社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－SE－155 (24)
2007／3／23

定理証明技法を用いた ユースケースの追加支援システムの研究

牛尾 遼平 † 服部 哲 † 落水 浩一郎 †
† 北陸先端科学技術大学院大学 情報科学研究科

概 要

ユースケースモデリングにおいて，システムに新たなユースケースの追加を行う際，既存のユースケースを再利用できる場合がある．しかし，実システムではユースケースの数は膨大になり，ユースケース記述も複雑かつ曖昧になることが多いので，新たに追加するユースケースが既存のどのユースケースに関連しているかを把握するのは困難である．本稿の手法では，ユースケース記述の事前条件，事後条件を一階述語論理の論理式で記述する．そして，新たなユースケースと既存のユースケースの間で成り立つ論理的関係を調べ，再利用できる可能性があるユースケースを抽出する．

Research on Support System for Addition of Use Case Using Theorem Proving Technique

Ryohei Ushio † Satoshi Hattori † Koichiro Ochimizu †
† School of Information Science, Japan Advanced Institute of Science and Technology

Abstract

In use case modeling, when we add a new use case to an existing system, some existing use cases can often be reused. But in actual systems, the number of the use cases may be too large and a use case description tends to be complicated and ambiguous, so it is difficult to understand which of the existing use cases is related to a new use case. In our method, the precondition and the postdondition in a use case description are expressed by the first-order predicate logic. We search use cases which have possibility of reuse by inspecting logical relationship between a new use case and the existing use cases.

1 はじめに

近年，オブジェクト指向ソフトウェア開発において，分析や設計を行うための言語としてUMLが用いられることが多い．その要素の1つであるユースケース図とユースケース記述は，機能要求の定義に用いられる．UMLのユースケースとは，アクターとシステムの相互作用をイベント系列として表現したもので，ユースケース図の各ユースケースにおいて，相互作用の詳細をユースケース記述として補う．アクターとは，対象としているシステムと直接関係する人やシステムなどを役割として表現したものである．

システムに新たなユースケースの追加を行う際，既存のユースケースを再利用できる場合がある．例として，携帯電話アプリケーションを考える．既存の携帯電話アプリケーションに新たなユースケース「写真付きメールを送信する」を追加するとしよう．この際，ユースケース「メールを送信する」が，携帯電話アプリケーションに既に存在すれば，このユースケースを新たに追加するユースケース「写真付きメールを送信する」のシナリオの一部として再利用可能である．この例のような，あるユースケースの一部に他のユースケースが用いられるという関係をユースケースモデリングでは，include（包含）関係という．しかし，実システムにおいて，ユースケースの数は膨大になり，ユースケース記述も複雑かつ曖昧になりがちであるので，新たに追加するユース

ケースが既存のどのユースケースに関連しているかが把握しにくいという問題がある．

本研究の目的は，システムに新たなユースケースを追加する際，関連する既存のユースケースを把握できる機構を構築することである．本研究では，include（包含）関係で包含できる可能性があるユースケースを抽出することにより新たなユースケースの追加を支援するようなシステムを開発する．新たなユースケースの追加といっても色々なものが考えられるが，本研究では既存のユースケースを変更せずに，新たなユースケースの一部として再利用できる場合を対象とする．

本研究のアプローチでは，ユースケース記述の事前条件，事後条件を一階述語論理の論理式で記述する．そして，新たなユースケースと既存のユースケースの間で成り立つ論理的関係を定理証明技法を用いて調べ，再利用できる可能性があるユースケースを抽出する．本研究では，再利用できる可能性があるユースケースを，定理証明技法を用いて抽出するアルゴリズムを提案し，それを実現するツールを開発した．

適用例として，エレベータ制御システムの事例研究 [1] を対象に，新たなユースケースを追加し，再利用できる可能性がある既存のユースケースの抽出を試みた．その結果，抽出に成功し本稿の手法の有効性を確認することができた．

2 本研究のアプローチ

本研究のアプローチでは，ユースケース記述の事前条件，事後条件を一階述語論理の論理式で記述する．そして，論理式の間の準含意（≫）関係を定義し，include（包含）関係によって包含できる可能性のあるユースケースを，準含意関係を調べることで抽出する．

