ステレオタイプによるUMLモデル間の整合性検証支援手法

鷲見 毅[†] 渡辺晴美[†] 大西 淳[†]

UMLは汎用的な言語であり，分析，設計，実装のすべての段階で利用されるモデルを表記できる．UMLでは種々の観点からシステムをとらえることができるように，複数のモデルを用意している．分析段階では，6つのモデル図が用いられる．しかし，UMLは表記法であり，方法論を規定していない．そのため，方法論によっては整合性を十分に保証できない場合がある．本研究では，UMLモデル間の整合性検証支援を目標として，UMLモデル間関連をステレオタイプを用いて明確にし，モデル間関連とモデルの意味記述から矛盾を検出する手法を提案する．

Supporting Method for Verifying the Consistency between UML Models with Stereotypes

Takeshi Sumi,[†] Harumi Watanabe[†] and Atsushi Ohnishi[†]

UML is a widely applicable modeling language and supports several modeling diagrams for object-oriented analysis, object-oriented design, and object-oriented programming phases. Since UML is just a modeling language, UML model definers can adopt a certain software development methodology. There exists a problem that the guarantee of the consistency among UML models depend on the selected methodology. We propose a method to detect inconsistent elements between two UML models by defining relations between these two models with stereotypes and by checking the inconsistency with the relations and the semantics of the models.

1. はじめに

1.1 背景と問題点

OMG（Object Management Group）によって，標準のモデリング言語として採用されたUML（Unified Modeling Language）は，オブジェクト指向分析・設計で用いられるモデリング言語として広く利用されるようになった[1)~4)]．

UMLは方法論を規定していないため，方法論によってはモデル間の整合性を十分に保証できないという問題点がある．またUMLでは，ある視点で記述したUMLモデル（たとえば，シーケンス図）を基に，別の視点のUMLモデル（たとえば，状態遷移図）を作成しようとした場合に，基となるモデルのどの部分を参照してモデルを導出すればよいかについて，明確な規則がない．このようなことから，UMLモデルを作成した段階で，モデル間に矛盾が存在する可能性がある．

しかし，UMLモデル間の矛盾を検出し，整合性を検証するための，“確立された手法”は存在しない．そのため，モデル間の整合性検証はいまだに困難な作業のままであり，検証の精度は開発者の技量に依存している．矛盾を含有したモデルを用いて分析・設計，実装を進めた場合，深刻なバグや手戻りが生じる結果となる．このような問題を解決するために，作成されたUMLモデル間で整合性を検証する手法を確立する必要がある．

1.2 本研究の目的

本研究では，UMLモデルの矛盾を検出するために，実世界のシステムを構成する要素間で成り立っている関係を利用する．ここで実世界のシステムを構成する要素とは，システムの処理やクラス，操作や属性を指す．これらの要素間には，様々な関係が成り立つ．たとえば，イベントやメッセージの順序関係，クラスが属性や操作を所有することを表す所有関係が考えられる．これらの関係は，UMLモデルにモデリングされた後もモデル要素間で保たれていなければならない．

本手法では，実世界のシステムにおいて関係を持つ要素が，それぞれどの要素にモデリングされているかを明確にし，対応するモデル要素間の関係が互いに矛盾していないかを検証することによってモデル間の矛盾を検出する．

† 立命館大学理工学部情報学科
Department of Computer Science, Ritsumeikan University

本手法で矛盾を検出するためには，UMLモデルの要素間の対応がとれていなければならない．UMLモデルは単体で意味を持つが，モデルが互いに完全に独立しているわけではない．このため，シーケンス図のあるメッセージが，状態遷移図の遷移イベントに相当するといったことがありうる．しかしUMLの記法では，モデル要素間の関連を明確に定義できないため，開発者自身がモデル要素間の対応を判断，解釈しなくてはならなかった．UMLのセマンティクスでも，個々の要素が何を表しているかは定義されているが，その要素が対応関係を持ちうる要素については定義されていない[15]．そこで，このモデル要素間の関連を示すために，ステレオタイプを定義し，これを用いることを提案する．

我々は，これまでにモデル要素間の関連を明確にすることでモデル間の矛盾を検出できることを確認している[10]．本稿では，モデル要素間で満たされる関係と，これを満たすべきモデル要素を定義している．そして，実世界のシステムをモデリングする関数を定義し，実世界のシステムにおける関係を矛盾の検出に利用している．

我々の手法は，モデル作成者に対してモデルの検証に必要な情報を記述することを要求するほかに，あるモデルを記述する際に他のモデルと関係が深い情報を今記述していることをモデル作成者に対して明確に認識させ，モデル作成者の備忘録としてステレオタイプを記入するという使い方も可能である．

1.3 関連研究

関連研究として，我々が進めてきたUMLモデル間の整合性検証についての研究[5]がある．この研究での整合性検証は，構文的な検証に限定されている．本稿で提案する手法では構文的な検証に加えて，意味的な検証が可能である．

