレーションを定義しているクラス）の xmi.id が等しいことを表している．このとき，（“x/name”＝“y/name”）∧（“x/receiver”=”y/../../xmi.id”）を満たさないような message 中の要素 x が，この整合性規則に違反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，モデリングツールによって要素に対して一意に割り当てられている ID（ID が割り当てられない要素はその親の ID），違反した規則番号によって保持し，これらの情報が，次節で述べる解消動作記述に使われる．

2.4　解消動作記述

本節では，解消動作記述について述べる．**2.3** で扱ったクラス図とシーケンス図に関する整合性規則それぞれに対して，その規則に違反した場合に実行すべき解消動作を記述する．このとき，実行すべき解消動作を求める必要があるが，以下の方針に基づいて考えられる解消動作を求めている．例えば制約式が「∀**var**∈**“XPath”** 式」の場合に考えられる解消動作は，次の 2 種類である．

（1）　変数 var に代入されている，式を “false” とするような要素を削除する（delete）

（2）　式を true にするように修正する（replace）

また，式が「∃**var**∈**“XPath”** 式」の場合に考えられる解消動作は次のとおりである．

（1）　式を true にするような，XPath に一致する要素を追加する（add）

（2）　XPath 上にある要素を選び，その要素において，式を “true” にする（replace）

解消動作記述には，XMI.difference [12] の表記を用いた．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，モデルの変更点のみを転送するという目的で用いられるもので，add，delete，replace という三つの動作を表す要素からなる．不整合の解消動作も，上述したように，add，delete，replace を使って記述することができる．

本研究では，XMI.difference 中に，XPath による

表 3　解消動作記述文法
Table 3　Syntax of repair actions.

```
Action   ::= Add | Delete | Replace
Add      ::= add VarDec hrefDec Desc Pri Ele
Delete   ::= delete VarDec hrefDec Desc Pri
Replace  ::= replace VarDec hrefDec Desc Pri Ele
Desc     ::= actionDescription
Pri      ::= diagramPriority
Ele      ::= ϵ | XMI Ele | XPath Ele | $var Ele
VarDec   ::= ϵ | var ∈ XPath VarDec
HrefDec  ::= ϵ | href ∈ XPath HrefDec | href ∈ $var HrefDec
```

ロケーションパス表記，変数宣言と，その参照を利用できるように拡張した．これにより，汎用的に解消動作式を記述することができる．

2.4.1　記 述 文 法

解消動作の文法を表 3 に示す．

上述のように，解消動作には Add，Delete，Replace の三つがあり，このうちの一つを利用して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>” タグに記述する．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る．要素 Add, Replace については，追加若しくは置換する要素 “Ele” をその子要素としてもつこととする．この要素 “Ele” は，UML モデル要素を XMI に準拠した形式で記述するが，この “XMI” において動的にしか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する．

また，ユーザから対話的入力を行うことでしか定められない値（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）に対しては，そのことを表す変数を用意した．

要素 “Pri” はその解消動作において優先度が一番高いダイアグラムの種類を表し，要素 “Desc” は自然言語で解消動作の説明を記述する部分を表す．

2.5　ツール概要

提案手法を実装したツール（Java，約 6000 行）について述べる．本ツールは，入力として XMI 形式で保存された UML モデルを与えると，出力として，初期不整合要素の位置情報と，それを修正する解消動作系列の集合をユーザに提示する．提示された解消動作系列の中から一つを選択すると，その解消動作を行った結果の XMI 形式のファイルを出力する．

これまで述べた以外のツールの機能を述べる．

2.5.1　解消動作のループ検出

解消動作を導出する際，ある不整合 C1 があり，その解消動作 A1 を実行した場合に不整合 C2 が起こるとき，C2 を解消する動作 A2 が A1 の逆向きの操作である場合（例えば，同じ要素の ADD と DELETE 操作）この部分でループが発生してしまう．本ツールでは，ある不整合が検出された場合に，それ以前に全く同様の不整合が検出されていないかどうかをチェックする．もし，同様の不整合が検出されている場合，その部分でループが発生するので，その不整合に関してはそれ以上の解消は行わず終了する．

2.5.2　解消動作の優先度付け

解消動作が多数導出された場合，その中からその場面に適した解消動作を選び出すことが，利用者には必要となる．そこで本手法では，利用者が解消動作を選ぶ基準として解消動作に優先度を付けて提示する．