2.1 include（包含）関係

あるユースケース A と別のユースケース B の一部に同様のシナリオがある場合に，その部分をくくり出して，独立にユースケース C とすることができる．この場合，ユースケース A とユースケース C，ユースケース B とユースケース C の間には包含の関係が成り立つ．包含の関係では，包含するユースケースから包含されるユースケースに対して，矢印付きの破線とステレオタイプ ≪include≫ でその関係を表す [2]．本稿では，包含するユースケースを基底ユースケース，包含されるユースケースをサブユースケースと呼ぶ．

エレベータ制御システム [1] を例に挙げると，ユースケース「エレベータを要求する」，ユースケース「行き先を選ぶ」は両方とも「エレベータを派遣する」，「エレベータをフロアに止める」というシナリオを含む．すなわち，これらのユースケースから，「エレベータを派遣する」，「エレベータをフロアに止める」のシナリオをサブユースケースとしてくくり出すことができる．そして，基底ユースケース「エレベータを要求する」，基底ユースケース「行き先を選ぶ」からサブユースケース「エレベータを派遣する」，サブユースケース「エレベータをフロアに止める」に対して，包含の関係を表すことができる．以下の図 1 に，エレベータ制御システム [1] のユースケース図を示す．

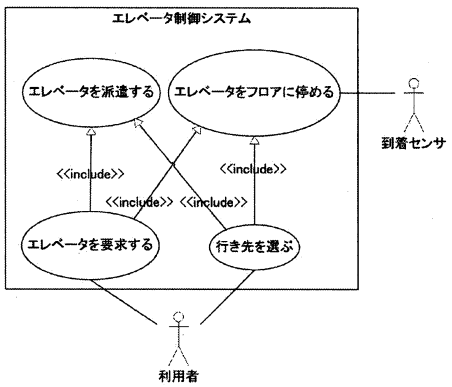

図 1 エレベータ制御システムのユースケース図

2.2 事前条件，事後条件の一階述語論理表現

UML では，ユースケース記述に事前条件 (precondition) と事後条件 (postcondition) を記述する．事前条件とはユースケースを実行する前にシステムが満たすべき条件で，事後条件とはユースケースを実行した後にシステムが満たすべき条件であ

る．本研究では，ユースケース記述の事前条件，事後条件を一階述語論理の論理式で表現する．一階述語論理とは，変数である引数とそれらに対する量化記号のみを許し，述語が定数である述語論理のことである [3].

ユースケース記述の事前条件，事後条件を一階述語論理の論理式で表現するというアプローチは文献 [4] を参考にした．[4] では，ユースケースの代替シナリオ，別シナリオを統合して状態遷移図を記述し，テストケースを生成するのに用いている．代替シナリオとは，シナリオの途中で条件分岐が生じた場合や，ユースケースが達成できなかった場合のシナリオのことである．

自然言語で記述されたユースケースにおいて，事前条件，事後条件の表現は曖昧で，通常は一文程度で単純に記述されることが多いが，実際には，ハードウェアに関する制約など様々な暗黙の条件が含まれる．本研究では，それらの暗黙の条件も視野に入れて，事前条件，事後条件を設定し論理式で表す．事前条件，事後条件の論理式による表現と数理論理学の含意記号を用いて，「事前条件 → 事後条件」と表すことができる．なぜなら，ユースケースにおいて，事前条件が成立していると仮定すると，シナリオを達成した後，事後条件が成り立つからである．ここで，シナリオを介して条件が変化することから，ユースケースによって，システムのある状態がシステムの別の状態に推移すると見ることができる．本稿の図では，「事前条件 → 事後条件」の関係を図 2 のように表す．

