ソフトウェア工学 130－8
（2001．3．22）

UMLモデル間の整合性検証支援

鷲見 毅　　大西 淳
立命館大学大学院理工学研究科 情報システム学専攻
525-8577　滋賀県草津市野路 1-1-1
E-Mail : {sumi, ohnishi}@selab.cs.ritsumei.ac.jp

オブジェクト指向分析・設計では、UMLを用いたモデリングが主流となっている。UMLは、非常に汎用的な言語であり、分析、設計、実装までもサポートしている。そのために、UMLでは８つのモデル図を用意している。しかし、UMLに記述されている記法に基づいてモデルを作成しただけでは、各モデル間の整合性が保証されないという問題がある。本研究では、UMLモデル間の整合性検証支援として、互いに独立に作成されたUMLモデル間関連を明確にし、そこから矛盾を発見する手法を提案する。

Support of verifying the consistency between UML Models

Takeshi SUMI, Atsushi OHNISHI
Department of Computer Science, Ritsumeikan University
1-1-1 Noji-Higashi,Kusatsu,Siga 525-8577,Japan

UML is widely spread as a modeling language in object-oriented analysis and design. UML aims to support object-oriented analysis, design and programming. Therefore, UML provides eight modeling diagrams. There is a problem that the consistency between each model with UML notation isn't assured. We propose a technique to detect inconsistency between UML models by defining relations between these two models.

1 はじめに

1．1 背景

近年、オブジェクト指向という考え方は、ソフトウェア開発における主流の考え方になってきている。それに伴い、オブジェクト指向分析・設計で用いる、モデリング言語が必要となった。モデリング言語は、開発者がシステムを構築する際に、問題点や解決策を議論する上で重要な役割を果たす。その一つとして、開発者間で議論をする際の共通言語としての役割がある。ソフトウェア分析・設計が、開発者間共通のモデリング言語に基づいて行われる事は、開発者間の誤解を減らす事に繋がる。このような共通言語としてのモデリング言語は、システムの様々な側面を記述するのに十分な表現力を持っている事、広く使われている事が重要である。このような事から、非常に豊かな表現能力を持っており、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欠かせない存在となると考えられる[1][2][3][4]。

1．2 問題点

UMLモデルは、その記法に基づいて作成しただけでは、モデル間の整合性が保証されないという問題点がある。また、UMLモデルを記述する順序が明確には決まっておらず、各モデルは独立に作成される事になる。このような事から、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はかなり高い。この矛盾を持っ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モデル間で整合性検証を行う必要がある。

しかし、UMLモデル間の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未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しまっている。

1．3 本研究の目的

本研究では、UMLモデル間関連を明確化することによって、モデル間の矛盾を発見する手法を提案する。

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。しかし、UMLの記法では、モデル間関連は明確に定義されていない。そのため、モデル間関連は開発者自身が判断、解釈しなくてはならなかった。このモデル間関連の有無が明確になることによって、モデル間の矛盾を発見できるようになる。

1．4 関連研究

関連した研究として、UMLモデル間の構文的な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、意味モデルを考慮しない検証に限定されている。

UMLモデルの意味的な検証についての研究としては、pUML(precise UML)[6]やOCL(Object Constraint Language)[7]によって形式的な意味を与える手法が提案されている。形式的な意味を与えることによって、意味的な検証が可能となる。

