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 (510014)

北陸先端科学技術大学院大学 情報科学研究科

2007 年 2 月 8 日

キーワード: ソフトウェア工学, UML, ユースケース, 定理証明, 一階述語論理, Prolog.

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．

2　本研究のアプローチ

UML では，ユースケース記述にユースケースの事前条件，事後条件を記述する．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．自然言語で記述されたユースケース記述において，事前条件，事後条件の表現は曖昧で，通常は一文で単純に記述されることが多い．しかし，実際にはハードウェアの条件などさまざまな暗黙の条件が含まれる．本研究では，これらの暗黙の条件も含めて，事前条件，事後条件を一階述語論理の論理式で形式的に表現する．本稿では，ユースケースの include（包含）関係によって，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．本研究では，論理式で記述した事前条件，事後条件において，論理的関係を定義する．そして，この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．

3　ユースケースから一階述語論理への変換

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へ変換する方法を提案する．述語を表現する際の引数は，クラス図とユースケース図より洗い出す．そして，引数の状態，役割を述語名で表現する．そして，満たすべき各条件を述語の組み合わせで表し，最後に各条件を論理積で結び事前条件，事後条件を表現する．

4　サブユースケース抽出アルゴリズム

論理式で記述した事前条件，事後条件における論理的な関係を定理証明技法で調べることにより，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．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．

5　ツールの開発

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 で開発した理由は，アルゴリズムの処理の中で，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．

6　適用例

本研究で提案するサブユースケース抽出アルゴリズムを Gomma のエレベータ制御システムの事例研究を題材に適用を試みた．

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た．次に，エレベータ制御システムに新たなユースケースを追加し，それらの事前条件，事後条件を一階述語論理の論理式で記述した．そして，開発したツールを用いてサブユースケースの抽出を試みた．

本適用では，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，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，ツールの出力結果について，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．適用の結果，適切なサブユースケースの組み合わせを抽出できた．しかし，同時に不適切なサブユースケースの組み合わせも抽出されることがわかった．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．そこで，利用者が出力結果を基に，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．

7　今後の課題

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．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．