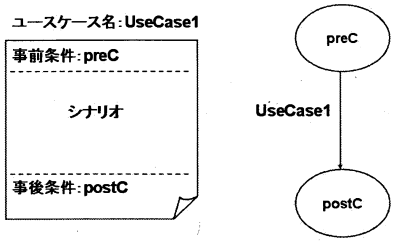

図 2　本稿の図で用いる「事前条件 → 事後条件」の関係

2.3　準含意関係

本研究では，論理式の間の準含意（≫）関係を以下のとおり定義する．

$$X \gg Y$$

$X = A1 \vee A2 \vee \cdots \vee An$　のとき，
$(A1 \rightarrow Y) \vee (A2 \rightarrow Y) \vee \cdots \vee (An \rightarrow Y)$ が成り立つ．

準含意関係が成立してもサブユースケースを基底ユースケースの任意の箇所に包含できるわけではない．包含できるのは図 3 に示すように，シナリオの最初または最後のみである．

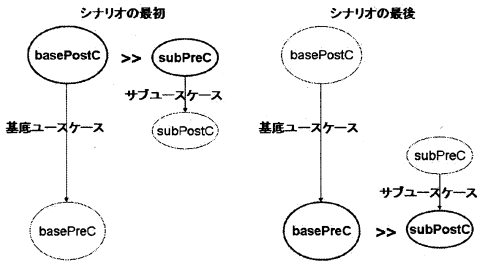

図 3　サブユースケースを包含できる箇所

ここでは，Design by Contract に基づくモジュラー検証のように，プログラムの正しさを確かめているのではなくて，単なる包含可能性を調べているだけである．ここでは，基底ユースケースの事前条件 ≫ サブユースケースの事前条件，基底ユースケースの事後条件 ≫ サブユースケースの事後条件が成立するかどうかを調べている．

勿論，シナリオの途中に包含できないわけではないが，途中に包含する場合は図 4 に示すように，包含する箇所における，シナリオの一部の事後条件を設定する必要がある．

5.3 でも述べるが，本研究で定義する準含意関係が成立してもサブユースケースを基底ユースケースに必ず include（包含）できるとは限らない．すなわち，準含意関係は基底ユースケースがサブユースケースと include（包含）関係を持つための必要条件であるが，必要十分条件ではない．

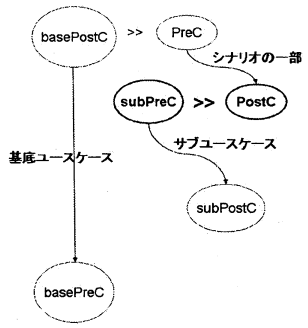

図 4　サブユースケースを途中に包含する場合

3　サブユースケース抽出アルゴリズム

3.1　アルゴリズムの概要

2.3 で述べた，準含意（≫）関係に基づき，基底ユースケースに包含できる可能性があるサブユースケースを抽出するアルゴリズムを提案する．既存のシステムに新たなユースケースを追加する際，そのユースケースを基底ユースケースとして，包含できる可能性があるサブユースケースを，このアルゴリズムにより抽出できる．初めに，アルゴリズムの入力と出力について述べる．入力は新たに追加するユースケースの事前条件，事後条件を一階述語論理の論理式で表したものである．出力は新たに追加するユースケースが包含できる可能性がある既存のサブユースケースの組み合わせである．なお，可能な限り全ての候補を出力する．図 5 にアルゴリズムの概要を示す．

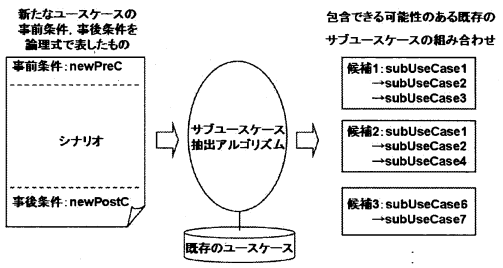

図 5　アルゴリズムの概要

3.2　アルゴリズムの詳細

サブユースケース抽出アルゴリズムは，事前条件からの抽出と事後条件からの抽出の両方を行う．どちらも，深さ優先探索により，以下の各項で説明する一つ目のサブユースケース抽出処理，二つ目以降のサブユースケース抽出処理を順番に行っていき，それぞれの処理において準含意関係が成立するか調べていく．そして，可能なサブユースケースの組み合わせを全て候補として保持し，最後に全候補を出力する．以下で，事前条件からの抽出における，アルゴリズムの一つ目のサブユースケース抽出処理，二つ目以降のサブユースケース抽出処理について説明する．