また，UMLモデリングツールであるRoseでは，モデルチェック機能をサポートしている．しかし，Roseのモデルチェック機能では，RUP（Rational Unified Process）という方法論に基づいてモデル要素間の関連付けを行っている．これに対し本研究の手法では，方法論によらず，要素の対応関係に基づいて関連付けを行っているため，RUP以外の方法論によって記述されたモデルに対しても適用できる．また，Roseでは要素の関連付けの正しさについてのみ検証を行っているが，本手法では，関連付けの正しさに加え，要素間で成立する関係についての検証も可能にしている．

UMLモデルではないが，OMTモデルに対して形式的な意味を与えることによって一貫性検証を可能とした研究[8]がある．この研究では機能モデルの中のデータフローが矛盾なく流れるかどうかを証明できることを示しているが，モデル間の整合性については述べられていない．

OMT法では，UMLのシーケンス図に相当する事象トレース図から状態遷移図を導く方法が示されている[11]．またシュレイアーらは，状態遷移図とアクションデータフロー図を関連付けながら記述する方法を示している[12),13]．これらの方法を用いた場合にも，モデル間の整合性を保証する手法が必要である．

UMLモデルの意味を与える研究としては，pUML（precise UML）[6]がある．形式的な意味をUMLモデルとは別に定義することによって，意味的な検証が可能となる．また，OCLによる制約条件を用いたUMLモデル間の整合性検査手法と不整合が検出されたときの修正支援手法が提案されており[7]，シーケンス図と状態遷移図ではともに対応するイベントが明示されているという前提を置いているが，状態遷移図に表れるイベントと同じ要素がシーケンス図に表れるとは限らない．

我々は，ステレオタイプを用いてモデルの要素を対応付けることによって，モデル間の矛盾の検出を行う．整合性検証にステレオタイプを用いた研究としては，クラス図とシナリオの整合性検証についての研究がある[9]．この研究では，クラス図にシナリオを表すボックスを記述し，シナリオと各クラスの関連について，ステレオタイプを用いて記述している．本研究では，異なるモデル間のモデル要素を関連付けるためにステレオタイプを用いている．これによって，クラスに限らず，イベントやメッセージといった様々なモデル要素について，その関連を記述することができるようにした．

2. 手　　法

本手法では，実世界における同一の概念を異なる要素で表す場合に，要素間に存在する対応関係と，異なる概念間で成り立っている関係を用いて矛盾の検出を行う．本手法では，同一の概念を表す要素間の対応関係をモデル間関連と呼び，異なる概念間で成り立つ関係を，モデル間の関係と呼んでいる．2.1節でモデル間関連について，2.2節でモデル間の関係について述べる．

2.1 モデル間関連

実世界のシステムの要素には，イベントや相互作用，属性，操作などがある．これらの要素間には，様々な対応関係が存在する．たとえば，『イベント E_1 はメッ

セージ M_1 の受信である』といったことが考えられる．この場合,『イベント E_1』と『メッセージ M_1 の受信』という異なる要素が，同一の概念を表しているといえる．また,『オブジェクト A_1 とオブジェクト B_1 の間に存在する関連 R_1 は，オブジェクト A_1 とオブジェクト B_1 間のメッセージ M_1 に相当する』といった場合も同様のことがいえる．この場合にも,『オブジェクト A_1 とオブジェクト B_1 の間に存在する関連 R_1』と『オブジェクト A_1 とオブジェクト B_1 間のメッセージ M_1』という異なる要素が，同一の概念を表しているといえる．本研究では，このような要素間の対応関係をモデル間関連と呼び，次のように定義する.「実世界のシステムにおいて同一の概念が，異なるモデル要素によってモデルに表されるとき，その要素間に存在する対応関係をモデル間関連と定義する」

このモデル間関連 X_i は，以下のような特徴を持つと考えられる．

i) モデル間関連 X_i を持つ要素には，タイプがある．
ii) あるタイプの要素から別のタイプの要素への写像として，モデル間関連 X_i が定義される．

先のイベントとメッセージの例では，モデル間関連 X_i が『受信する』という関連であるとすると，メッセージ受信型のイベント E_1 と関連 X_i を持つ要素は，受信型のメッセージ M_1 であるといえる．すなわち，E_1=$X_i(M_1)$ が成り立つ．同様に，関連とメッセージの例でも，関連 X_i が『相当する』という関連であるとすると，$R_1 = X_i\,(M_1)$ が成り立つ．

2.2 モデル間の関係

実世界のシステムの要素間で成り立つ，関係 r について考える．ここで，実世界のシステムの要素 a，b 間で関係 r が成り立っている事を r{a，b} と記述する．このとき，この要素 a，b とモデル間関連を持つ要素，すなわち，a'=X_i(a)，b'=X_i(b) となる要素について，次のようになる関係 r が存在する場合がある．

r{a, b} ⇔ r{a', b'}

これは，実世界のシステムの要素 a，b 間で，関係 r が成り立つことと，a，b とモデル間関連を持つ a'，b' の間で関係 r が成り立つことが必要十分であることを意味する．このような関係 r が存在した場合に，

r{a, b} ⇔ r{a', b'}

を検証することによって，矛盾を検出できる．このような関係 r は，実世界のシステムの要素 a，b，a'，b' に依存した関係である．すなわち，要素 a，b，a'，b' が特定されなければ，その間で成り立つ関係 r は特定されない．関係 r には，イベントやメッセージの順序関係，メッセージや依存の方向関係，属性や操作の所有関係や参照関係があると我々は考えている．