2 手法

2．1 手法

本研究では、モデル間関連を明確にし、この情報を矛盾発見に取り入れる事を試みる。これにより、モデル間関連についての矛盾が発見できると考える。

UMLモデル間関連は、システムをモデル化する際に生じる。システム上の共通の部分をモデル化した部分については、2つのモデル間で、

その記述が互いに整合していなければ矛盾となる。

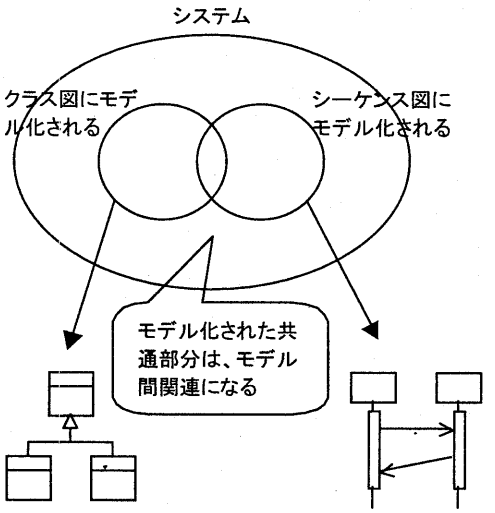

図１：モデル間関連の発生

このような矛盾が生じる理由として、UMLモデル間関連が明確でない事が考えられる。例えば、状態遷移図と相互作用図を考えた場合、状態遷移図はクラス単体の状態遷移を記述し、相互作用図は複数オブジェクトの協調動作を記述するという違いはあるが、関連をしあっている部分がある。しかし、互いに独立した部分と関連を持っている部分は明確に区別されているわけではない。その結果、状態遷移図と相互作用図が独立に作成された場合、状態遷移図に記述されたシステムの動作と、相互作用図に記述されたシステムの動作が矛盾するという事が起きる。

モデル記述とモデル間関連を比較する事で、モデル間関連を介してモデル間の矛盾発見が可能となり、整合性検証を支援できる。

また、この手法は作成されたUMLに対して適用する事を前提としている。これは、モデル間関連は、モデル化して初めて明確にできると考えるからである。

２．２ モデル間関連

本研究において、２つのモデル間にモデル間関連があるとは、モデルの構成要素同士がシステム上の同一の“もの(クラスや操作、イベント等)”を指す事を意味する。

構成要素間に関連があるならば、その要素を持つモデル間には、要素間の関連にそったモデル関連があると考えられる。すなわち、モデル間関連は構成要素の関連によって表されているといえる。図２と図３で、とシーケンス図上のメッセージMと状態遷移図上のイベントEにモデル間関連があるとする。これはシステム上、イベントEとメッセージMが同一の意味を持っていると考える。すなわち、このシステムでイベントEとは、メッセージMである事を意味する。

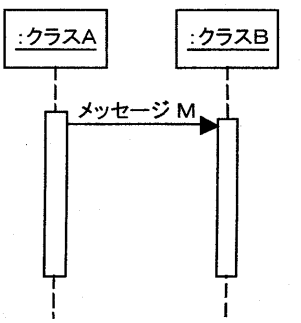

図２：シーケンス図

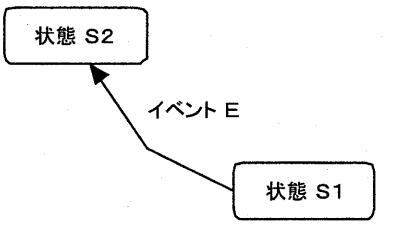

図３：状態遷移図

２．３ 矛盾

本研究で扱う矛盾とは、「UMLのモデル間関連が明確なときに、モデル間関連に対してモデ

ル記述の整合が取れていない事」である。

モデル間関連での例と同様に、図４のクラス図上の依存Rと図５のシーケンス図上のメッセージMにモデル間関連があるとする。しかし、依存Rはクラス図上でクラスAとクラスCの依存関係を示しているが、メッセージMはシーケンス図上でクラスAのオブジェクトとクラスBのオブジェクト間の相互作用として記述されている。この記述は、この２つの要素が関連を持つ事、すなわち、依存Rとメッセージが同一のものである事と矛盾している。