3.2.1　一つ目のサブユースケース抽出処理

新たなユースケースの事前条件の論理式 X について，$X \gg Y$ を満たす論理式 Y を事前条件として持つ既存のユースケースを抽出する（図 6）．抽出に成功すれば，そのユースケースを候補として保持する．

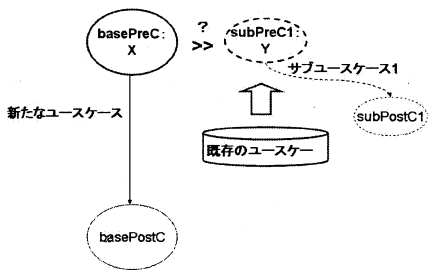

図 6　一つ目のサブユースケース抽出処理

3.2.2　二つ目以降のサブユースケース抽出処理

3.2.1 で抽出したユースケースの事後条件の論理式を Y' としたとき，$X' \gg Y'$ を満たす論理式 X' を事前条件として持つ既存のユースケースを抽出する（図 7）．

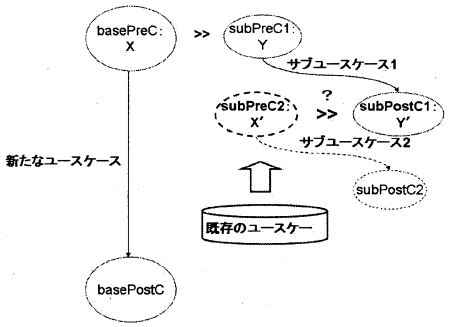

図 7　二つ目以降のサブユースケース抽出処理

以降は，抽出したユースケースに対して，同様に準含意関係による既存のユースケースの抽出を行っていき，ユースケースを抽出できなくなるまで続ける．抽出に成功する毎に，それまでに抽出したユースケースの組み合わせを候補として保持する．

3.2.3　事後条件からの抽出

事後条件からの抽出については，一つ目のサブユースケース抽出処理で新たなユースケースの事後条件の論理式を X としたとき，$X \gg Y$ を満たす論理式 Y を事後条件として持つ既存のユースケースを抽出する（図 8）．

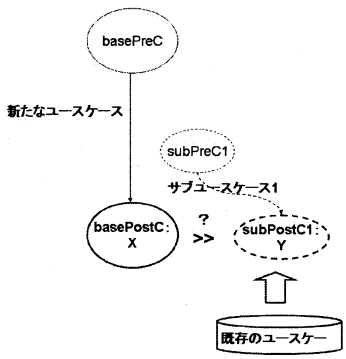

図 8　事後条件からの抽出における，一つ目のサブユースケース抽出処理

二つ目以降のサブユースケース抽出処理では，まず一つ目のサブユースケース抽出処理で抽出したユースケースの事前条件の論理式を X' としたとき，$X' \gg Y'$ を満たす論理式 Y' を事後条件として持つ既存のユースケースを抽出する（図 9）．

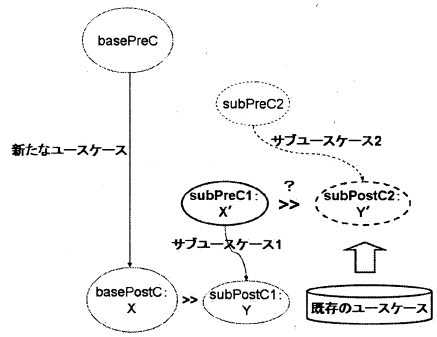

図 9　事後条件からの抽出における，二つ目以降のサブユースケース抽出処理

以降は，抽出したユースケースに対して，同様に準含意関係による既存のユースケースの抽出を行っていき，ユースケースを抽出できなくなるまで続ける．抽出に成功する毎に，それまでに抽出したユースケースの組み合わせを候補として保持する．