2.3 モデリング関数

本研究では，実世界のシステムにおける関係を利用してモデル間の矛盾を検出する．そこで，実世界のシステムの要素を UML モデルの要素にモデリングする，以下のような関数 f_i を考える．

- f_i は実世界のシステムの要素（処理や属性）を，UML の要素にモデリングする．
- モデリング関数 f_i は，各モデルの要素ごとに，異なる関数として定義される．
- モデリング関数 f_i は，実世界のシステムの要素の集合 A の部分集合 Aj から，UML モデル要素に対しての，1 対多かつ onto な関数.
  ここで 1 対多は，同一の内容をモデリングしていても，その記述結果が開発者によって異なることに対応している．また onto は，開発者がモデリングしようとする対象は，すべてモデルとして記述可能であると仮定しているからである．
- 1 対多かつ onto な関数なので，f_i は逆関数を持つ．

このモデリング関数 f_i を用いてモデリングされたモデル間では，以下のことを仮定する.「実世界のシステムの要素を a，b とし，a，b 間で関係 r が成り立つとき，モデリングされた要素 f_i(a)，f_i(b) においても，関係 r が成り立つ」

これを，

r{a, b} ⇒ r{f_i(a), f_i(b)}

と表現する．たとえば，r を順序関係とし，実世界のシステムにおけるイベントが a，b の順に起きるとき，モデリングされた要素においても f_i (a)，f_i (b) の順にイベントが起きるとする．

f_i をこのように定義すると，

r{a, b} ⇔ r{X_i(a), X_i(b)}

を検証することは，

r{f_i(a), f_i(b)} ⇔ r{f_i(X_i(a)), f_i(X_i(b))}

を検証することと同義となる．すなわち，矛盾の検出とは，UML モデルにおいて，モデル要素 f_i (a)，f_i (b)，f_i (X_i (a))，f_i (X_i(b)) を特定し，その間の関係 r が互いに整合しているかを検証することである（**図 1**）．

これらのことから，UML モデル間でモデル間関連を持つ要素と，その要素間の関係が整合しているかを検証することによって，モデル間の矛盾を検出することが可能であるといえる．

2.4 モデル間意味

関数 f_i を定義したとき，モデル要素間で成立すべき事柄をモデル間意味として，以下のように定義する.「実世界のシステムの要素 a，b と，これとモデル

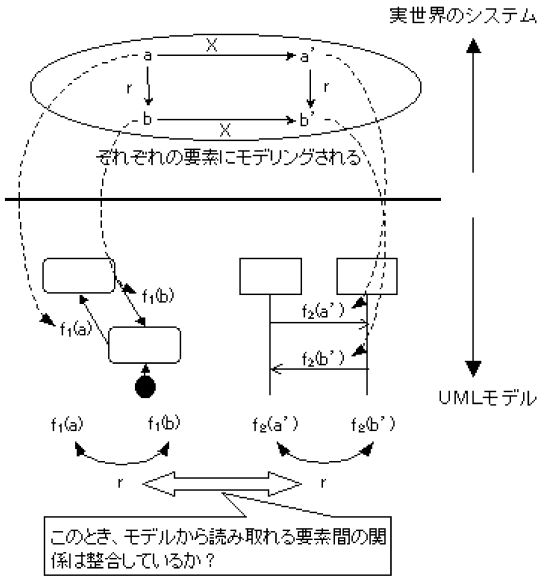

図 1 手法の概要
Fig. 1 Outline of our proposed method.

間関連を持つ要素 a'=X_i(a)，b'=X_i(b) が，それぞれ関係 r を満たすとき，これらがモデリングされた要素 f_1(a)，f_1(b)，f_2(a')，f_2(b') において，

r{f_1(a), f_1(b)} ⇔ r{f_2(a'), f_2(b')}

となる事柄」

これは，モデル間関連を持つ実世界のシステムの要素間で成立する関係 r は，モデリングしたモデル要素間でも同様に成立しなければならないということである．そしてこれは，実世界のシステムの要素 a，b と，これとモデル間関連を持つ要素 a'，b' の種類を定義し，その要素間において成立する関係 r を定義することといえる．要素間で満たされる関係 r の具体例としては，モデルに要素が存在するという関係，要素が一致するという関係，イベントやメッセージの時間的順序関係やメソッドの使用関係，クラスの変数に対する所有関係が考えられる．

このような関係と，それを満たす要素を定義したモデル間意味の具体的な例として，以下の 5 つをあげる．モデル間意味はこれ以外にも存在するが，ここではシーケンス図と状態遷移図の間で定義されるモデル間意味をあげる．