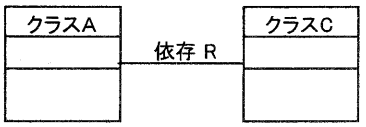

図４：クラス図

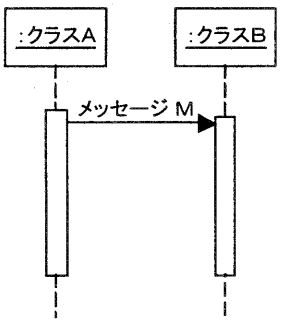

図５：シーケンス図

２．４ 関連付け

構成要素の関連付けるために、新たなステレオタイプを定義し、用いる。現段階では、ステレオタイプは単なるラベルに過ぎない。同一のステレオタイプが付けられた要素同士は関連があると判断する。

ステレオタイプを付ける構成要素は、ユーザが選択する。これは、ラベルとして用いられるステレオタイプでは、要素間の関連を見つけだす事ができないからである。これは今後、ステレオタイプを型として定義し、これを体系化していく事によって、関連を持つ可能性がある要素を自動的に見つけ出せるようにする必要がある。

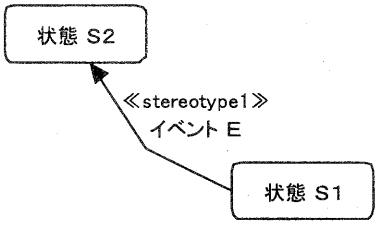

図６：ステレオタイプを付けた状態遷移図

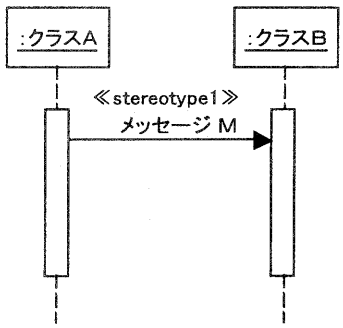

図７：ステレオタイプを付けたシーケンス図

２．５ 矛盾発見

ステレオタイプを単なるラベルとして用いた時に発見できる矛盾は、要素の抜けと、値(文字列)の間違いである。

要素の抜けは、必要な要素が抜けている場合の矛盾である。関連が付けられない要素がある場合に、要素の抜けがある可能性がある。モデル内の全ての要素が関連を持つわけではないので、関連が付けられない要素が即、抜けに繋がるとは言えない。だが、関連を持つ要素の中に、関連を持たない要素がある場合、何らかの要素が抜けている可能性が高い事は言える。

値の間違いは、ステレオタイプが付けられた要素が同一の値を持たない場合の矛盾である。

システム上同一の操作やメッセージであるならば、名称やその引数も同じでなければならない。特に、操作、メッセージの引数の間違いが考えられる。また、操作名やメッセージ名等のラベル自体の間違いも考えられる。