4　ツールの開発

3 で述べたサブユースケース抽出アルゴリズムを実現するツールを論理型言語 prolog で開発した．prolog の処理系は SWI-Prolog[5] を用いた．prolog で開発した理由は，3 で述べた準含意関係を調べる処理で，論理式を扱うため，論理型言語を用いるのが適切だと考えたからである．また，サブユースケースを抽出する際に，再帰処理が必要となるので，これに関しても論理型言語が適していると考えた．このツールを用いてサブユースケースの抽出を試みる際，既存のユースケースの情報（ユースケース名，事前条件，事後条件）をあらかじめ規則としてソースプログラム内に記述しておく必要がある．

5　適用例

本研究で提案したサブユースケース抽出アルゴリズムをエレベータ制御システムの事例研究 [1] を題材に適用を試みた．

5.1 適用内容

初めに，エレベータ制御システムの全てのユースケースに対して，事前条件，事後条件を一階述語論理の論理式で記述し，4 のツールのソースプログラム内に記述した．次に，エレベータ制御システムに新たなユースケースとして，A 階から B 階へは停まらず，B 階から各階に停まる高速エレベータに関するユースケース「高速エレベータを要求する」「高速エレベータで行き先を選択する」を追加した．そして，それらの事前条件，事後条件を一階述語論理の論理式で表し（記述例を付録 A に示す），4 のツールに入力として与え，サブユースケースの抽出を試みた．図 10 に，新たなユースケース追加後のユースケース図を示す．

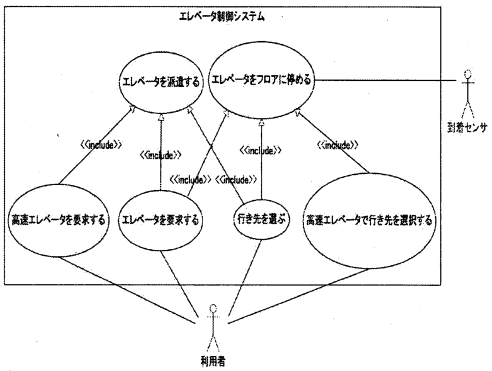

図 10　新たなユースケース追加後のエレベータ制御システムのユースケース図

5.2 適用結果

ツールの出力結果より，一部抜粋したものを表 1 ～表 4 に示す．図 10 で新たなユースケースが既存のユースケースと include（包含）関係にあるように，本適用では新たなユースケースのシナリオを設計した．つまり，あらかじめ包含可能なサブユースケースを把握していたので，ツールの出力結果よりサブユースケースの組み合わせが適切か不適切かを判断することができた．なお，表中の「上昇フロアボタンを押す」，「下降フロアボタンを押す」，「エレベータボタンを押す」，「エレベータから降りる」は新たなユースケースのシナリオの一部である．

表 1　新たなユースケース「高速エレベータを要求する」の事前条件からの抽出による出力結果

		順番1	順番2	順番3	順番4
適切	候補1	上昇フロアボタンを押す	エレベータを派遣する	エレベータをフロアに停める	
	候補2	下降フロアボタンを押す	エレベータを派遣する	エレベータをフロアに停める	
不適切	候補1	上昇フロアボタンを押す	エレベータをフロアに停める		
	候補2	下降フロアボタンを押す	エレベータをフロアに停める		
	候補3	上昇フロアボタンを押す	エレベータを派遣する	エレベータをフロアに停める	エレベータから降りる
	候補4	下降フロアボタンを押す	エレベータを派遣する	エレベータをフロアに停める	エレベータから降りる

表 2　新たなユースケース「高速エレベータを要求する」の事後条件からの抽出による出力結果

		順番1	順番2	順番3
適切	候補1	上昇フロアボタンを押す	エレベータを派遣する	エレベータをフロアに停める
	候補2	下降フロアボタンを押す	エレベータを派遣する	エレベータをフロアに停める
不適切	候補1	上昇フロアボタンを押す	エレベータをフロアに停める	
	候補2	下降フロアボタンを押す	エレベータをフロアに停める	
	候補3	エレベータボタンを押す	エレベータを派遣する	エレベータをフロアに停める