(1) シーケンス図に記述されたあるオブジェクトのメッセージ順が，そのクラスの状態遷移図の遷移を駆動するイベント順と矛盾しない．
(2) シーケンス図で消滅するオブジェクトは，状態遷移図では終了状態となる．
(3) 状態遷移図におけるアクション，アクティビティの要求元，要求先となるオブジェクトがシーケンス図に存在する．
(4) 状態遷移図におけるアクション，アクティビティの要求元，要求先がシーケンス図の送信元，送信先と一致する．
(5) 状態遷移図，シーケンス図に記述される変数やメソッドは，クラス図においてクラスが持つ属性，操作として定義されている．

2.5 モデル間関連の識別

モデル間関連を持つ要素は，それぞれ異なるモデルにモデリングされており，その対応は明確でない場合が多い．そこで，本研究ではこの対応を示すために，ステレオタイプを利用する．たとえば，『イベント E は，メッセージ M を受信するイベントである』ことを表すために，メッセージ受信型のステレオタイプを定義し，イベントに付加することによって，イベントとメッセージの対応を得ようとするものである．これは，あるモデル要素のタイプを表すものである．これによって，ステレオタイプが付けられた要素とモデル間関連を持ちうる要素の種類が決定される．

現段階では，状態遷移図とシーケンス図においてモデル間関連を示すために，以下の 5 つのステレオタイプを用意している．利用者は，定義されているステレオタイプの中から適切なステレオタイプを選択し，要素に付けてモデル間関連を示す．それぞれステレオタイプについて，以下に説明する．

- イベントに付けられるステレオタイプ．
  ≪ MsgSnd ≫ ：他オブジェクトにメッセージを送信．
  ≪ MsgRec ≫ ：他オブジェクトからメッセージを受信．
- アクション，アクティビティに付けられるステレオタイプ．
  ≪ update ≫ ：自オブジェクトが持つ値を更新．
  ≪ notice ≫ ：自身が持つ値を他オブジェクトに通知．
  ≪ call ≫ ：他オブジェクトのメソッドを呼び出す．

2.6 矛盾の検出手順

本研究で扱う矛盾は，

(1) ステレオタイプが付けられているのにモデル間関連を持つ要素がない，
(2) モデル間意味を満たさない記述がモデルに存在する，

という 2 つの場合が考えられる．

以下に，本手法による矛盾の検出手順を説明する．

A) モデル要素に，要素のタイプを表すステレオタイ

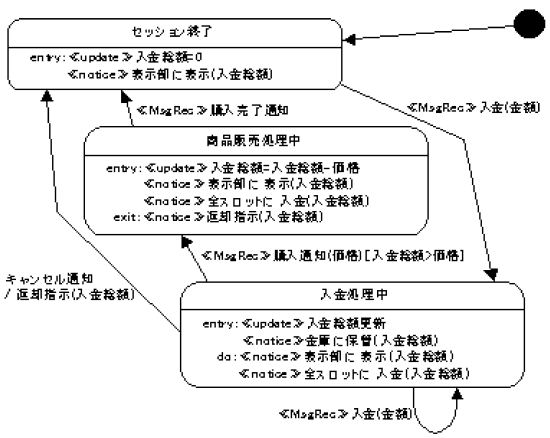

図 2 セッション管理の状態遷移図
Fig. 2 State chart of session control class.

プをモデル作成者が付ける．

・モデル作成者にとって要素のタイプは容易に判断できると我々は考えており，モデル作成者に要素のタイプを判断してもらい，ステレオタイプをつけてもらう．たとえば，メッセージを受信するというイベントは，メッセージ受信型のイベントであるので，メッセージ受信型を表す≪ MsgRec ≫を付けてもらう．

B) ステレオタイプが付けられたモデル要素に対応しうる要素を計算機によって示す．
C) 対応する要素が一意に決まらない場合は，利用者が候補の中から対応する要素を選択する．
D) 対応する要素がない場合は，計算機は矛盾として提示する．
E) モデル間関連を持つモデル要素間について，要素間の関係を利用者はまとめる．
F) モデル間関連を持つモデル要素間の関係が，モデル間で整合しているかを計算機によって検証する．
G) 計算機は検出された矛盾を利用者に提示する．

3. 適 用 例

3.1 例 題

適用例として，缶飲料の自動販売機について考える[14]．この例では**図 2** に示す「セッション管理クラス」の状態遷移図と，**図 3** に示す「自動販売機からの缶飲料購入」についてのシーケンス図の間で，手法を適用し，矛盾の検出を行う．

なお，この例題では，説明のために意図的に状態遷移図とシーケンス図の間で矛盾のある記述をしている．以下に，例として用いる「缶飲料の自動販売機システム」の各クラスについて述べる．

