2003－SE－142 (3)
2003／5／30

Javaアプリケーションの要求仕様とセキュリティポリシィのトレードオフ分析

海谷治彦† 佐々木宏太† 海尻賢二†

モバイルコードを利用したJavaアプリケーションは，実行する環境下でのセキュリティポリシィによって，その機能を制約される．本稿では，Javaモバイルコードのためのセキュリティポリシィと，アプリケーションの要求仕様とのトレードオフを分析する手法を提案する．我々はすでに，あるセキュリティポリシィ下において，Javaモバイルコードのどのような機能が実行可能か否かを確かめるツールを開発している[1)]．このツールを用いて，あるポリシィ下において，どの機能が実行可能で，どの機能が不可能かを明確にすることができる．我々はゴール指向要求分析手法を用いて，アプリケーションの要求仕様を充足するためには，どのような機能が必要かを導く．ポリシィから得られた実行可能な機能と，要求仕様から得られた実行したい機能を比較することで，ポリシィと要求の衝突を分析したり，要求の曖昧さを明確にしたりすることが可能となる．我々の手法では，衝突や曖昧さを解消するために，ポリシィを変更指針を示す．しかし，ポリシィが変更不可能である場合には，要求の一部を取り下げを勧告する．

Trade-off Analysis between Security Policies for Java Mobile Codes and Requirements for Java Application

Haruhiko Kaiya ,† Kouta Sasaki † and Kenji Kaijiri†

We propose a method for analyzing trade-off between security policies for Java mobile codes and requirements for Java application. We assume that mobile codes are downloaded from different sites, they are used in an application on a site, and their functions are restricted by security policies on the site. We clarify which functions can be performed under the policies on the site using our tool. We also clarify which functions are needed so as to meet the requirements for the application by goal oriented requirements analysis(GORA). By comparing functions derived from the policies and functions from the requirements, we can find conflicts between the policies and the requirements, and also find vagueness of the requirements. By using our tool and GORA again, we can decide which policies should be modified so as to meet the requirements. We can also decide which requirements should be abandoned so as to meet policies which can not be changed.

1. はじめに

Javaアプリケーションは，複数の異なる提供者にが提供するモバイルコードを同時に利用することが可能である．よって，我々は多様な機能を容易かつ効率的に，アプリケーションに組み込むことが可能となる．アプリケーションが実行されるマシンには，ファイルシステムやネットワークコネクション等の重要な資源があるため，それら資源をモバイルコードの悪意ある，もしくは不適切な動作から保護しなければならない．Javaの場合，モバイルコードの振る舞いを制限するためのセキュリティポリシィによって，資源の保護が行われている．

しかし，セキュリティポリシィによって振る舞いを制限されたアプリケーションが，そのアプリケーションの本来の要求仕様に合致した振る舞いをしているか否かを確認することは容易ではない．特に，複数のサイトからダウンロードされたモバイルコードを利用したアプリケーションに関しては，その確認が，より困難だと思われる．ここで簡単な例題を考えてみる．二つの異なる場所からダウンロードされたモバイルコードAとBを利用するアプリケーションを考える．そして，Aの中の必要な機能を実行できるためには，BのためのポリシィP_Bより制限の少ないセキュリティポリシィP_Aが必要であるとする．もし，P_Aを，そのアプリケーションのポリシィとして採用した場合，Bに含まれる悪意ある，もしくは不適切な機能が実行される危険がある．しかし，P_Bを採用したのでは，Aに含まれる必要な機能を実行することができなくなる．ここでの例は単純なため，AにはP_Aを，BにはP_B

† 信州大学 工学部 情報工学科
Department of Information Engineering
Faculty of Engineering, Shinshu University
kaiya@acm.org http://www.cs.shinshu-u.ac.jp/~kaiya/

をポリシィとして適用すれば済むことは容易にわかる．

しかし，単純に異なるコードのための異なるポリシーを両立して利用できない状況も場合によっては考えられる．引き続き，この単純な例について考える．P_A は，本質的に B の悪意ある，もしくは不適切な機能の実行を許してしまうと仮定しよう．B の不適切な機能の実行を回避するためには，ポリシィ P_A を修正する (制限を多くする) だけでなく，コード A に期待していた機能の一部を諦めなければならない．なぜなら，修正されたポリシィ P_A 下では，実行することが許されない機能が発生してしまうためである．

モバイルコード自身，そのネットワーク上の配置，そして，それらを実行する際に適用されるセキュリティポリシィを本稿では環境と呼ぶことにする．そのような環境はアプリケーションの要求とゴールを明確にするのに役立つ場合がある．例えば「発生してほしくない事」を示す否定的な要求仕様を認識するのは，してほしい事を認識するよりも困難である．結果として，否定的な要求仕様が欠落し，要求仕様書が曖昧になる場合がある．ある環境下で，起こりうることと，起こってほしいことを比較することで，そのような曖昧さの一部を解消することが可能であると思われる．

ここまでの議論から，セキュリティポリシィを含む環境と，アプリケーションの要求仕様の妥協点を探るための手法が必要であると思われる．特に，どのような点をどのように妥協したかを明確にすることが，後の環境変化への対応を鑑みると重要となる．前述のように，そのような妥協点を模索する過程において，曖昧な要求を明確化する機会を得ることもできる．本稿で提案する手法は，このような目的に利用できる．我々の手法では，大まかに言って，以下のような流れで妥協点を探る．