表 3　新たなユースケース「高速エレベータで行き先を選択する」の事前条件からの抽出による出力結果

		順番1	順番2	順番3	順番4
適切	候補1	エレベータボタンを押す	エレベータを派遣する	エレベータをフロアに停める	エレベータから降りる
不適切	候補1	エレベータボタンを押す	エレベータをフロアに停める	エレベータから降りる	

表 4　新たなユースケース「高速エレベータで行き先を選択する」の事後条件からの抽出による出力結果

		順番1	順番2	順番3	順番4
適切	候補1	エレベータボタンを押す	エレベータを派遣する	エレベータをフロアに停める	エレベータから降りる
不適切	候補1	エレベータボタンを押す	エレベータをフロアに停める	エレベータから降りる	
	候補2	エレベータボタンを押す	エレベータを派遣する	エレベータをフロアに停める	
	候補3	上昇フロアボタンを押す	エレベータを派遣する	エレベータをフロアに停める	エレベータから降りる
	候補4	下降フロアボタンを押す	エレベータを派遣する	エレベータをフロアに停める	エレベータから降りる

5.3 考察

ツールの適用の結果，あらかじめ把握していた包含可能なサブユースケースを抽出できた．すなわち，本研究で提案した手法で，新たなユースケースに対して包含できる可能性のあるサブユースケース

を抽出できることを確認できた．不適切な候補が抽出されたことに関して，表1〜表4よりそれらの候補を分析すると，以下のことが分かった．

- 適切な候補に比べて，途中の順番で出力されるはずのユースケースが抽出されていない．
- 適切な候補に比べて，余分なユースケースが抽出されているか，途中から別のユースケースが抽出されている．

前者については，途中に出力されるべきサブユースケースをサブユースケースNとすると，基底ユースケースの事前条件 ≫ サブユースケース N+1 の事前条件（図11），基底ユースケースの事後条件 ≫ サブユースケース N-1 の事後条件（図12），サブユースケース N+1 の事前条件 ≫ サブユースケース N-1 の事後条件（図13）のいずれかが成り立ったと考えられる．

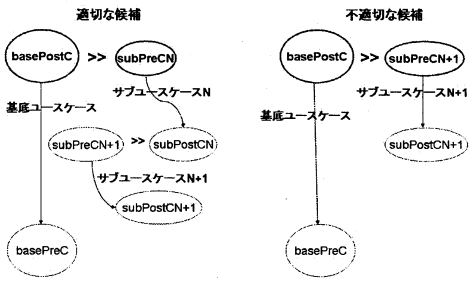

図11　途中に出力されるべきサブユースケースNが出力されなかった例1

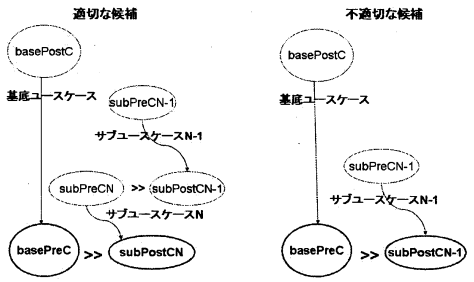

図12　途中に出力されるべきサブユースケースNが出力されなかった例2

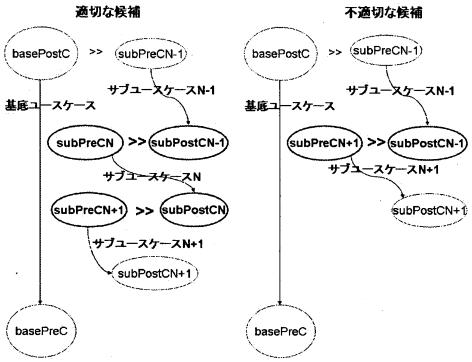

図13　途中に出力されるべきサブユースケースNが出力されなかった例3

後者については，3のサブユースケース抽出アルゴリズムで，可能な限りのサブユースケースの組み合わせを出力した結果と考える．

前者，後者いずれにおいても，3のサブユースケース抽出アルゴリズムでは準含意関係を調べるので，不適切な候補が出力されるのは理論上避けられない問題である．今回の適用では，あらかじめ適切な候補を予想できた．しかし，一般的には既存のシステムに新たなユースケースを追加する際，関連する既存のユースケースを予想することはできない．よって，利用者が事前条件からの抽出による出力結果と事後条件からの抽出による出力結果を基に，出力された候補が適切か不適切かを検討する必要がある．