(1) 客は，自動販売機のアクターである．
(2) セッション管理は，自動販売機における一連の缶飲料購入処理を管理する．
(3) 入金部は，客からの入金を受け付ける．
(4) 表示部は，入金総額を表示する．
(5) 缶飲料の価格は，すべて 120 円とする．
(6) 缶飲料スロットは，入金総額が缶飲料の価格以上になれば，購入可能と判断し，ランプを点灯する．
(7) 缶飲料スロットは，入金総額が缶飲料の価格以下になれば，購入不可と判断し，ランプを消灯する．
(8) 缶飲料スロットは，缶飲料の種類ごとにオブジェクトが生成される．
(9) 金庫は，入金されたお金を保管する．
(10) 金庫は，セッション管理からの指示に従って返却口に返却額を通知する．
(11) 取り出し口は，購入された缶飲料を客に渡す．
(12) 返却口は，釣銭を返す．

セッションとは，入金されてから商品と釣銭を排出するまでの一連の処理を指す．セッション管理クラスは，この一連の処理を管理するクラスである．図 2 は，このセッション管理クラスの状態遷移図である．

また，図 3 のシーケンス図は，客が 200 円を入金して缶飲料 B を購入し，釣銭を受け取るという一連の流れを表している．図 3 のシーケンス図の開始時におけるセッション管理クラスの状態は，セッション終了状態である．

3.2 適 用 手 順

上記の例題に対して手法を適用し，矛盾を検出する具体的な手順について述べる．2.6 節で示した手順に対応させて説明する．

A)-1 状態遷移図中の商品販売処理中状態の entry アクティビティ「入金総額=入金総額 − 価格」に≪ update ≫を付ける．
B)-1 シーケンス図の要素から,「入金総額=入金総額 − 価格」とモデル間関連を持ちうる要素をあげる．≪ update ≫は自身が持つ値を更新するので，自身へのメッセージと対応しうる．そのため,「入金総額=200」と「入金総額=入金総額 − 価格」,「入金総額=0」が候補としてあげられる．
C)-1 システムは，このいずれのメッセージと対応するかを利用者に問い合わせ，利用者から「入金総額=入金総額 − 価格」と対応するという回答を得る．

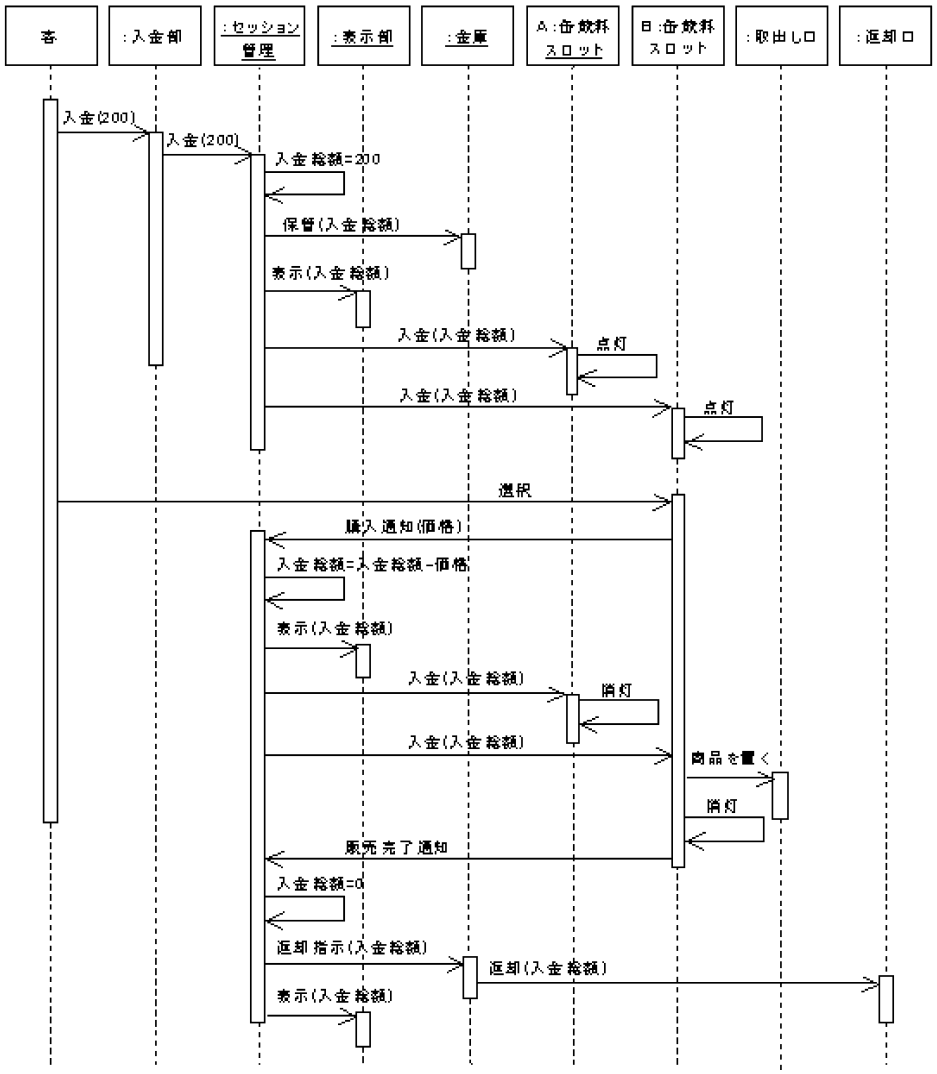

図3 缶飲料購入のシーケンス図
Fig. 3 Sequence chart of purchasing canned drinks.