(1) ダイアグラムの種類に応じた優先度付け：あらかじめ指定された優先ダイアグラムを優先する．(2) 解消動作（Add，Delete，Replace）の種類に応じた優先度付け：Delete 動作は，現在の状態を以前の状態に戻すものであり，有効な解消動作である可能性は低い．解消動作系列に含まれる Delete の数によって，その解消動作の優先度を低くする．

2.5.3　解消動作系列の提示

解消動作を，ユーザに提示することが必要となる．本手法のの特徴を生かし，解消動作記述と同時にその説明を，自然言語と，検出される不整合要素情報を示す変数を用いて一般的な形であらかじめ与える．解消動作系列は，これらの記述とともにユーザに提示する．

3.　例 題 適 用

本章では，前章で提案した手法についてプロトタイプツールを実装し，次節で述べる例題に適用する．

3.1 例題概要

ドリンクメーカー（DrinkMaker）は，コーヒーや紅茶などの飲み物を作る，レストランなどに置かれている機械である [17]．図 3 と図 4 が例題の UML ダイアグラムであり，それぞれクラス図と，カップに注入された紅茶にパウダーを入れるというユースケースをモデリングしたシーケンス図の一部である．それぞれのダイアグラムのサイズを表 4 に示す．

UML モデルで最頻出の不整合は，「関連の定義されていないクラス間でメッセージ交換を行うこと」だといわれている．ある産業用システムの UML ダイアグラムに対して調査したところ，関連を引くべき箇所のうち 77%の割合で関連の引き忘れがあったとしている．学生が作成した UML ダイアグラムにおいても，やはり 70%ほどの割合で関連の引き忘れがあったとしている．また，「メッセージに対応しているオペレーションがクラス図で定義されていない」という不整合もよく起こるということも報告されている [18]．

上記を考慮し，例題のダイアグラムには，以下の二つの不整合を埋め込んだ．

（1） シーケンス図中の DrinkMixer クラスのオブジェクトと，Tea クラスのオブジェクトとの間でやり取りされているメッセージ addPowder が，その受信オブジェクトのクラスである，クラス Tea 中のオペレーションで定義されていない
これは，Tea クラスが Drink クラスを継承することを忘れているために起こっている不整合である

（2） シーケンス図中で，DrinkMixer クラスのオブジェクトと，Tea クラスのオブジェクトとの間でメッセージがやり取りされているが，クラス図中で，それらのクラスの間に関連が引かれていない
これは，DrinkMixer クラスと Tea クラスの間に関連を引き忘れたために起こっている不整合である