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。この例では「自動販売機からの缶飲料

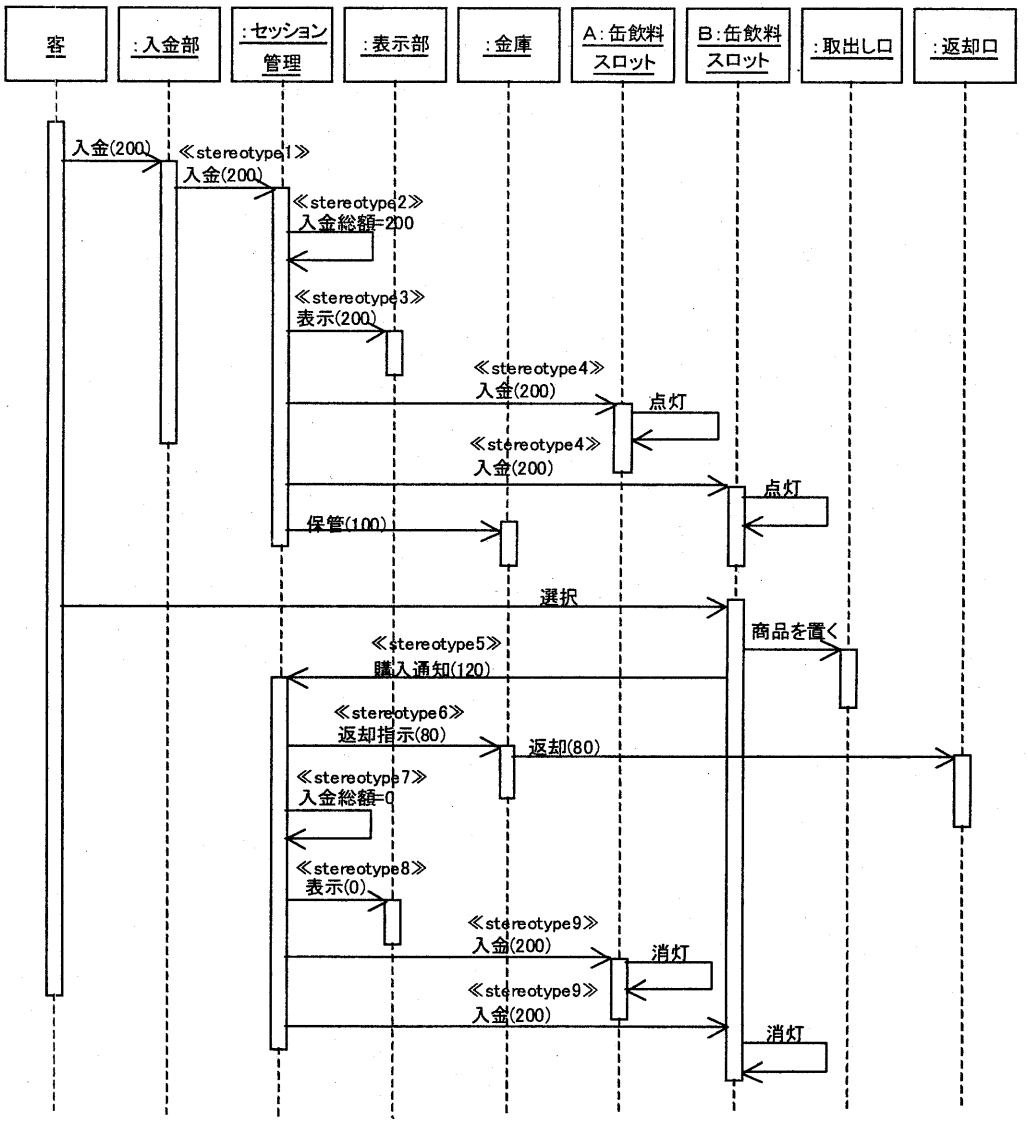

図８：缶飲料購入のシーケンス図

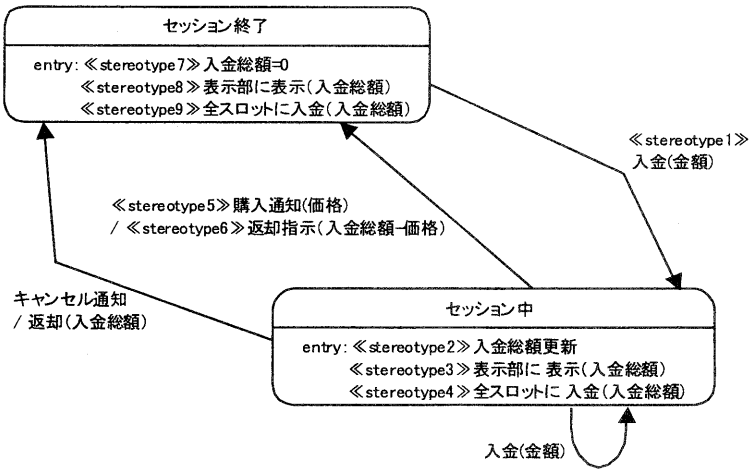

図９：セッション管理の状態遷移図

購入」についてのシーケンス図(図８)と、「セッション管理クラス」の状態遷移図(図９)の間で、手法を適用し、矛盾発見を行う。

なおこの例題では、シーケンス図と状態遷移図の間に、意図的に矛盾を記述している。

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、簡単に述べる。

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、自動販売機の処理を管理する。
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. 缶飲料スロットは、入金総額に対して購入の可否を返す。缶飲料の種類毎にオブジェクトが生成される。
6. 金庫は、入金されたお金を保管する。
7. 取出し口は、購入された缶飲料を客に渡す。
8. 返却口は、釣銭を返す。