A)-2 続いて「表示部に表示(入金総額)」というentryアクティビティに，≪notice≫を付ける．

B)-2 シーケンス図の要素から,「表示(入金総額)」という3つのメッセージが,「表示部に表示(入金総額)」というentryアクティビティに対応する要素の候補として挙げられる．

C)-2 システムは，どのメッセージと対応するかを利用者に問い合わせ，利用者から,「入金総額=入金総額 − 価格」の直後のメッセージに対応するという回答を得る．

A)-3 以下，同様にして各要素についてステレオタイプを付け，対応付けていく．

D) ステレオタイプが付けられた要素について，モデル間関連を持つ要素があることを確認する．

E) モデル間関連を持つ要素について，要素間の関係をまとめる．

E)-1 入金部からのメッセージ「入金(200)」を受信後に,「入金総額=200」メッセージを自身に送信する．

E)-2「入金総額=200」メッセージを自身に送信後に，表示部に「表示(入金総額)」を送信する．

E)-3 以下，順に要素間の関係をまとめる．

F) モデル間関連を持つ要素について，まとめられた関係が整合しているかを検証し，矛盾を検出する．

G) 検出できた矛盾を利用者に提示する．

セッション管理の状態遷移図中のイベントやアクションやアクティビティにいずれかのステレオタイプがつけられた場合，シーケンス図中の「セッション管理」

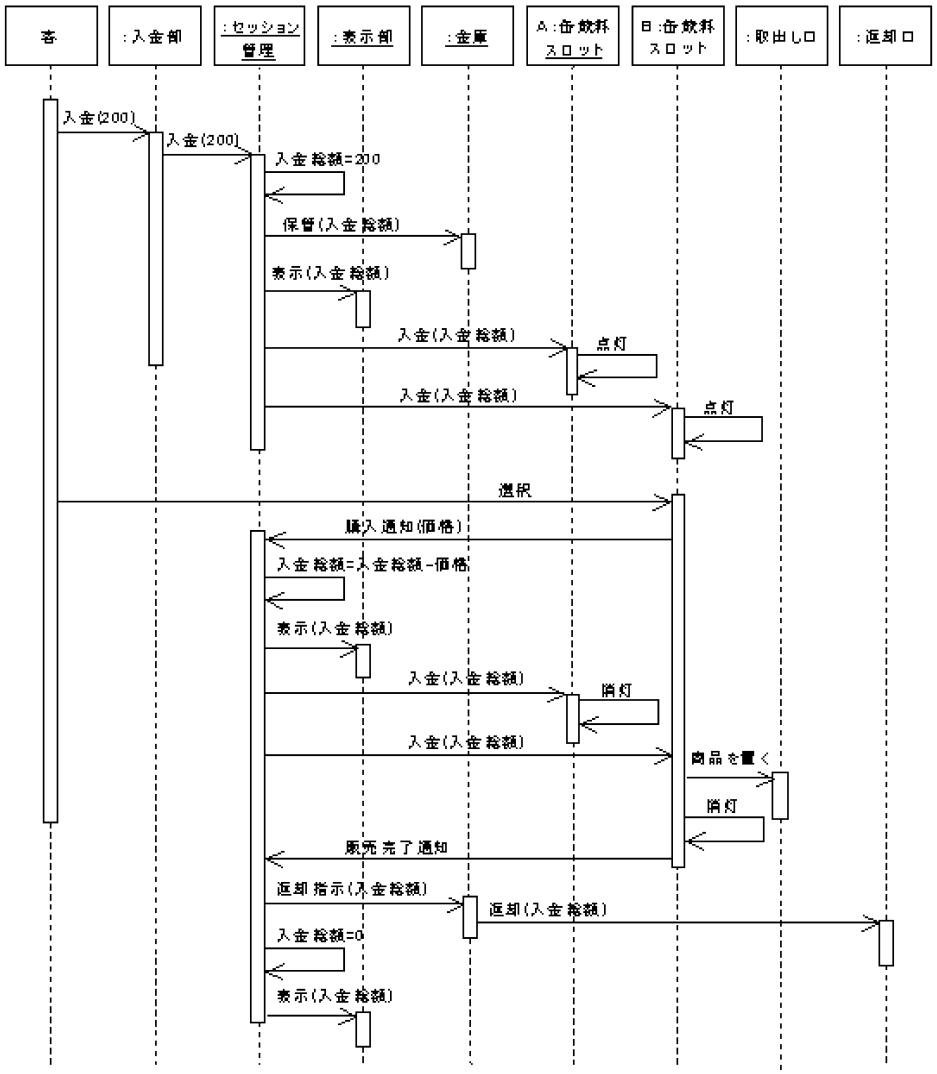

図 4 修正された缶飲料購入のシーケンス図
Fig. 4 Corrected sequence chart of purchasing canned drinks.