提案手法を実装したツールに適用することで次の点を確認する．

（1） 上記の二つの不整合要素が正しく検出できるかどうか

（2） 検出された二つの不整合を解消する一連の解消動作系列の中に，それを実行することによって，本来意図していた UML ダイアグラムが導き出されるよ

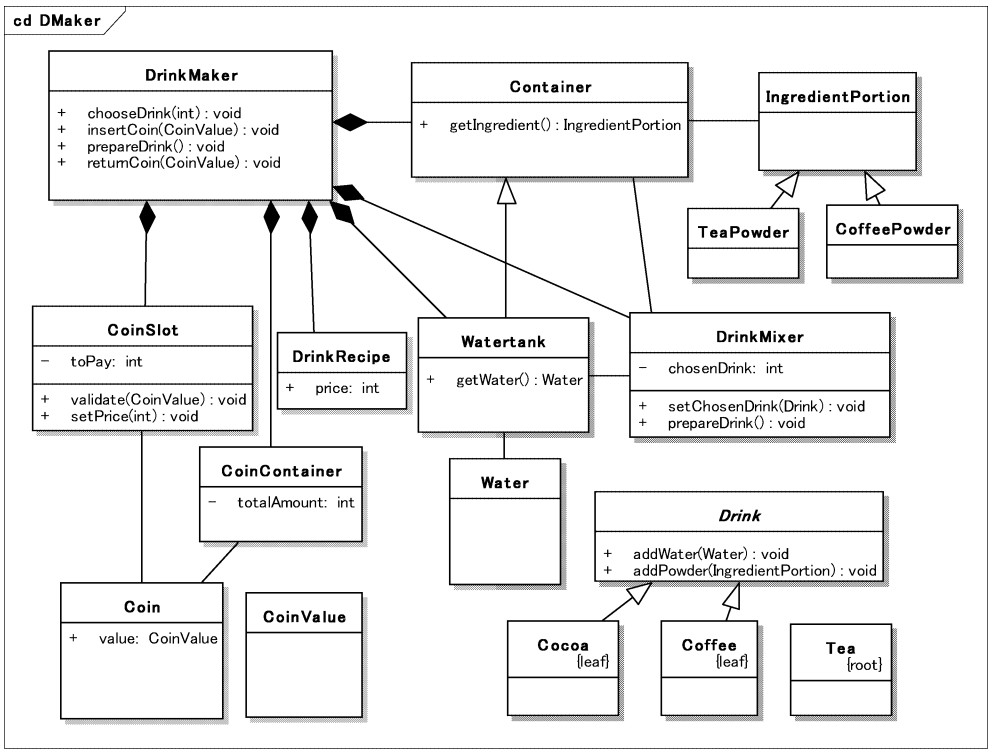

図 3 例題クラス図
Fig. 3 Class diagram.

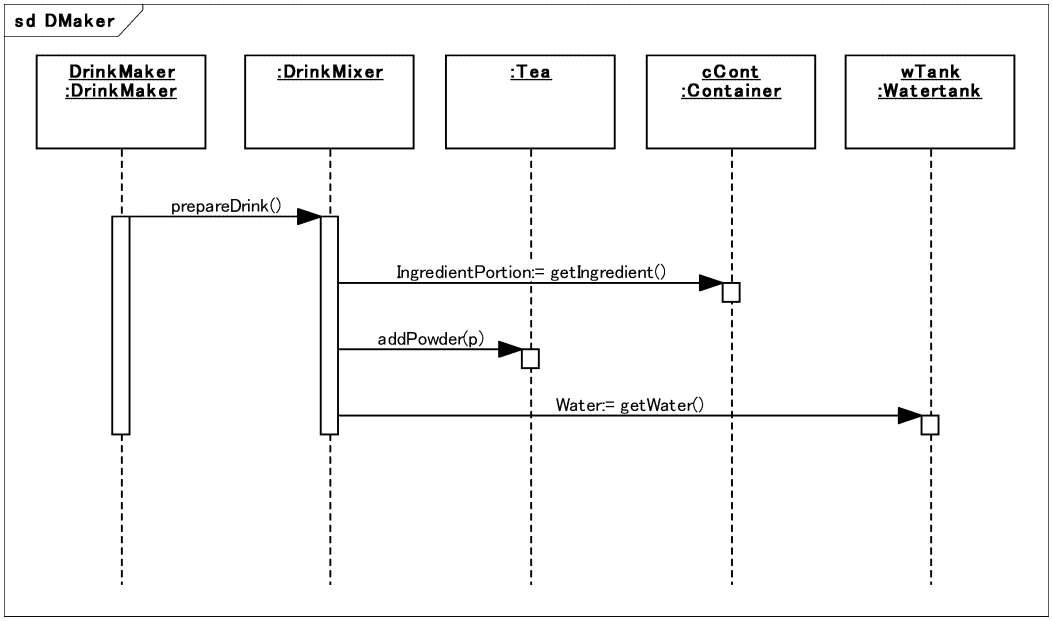

図 4　例題シーケンス図
Fig. 4　Sequence diagram.

表 4　例題のサイズ
Table 4　Size of the example.

クラス数	17	クラス間関連数	17
属性数	5	オブジェクト数	5
操作数	12	メッセージ数	10

うな解消動作系列が含まれているかどうか

（3） 解消動作のループを検出することができるかどうか

（4） 導出された解消動作系列の優先度付けが，多くの解消動作の中からその場面に適したものを選択することをサポートするという点において有効かどうか

3.2 結果と考察

3.2.1 不整合の検出について

初期不整合が二つ検出され，不整合が正しく検出されていることが確認できた．

検出時間は，UML モデル（XMI ファイル）のサイズ，適用規則の数に比例すると考えられる．用意した整合性規則記述 48 個を例題の UML モデル（約 2000 行）に適用し計測した結果，上記の初期不整合要素に対する検出時間は約 15 秒であった．