ここでのセッションとは、缶飲料の購入に関する一連の処理を指し、セッション管理クラスは、この処理を管理するクラスである。

この例では、セッション管理クラスの初期状態は、「セッション終了」である。これは、セッションが終了した段階では、セッション終了状態のまま、次のセッション開始まで待機しているからである。

３．２ 適用結果

例題のシーケンス図と状態遷移図で、構成要素同士を関連付けるためのステレオタイプとして、≪stereotype1～5≫を定義し、要素同士を関連付けた。

これによって、以下の２点について矛盾が発見できた。

1. 保管に関して、状態遷移図で記述されていない。
2. 缶飲料購入後の、スロットへの入金通知の金額が間違っている。

以上の２点について、具体的に説明する。

まず１つめの矛盾は、≪stereotype2～4≫に

よって、メッセージと入状イベントが関連付けられている。その後、≪stereotype5≫によって購入通知のメッセージと、セッション中状態からセッション終了状態への遷移が関連付けられている。しかし、シーケンス図では、その間に「保管(100)」のラベルが付いたメッセージが記述されている。この事から、状態遷移図のセッション中状態の内部遷移として、「保管」に関してのアクションが記述されていなければならない。

次に、ステレオタイプ≪stereotype9≫によって関連付けられたメッセージとアクションにおいて、引数の値が違っている。

以上の矛盾は、ステレオタイプをラベルとして用いたときに発見する事ができる矛盾の例といえる。

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、矛盾を発見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ていくため手間が掛かるが、モデル間関連を明確化する事によって、一部の矛盾を発見できる事が確認できた。

また、ステレオタイプを要素に付加する事によってモデル間関連を明確化するため、既に作成された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事ができるため、新たな記法を学ぶ必要がない。また、UMLを従来通りに用いる事ができる。

4．2 今後の課題

現段階では、ステレオタイプは要素に付けるラベルとして扱っている。しかし、このままでは事象の一側面しか捉える事ができない。より複雑な矛盾の発見を行うには、ステレオタイプにより要素の型付けをする必要がある。これにより、要素の前後関係を捉えて矛盾発見を行う事が可能になると考える。

また、ステレオタイプによる型定義を体系化する事により、モデル間関連を持つ可能性がある要素を自動的に見つけだす事ができれば、関連付けを行うユーザの負担を軽減することができる。

さらに、ステレオタイプによる型付けが進めば、モデル変更の波及範囲を特定する事ができるのではないかと考えている。変更の波及範囲に含まれる要素を自動的に変換し、モデルを再利用する事ができればよりユーザの負担が軽減できる。これには、構成要素間の関連に加えて、モデルとドメインとの関連情報が必要になるのではないかと考えている。

参考文献

[1]グラディ・ブーチ，UMLユーザーガイド，ピアソン・エデュケーション(1999)．

[2]Grady Booch，James Rumbaugh，Ivar Jacobson，The Unified Modeling Language User Guide，ADDISON-WESLEY(1999).

[3] Sinan Si Albir, UMLクイックリファレンス，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—，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 システム特集号掲載予定(2001)

[6]Andy Evans, Stuart Kent, Core Meta-Modelling Semantics of UML:The pUML Approach,
In: 2nd International Conference on the Unified Modeling Language. Editors: B.Rumpe and R.B.France,

Colorado, LNCS 1723, 1999.
http://www.cs.york.ac.uk/puml/publications.
html
[7]Object Constraint Language Specification
http://page.freett.com/UML/draft/OCL/copy.
html