オブジェクトが送信するメッセージ，あるいは受信するメッセージと要素間関連があることになる．図 2 の状態遷移図には同じステレオタイプを持つ構成要素が複数現れている．また図 3 のシーケンス図でも，セッション管理オブジェクトから出されるメッセージや，それが受け取るメッセージも複数存在する．

このようにステレオタイプを付けた場合，一般にモデル要素間の対応は 1 対多となる．これは，実世界のシステムにおいてモデル間関連 X_i を持つ要素をモデリングした際に，モデリングされた要素の抽象度がモデルごとに異なるためである．このため，モデル間関連のある具体的な要素を利用者に問い合わせることによって，対応をとるようにしている．その際，ステレオタイプによって対応しうる要素を絞ることができる．これは，モデル間関連 X_i が特定できれば，そのモデル間関連 X_i を持つ要素の種類が特定できるからである．

3.3 適用結果

モデル間関連を対応付け，矛盾を検証した結果，以下の矛盾を検出できた．

- シーケンス図の自身へのメッセージ「入金総額=0」と表示部へのメッセージ「表示 (入金総額)」の順序と，状態遷移図のアクション「返却指示 (入金総額)」とセッション終了状態の entry アクティビティ「入金総額=0」の順序が矛盾する

手法を適用し，モデル間関連によってモデルを対応付けた結果，シーケンス図の「B：缶飲料スロット」からのメッセージ「購入完了通知」とイベント「購入完

了通知」，メッセージ「返却指示 (入金総額)」とアクション「返却指示 (入金総額)」，そして自身へのメッセージ「入金総額=0」とセッション終了状態の entry アクティビティ「入金総額=0」がそれぞれ対応付いていることが分かる．

この状態遷移図では,「購入完了通知」のラベルが付いたイベントに付随するアクションとして「返却指示 (入金総額)」が記述されている．そして，状態遷移後の,「セッション終了状態」の entry アクティビティとして「入金総額=0」が記述されている．しかし，これらとモデル間関連を持つメッセージ「入金総額=0」と「返却指示 (入金総額)」の順序は，状態遷移図の記述と逆になっている．これは,『シーケンス図に記述されたあるオブジェクトのメッセージ順が，そのクラスの状態遷移図の遷移を駆動するイベント順と矛盾しない』を満たさないので，矛盾といえる．この場合は，矛盾の検出を基にしてモデル作成者が検討した結果，シーケンス図においてメッセージの順序が間違っていたことが判明する．その結果，シーケンス図は**図 4** のように修正される．

このように，ステレオタイプによってモデル間関連を付けることによって，要素間の矛盾を検出できる．矛盾を検出した場合は，

(1) 矛盾を検出したモデルの双方が間違っている，
(2) 片方のモデルが間違っている，
(3) ステレオタイプによる関連付けが間違っている，
(4) モデル要素間の対応付けが間違っている，

のいずれかが考えられるが，現段階では矛盾の修正はモデル作成者に任すようにしている．本稿では，説明のために矛盾を混入した例題を用いている．しかし，意図的に矛盾を混入していない実際の例題にも適用し，矛盾を検出できていることから，手法は有用であると判断できる．また，適用例で検出できた矛盾は，モデル要素間の関係についての矛盾である．これは，本手法によって意味的な検証を行ったからこそ検出できた矛盾である．

4. おわりに

4.1 まとめ

本稿では，UML のモデル間関連を，ステレオタイプを用いて明確化し，モデル間意味と照らし合わせることによって，矛盾を検出する手法を提案した．モデル間関連を明確化する際，ユーザ自身がステレオタイプを用いて要素間を関連付け，さらに対応する要素を指定しなければならないため手間がかかるが，モデル間関連を明確化し，これを用いて要素間の関係を検証することによって，一部の矛盾を検出できることが確認できた．

ステレオタイプを要素に付加することによってモデル間関連を明確化するため，すでに作成された UML モデルに対して適用できる．この手法は，UML モデル作成時に，従来の UML の表記法をそのまま用いることができるため，新たな記法を学ぶ必要がない．また，ステレオタイプやモデル間意味を新たに定義することによって，より多くの矛盾を検出することができる．

さらにモデル作成者に対してモデルの検証に必要な情報を記述することを要求するほかに，あるモデルを記述する際に他のモデルと関係が深い情報を今記述していることをモデル作成者に対して明確に認識させ，モデル作成者の備忘録としてステレオタイプを記入するという使い方も可能であり，備忘録として記録した内容の整合性を検証できる．

4.2 今後の課題

本研究の手法では，モデル間意味を定義した項目について検証を行う．したがって，別の検証を行うためには，それに対応したモデル間意味[5)]を定義することによって対処できる．