3.2.2 解消動作の導出について

前項で検出された不整合要素二つに対して解消動作を導出した．導出された不整合解消動作系列は全部で 21 個あった．

（1） クラス Tea がクラス Drink を継承する

（2） クラス Drink（あるいはクラス Tea）とクラス Mixer 間に関連を与える

（3） クラス Drink の root 特性を削除する

（この解消動作は，解消動作によって新しく発生した不整合を解消する動作となっている）

以上より，本来意図していた UML ダイアグラムに修正できる解消動作が導出されることが確認できた．

クラス Tea は root 特性をもっている（誤って付けられていたがそれまでは不整合ではなかった）ため，上記 1 の解消動作を行うことによって「親クラスをもつクラスが root 特性をもっている」という不整合が新たに発生することになるが，その不整合が検出され，解消する動作も導出されている．つまり，複数ステップに及ぶ解消動作を扱えていることが確認できる．

例題で使用されたルールを以下に列挙する．

「シーケンス図中のオブジェクトはクラス図中のクラスのインスタンスでなければならない」

「シーケンスメッセージ中で呼ばれる各オペレーションはクラス図中で定義されていなければならない」

「二つのオブジェクト間の各メッセージには，適切なパスがなければならない」

「leaf 特性をもつ（子クラスをもたない）クラスは拡

張できない」
「root 特性をもつ（最上位の親クラスである）クラスは他のクラスを拡張しない」
「同じ名前をもつような，同じパッケージ中の二つのクラスはない」
「同じ名前をもつような，同じクラス中の二つの属性はない」

表 2 で挙げたルールのうちこのツールで採用したルールに反する項目も当然チェックできる．例えばメソッドの可視，不可視に関する制約が子クラスに継承されそのオブジェクトの不可視メソッドがシーケンス図で使われているような誤りが発見できる．

3. 2. 3 解消動作のループ検出について

(1)「親クラスをもたないクラスを表す<<root>>特性が付いているクラスがあり，そのクラスが親クラスをもっている」という不整合と，(2)「シーケンス図中のメッセージのやり取りをしているオブジェクトの属するクラス間に適切なパスがない」という不整合との間でループが発生している．本ツールでは，この解消動作のループを 3 箇所で検出し，それぞれループが発生していたことを確認した．

3. 2. 4 解消動作の優先度について

クラス図，シーケンス図どちらのダイアグラムを優先する場合においても，その結果，優先度が高くなる解消動作系列が提示されることを確認した．

重点を置くダイアグラムがはっきりしている場合や，修正作業が項目の削除よりは項目の変更が主体である場合などは提案する優先度付けが多くの選択肢から有意なものを選択支援することにある程度有効であると思われる．一方，このような特徴が当てはまらない例も考えられる．

前者については，実際上は，優先ダイアグラムの存在は厳しい仮定ではないと考えられる．ただし，特定のダイアグラムのみ優先というような単純なものではなく，様々な観点に基づいた優先付けが必要な状況も実用上はあり得ると思われる．そのような場面での優先度の変更には比較的容易に対応できる．後者については，削除優先という方針が全面的に適用される状況は実用的には少ないと考えられる．ただし，適用範囲を限定しての適用はある程度有用と予想される．

いずれにせよ，導出された解消動作系列の中からその場面に適したものを選択することをサポートするという点において提案する優先度手法により，有効な選択支援が見込まれると期待される．

4. むすび

本論文では，UML ダイアグラム（クラス図とシーケンス図）の構成要素 UML モデルの各構成要素間が満たすべき静的な整合性を検査し，不整合が検出された場合にはその不整合の解消動作を行うための手法を提案し，その手法を実現するツールに関して述べた．不整合要素の検出と解消動作の実行を繰り返すことで，従来は扱えなかった，不整合の解消によって新たに発生する不整合を解消する動作をも，候補として提示することができる．また，導出された複数の解消動作から，適切な候補の選択支援をするため，優先度付けを行った．これらについて例題を通して有用性を確認した．今後の課題としては，クラス図，シーケンス図以外のダイアグラムへの拡張が考えられる．OCL 記述への対応も興味深い．