6　おわりに

本研究では新たなユースケースの追加を行う方式を，利用者が既存のユースケースを再利用しつつ，定理証明システムを適用するものとして開発した．これを行うにあたり，ユースケースの事前条件，事後条件を一階述語論理の論理式で表現したものから，基底ユースケースとサブユースケースの論理式の間で成立すべき論理的関係（準含意関係）を定義した．この関係を用いて，新たなユースケースに包含できる可能性があるサブユースケースを抽出するアルゴリズムを提案し，そのアルゴリズムを論理型言語で実現した．

本研究で開発したツールは，新たなユースケースが既存のユースケースの組み合わせで実現できる場合，再利用可能なユースケースを抽出できるが，既存のユースケースの一部を変更して使用する場合には対応できない．よって，既存のユースケースの変更に伴う事前条件，事後条件の変更が他のユースケースにどのような影響を及ぼすかが調べられるようなシステムも実現したい．図 14 に既存のユースケースの事後条件を変更した場合の例を示す．

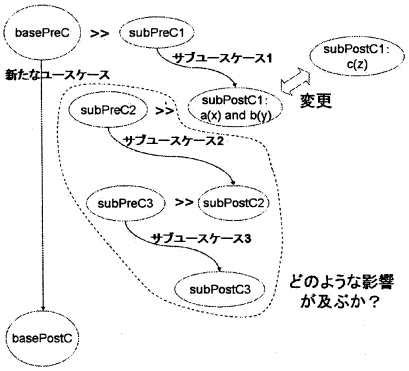

図 14　既存のユースケースの一部の変更

参考文献

[1] H.Gomma, Designing Concurrent, Distributed, and Real-time Application with UML, Addison-Wesley, 2000.

[2] 竹政昭利, はじめて学ぶ UML, ナツメ社, 2003.

[3] 小野寛晰, 情報科学における論理, 日本評論社, 1994.

[4] C.Nebut, F.Fleurey, Y.Le Traon, J.-M.Jezequel, " Automatic Test Generation : A Use Case Driven Approach" , IEEE TRANSACTIONS ON SOFTWARE ENGENEERING, NO.3, 2006.

[5] SWI-Prolog's Home, http://www.swi-prolog.org/

[6] Project Triskell, http://www.irisa.fr/triskell/Softwares/protos/ucts/

[7] A.Cockburn, ユースケース実践ガイド, 山岸耕二, 矢崎博英, 水谷雅宏, 篠原明子訳, 翔泳社, 2001.

[8] L.Sterling, E.Shapiro, 松田利夫訳, Prolog の技芸, 共立出版, 1988.

付録 A　ユースケースの一階述語論理による表現

ユースケース「高速エレベータで行き先を選択する」

pre	beInTheElevator(user, elevator) and needToGo(user, userDestinationNumber) and fastElevator(elevator) and not haveProhibitionFloorNumber(elevator, userDestinationNumber)
post	fastElevator(elevator) and not haveProhibitionFloorNumber(elevator, userDestinationNumber) and arrived(elevator, userDestinationNumber) and doorOpened(elevator, userDestinationNumber) and arrivalSensorDetectedApproach(elevator, userDestinationNumber) and motorOff(elevator) and beOnTheFloor(user, userDestinationNumber) and not beInTheElevator(user, elevator) and not needToGo(user, userDestinationNumber) and not move(elevator, userFloorNumber, userDestinationNumber) and not arrived(elevator, userFloorNumber) and not doorOpened(elevator, userFloorNumber) and not arrivalSensorDetectedApproach (elevator, userFloorNumber)