モデル間意味を整理すれば，つねに成立すべきモデル間意味と，ある特定のドメインや特殊な状況でのみ成立する意味とに分類できると我々は考えている．本稿 2.4 節に示したモデル間意味は，ドメインに依存せずに成立すべきものである．このようなつねに成立すべきモデル間意味はデフォルトで用意してやり，特定のドメインや特殊な状況でのみ成立するモデル間意味については利用者がステレオタイプを追加し，モデル間意味を定義することによって，本稿で議論した検証が可能となろう．

今後はモデルの要素間の関係 r と，r を満たす要素のタイプについて，利用者が定義できる環境を提供するとともに，モデル間意味を定義できる言語，およびその処理系の開発と導入を進めていきたい．

謝辞 本研究にあたって，本学大学院理工学研究科修士課程 1 回生の長田真寿美氏をはじめとする，大西研究室の諸君に感謝する．本研究は一部文部科学省科学研究費補助金基盤研究（C）13680427 による．

参考文献

1) グラディ・ブーチ：UML ユーザガイド Version 1.3，オージス総研オブジェクト技術ソリューション事業部（訳），ピアソン・エデュケーション (1999).
2) Booch, G., Rumbaugh, J. and Jacobson, I.: *The Unified Modeling Language User Guide*,

Addison-Wesley (1999).
3) Albir, S.S.: UML クイックリファレンス，オブジェクト指向研究会（訳），オライリー・ジャパン (1999).
4) ペルディタ・スティーブンス，ロブ・プーリー：オブジェクト指向とコンポーネントによるソフトウェア工学—UML を使って，児玉公信（監訳），ピアソン・エデュケーション (2000).
5) 大西　淳：UML におけるモデル整合性検証支援システム，電子情報通信学会論文誌，Vol.J84-D-I, No.6, システム開発特集号，pp.671–681 (2001).
6) Evans, A. and Kent, S.: Core Meta-Modeling Semantics of UML: The pUML Approach, *2nd International Conference on the Unified Modeling Language*, Colorado, Rumpe, B. and France, R.B.(Eds.), LNCS 1723 (1999).
7) 白井　明，河合正至，佐伯元司：ソフトウェアプロダクトの整合性検査システムの構築法，情報処理学会第 62 回全国大会特別トラック講演論文集，2H-02 (2001).
8) 青木利晃，片山卓也：オブジェクト指向方法論のための形式的モデル，日本ソフトウェア科学会学会誌，コンピュータソフトウェア，Vol.16, No.1, pp.12–32 (1999).
9) Glinz, M.: A Lightweight Approach to Consistency of Scenarios and Class Models, *Proc. IEEE ICRE*, pp.49–58 (2000).
10) 鷲見　毅，大西　淳：ステレオタイプによる UML モデル間の整合性検証支援，オブジェクト指向最前線 2001，pp.25–32, 近代科学社 (2001).
11) J. ランボー，M. ブラハ，W. プレメラニ，F. エディ，W. ローレンセン：オブジェクト指向方法論 OMT，羽生田栄一（監訳），株式会社トッパン (1993).
12) S. シュレイアー，S.J. メラー：オブジェクト指向システム分析上流 CASE のためのモデル化手法，本位田真一，山田　亨（訳），啓学出版 (1993).
13) S. シュレイアー，S.J. メラー：続オブジェクト指向システム分析オブジェクト・ライフサイクル，本位田真一，伊藤　潔（監訳），啓学出版 (1992).
14) 中谷多哉子，青山幹雄，佐藤啓太：ソフトウェアパターン，共立出版 (1999).
15) Object Management Group: UML 仕様書，OMG Japan SIG 翻訳委員会 UML 作業部会（訳），株式会社アスキー (2001).

(平成 13 年 10 月 9 日受付)
(平成 14 年 3 月 14 日採録)

鷲見　毅（学生会員）

1978 年生．2000 年 3 月立命館大学理工学部情報学科卒業．同年 4 月同大学大学院理工学研究科情報システム学専攻博士課程前期課程に入学．2002 年 3 月同修了．オブジェクト指向分析，設計に関する研究に従事．

渡辺　晴美（正会員）

1990 年東京工科大学工学部情報工学科卒業．1990 年から 1993 年まで日本電気マイコンテクノロジー（株）勤務．1995 年東京工科大学大学院工学研究科システム電子工学専攻修士課程修了．1998 年東京工業大学大学院情報理工学研究科計算工学専攻修了．博士（工学）．1998 年から 2000 年北陸先端科学技術大学院リサーチアソシエイト．2000 年から立命館大学理工学部情報学科助手．ソフトウェアのテスト，リアルタイムシステム開発方法論の研究に従事．日本ソフトウェア科学会会員．

大西　淳（正会員）

1957 年生．1979 年京都大学工学部情報工学科卒業．1981 年同大学院工学研究科修士課程情報工学専攻修了．1983 年同博士課程中退．1983 年京都大学助手，1989 年京都大学助教授を経て 1994 年より立命館大学教授．理工学部情報学科に勤務．京都大学工学博士．要求工学やソフトウェア開発技法に興味を持つ．電子情報通信学会，日本ソフトウェア科学会，IEEE Computer Society, ACM 各会員．