文　献

[1] M. Elaasar and L.C. Briand, “An overview of UML consistency management,” Carleton Univ., Technical Report, SCE-04-18, Aug. 2004.
[2] W. Shen, Y. Lu, and W.L. Low, “Extending the UML metamodel to support software refinement,” Workshop on Consistency Problems in UML-based Software Development II, pp.35–42, San Francisco, USA, Oct. 2003.
[3] B. Hnatkowska, L. Kuzuniarz, Z. Huzar, and L. Uzinkiewicz, “Refinement relationship between collaborations,” Workshop on Consistency Problems in UML-based Software Development II, pp.51–57, San Francisco, USA, Oct. 2003.
[4] J. Hausmann, R. Heckel, and S. Sauer, “Extended model relations with graphical consistency conditions,” Workshop on Consistency Problems in UML-based Software Development, pp.61–74, Dresden, Germany, Oct. 2002.
[5] L.C. Briand, Y. Labiche, and L. O’Sullivan, “Impact analysis and change management of UML models,” Proc. Int. Conf. on Software Maintenance, pp.256–265, Amsterdam, The Netherlands, Sept. 2003.
[6] L.C. Briand, Y. Labiche, and G. Soccar, “Automating impact analysis and regression test selection based on UML designs,” Proc. Int. Conf. on Software Maintenance, pp.252–261, Montreal, Canada, Oct. 2002.
[7] B. Litvak, S. Tyszberowicz, and A. Yehudai, “Behavioral consistency validation of UML diagrams,” Proc. First Int. Conf. on Software Engineering and Formal Methods, pp.118–125, Brisbane, Australia, Sept. 2003.
[8] 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，pp.573–577, Sept. 2005.
[9] C. Nentwich, W. Emmerich, and A. Finkelstein, “Flexible consistency checking,” ACM Trans. Softw. Eng. Methodol., vol.12, no.1, pp.28–63, Jan. 2003.
[10] C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein, “xlinkit: A consistency checking and smart link generation service,” ACM Trans. Internet Technology, vol.2, no.2, pp.151–185, May 2002.
[11] C. Nentwich, W. Emmerich, and A. Finkelstein, “Consistency management with repair actions,” Proc. 25th Int. Conf. on Software Engineering, pp.455–464, Portland, USA, May 2003.
[12] OMG. UML 2.0 Diagram interchange specification, 2005. http://www.omg.org/docs/ptc/05-06-04.pdf
[13] XML Path Language (XPath) 2.0. http://www.w3.org/TR/2005/WD-xpath20-20050404/
[14] OMG. UML 2.0 OCL specification, 2003. http://www.omg.org/docs/ptc/03-10-14.pdf
[15] R. Wagner, H. Giese, and U.A. Nickel, “A plug-in for flexible and incremental consistency management,” Workshop Consistency Problems in UML-based Software Development II, pp.78–85, San Francisco, USA, Oct. 2003.
[16] 佐々木亨，岡野浩三，楠本真二，“UML モデルに対する XPath と XMI-difference を用いた不整合検出と解消,” 信学技報，SS2005-48, 2005.
[17] K. Ludwik and S. Miroslaw, “Inconsistencies in student designs,” Workshop on Consistency Problems in UML-based Software Development II, pp.9–17, San Francisco, USA, Oct. 2003.
[18] C. Lange, M.R.V. Chaudron, J. Muskens , L.J. Somers, and H.M. Dortmans, “An empirical investigation in quantifying inconsistency and incompleteness of UML designs,” Workshop Consistency Problems in UML-based Software Development II, pp.26–34, San Francisco, USA, Oct. 2003.

（平成 18 年 7 月 14 日受付，10 月 31 日再受付）

佐々木　亨

平 16 阪大・基礎工・情報卒．平 18 同大大学院前期課程了．現在，日本オラクル（株）勤務．在学中は UML の一貫性に関する研究に従事．

岡野　浩三　（正員）

平 2 阪大・基礎工・情報卒．平 5 同大大学院博士後期課程中退．同年同大助手，平 14 ケント大客員研究員．平 15 バーミンガム大客員講師．現在，阪大・情報・コンピュータサイエンス・助教授．博士（工学）．形式手法によるソフトウェア開発・検証方法などの研究に従事．情報処理学会，IEEE CS 各会員．

楠本　真二　（正員）

昭 63 阪大・基礎工・情報卒．平 3 同大大学院博士課程中退．同年同大・基礎工・情報・助手．平 8 同学科講師．平 11 同学科助教授．平 14 阪大・情報・コンピュータサイエンス・助教授．平 17 同教授．博士（工学）．ソフトウェアの生産性や品質の定量的評価，プロジェクト管理に関する研究に従事．情報処理学会，IEEE，JFPUG, PM 各会員．