(1) 要求仕様で必要とされる機能と，コード，コードの配置そしてセキュリティポリシィによって実行が許される機能をそれぞれに明確にする．
(2) それら二種類の機能の違いを明確にする．
(3) ポリシィを修正したり，要求を取り下げたりしつつ，それらの対立点を解消する．
(4) 比較を通して明らかになった不明確な要求を補完する．

我々はゴール指向要求分析法[2] を要求仕様の分析に用いる．Java モバイルコードおよびそれらのネットワーク上の配置，そしてセキュリティポリシィによってどのような機能が実行可能か否かを分析するために，我々が開発した “セキュリティポリシィジェネレーター&チェッカー”[1] を利用する．

続く 2 節では，本稿で扱うモバイルコードを用いたアプリケーションとは何かについて説明する．3 節では，モバイルコードアプリケーションの要求とゴールの関係について議論する．4 節において，ポリシィと要求仕様のトレードオフ分析法を紹介し，5 節の例題を通して，手法の詳細と有効性を示す．最後にまとめと今後の課題について述べる．

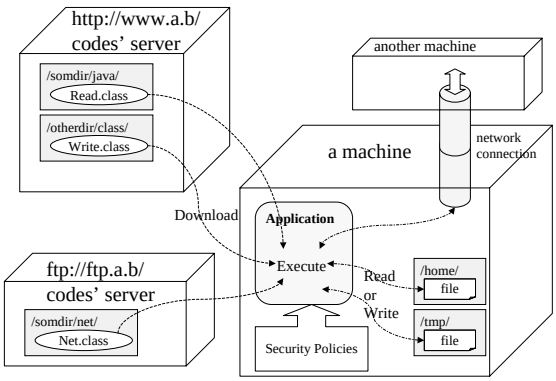

図 1 モバイルコードアプリケーションとその環境

2. モバイルコードを用いたアプリケーション

サーベイ論文[3] でも述べられているように，“モバイルコード” という用語は多様な意味で利用される．本稿では，図 1 に示すような，単純なモバイルコードアプリケーションを想定する．アプリケーションはあるマシンで実行され，そのアプリケーションはいくつかのサイトからモバイルコードをダウンロードし，利用する．本稿では，ダウンロードされるコードと，そのダウンロード元の対を，配置と呼ぶ．アプリケーションがファイルやネットワーク接続等の重要な資源を処理する際に，モバイルコードがその取り扱いをする場合がある．

他のマシンからダウンロードしたモバイルコードを利用することは，そのコード作成者が直接に自分のマシンを操作しているのとほとんど同じである．そこで，モバイルコードの振る舞いを制限するための仕組みが必要となる．Java では，そのような制限のためのルールをセキュリティポリシィとして記述し，各アプリケーションの実行時に与える．

モバイルコードは危険な面がある反面，いくつかの利点もある．第一に，モバイルコードを利用することで，柔軟に機能を合成し，新しいサービスを容易に作成することができる．第二に，モバイルコードによって，アプリケーションの振る舞いや機能を，実行時でさえ変更することができる．第三に，CGI や ASP などのサーバーサイド処理技術に比べ，モバイルコードを利用したアプリケーションは，処理が集中しないためスケーラブルである．よって，この分野における要求工学についての研究は重要かつ有益である．

前述のように，モバイルコードの特性や機能，それらの配置，そしてアプリケーションのためのセキュリティポリシィを，本稿では環境と呼ぶことにする．ま

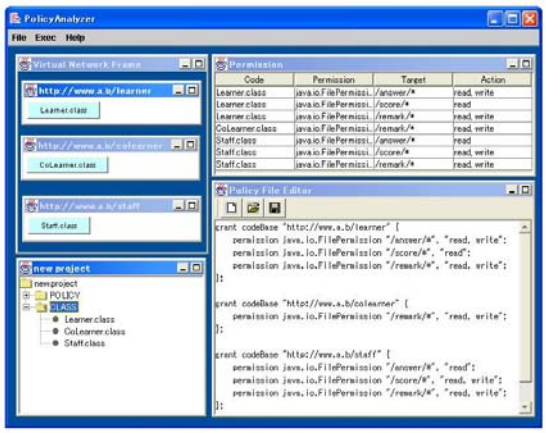

図 2 ツールの概観

た，本稿では，セキュリティポリシィはアプリケーションのユーザーの責任において変更が可能であるが，それ以外の環境は，変更が困難であると仮定する．これは，あるモバイルコードやその配置が，他のアプリケーションやシステムによっても利用されている可能性があるからである．

ここで本稿で紹介する分析法で利用するツール “セキュリティポリシィチェッカー&ジェネレータ” を簡単に紹介する．図 2 にツールの概観を示す．本ツールは，ツール内に仮想的なコード配置を記述し，別途記述されたセキュリティポリシィとの関係を分析するツールである．本ツールは主に以下の二つの機能を提供する．

- あるセキュリティポリシィ下において，各コードが実行可能か否かを判定する．
- 配置されたコード全てが実行可能であるようなセキュリティポリシィを自動生成する．

現バージョンのこのツールでは，各コードに必要なパーミッション，ターゲットそしてアクションを半自動的に取り出すことが可能となっている．また，Jode デコンパイラ[4] を利用することで，ソースコードが無い場合でも，パーミッション等の取り出しが可能である．

3. モバイルコードアプリケーションの要求とゴール

モバイルコードは，e コマースや，e ラーニング，ネットワークゲーム等のアプリケーション分野に有益であると思われる．それらの共通点は以下の通りである．第一に，要求が動的に変化するため，提供する機能も動的に変化する必要がある．第二に，スケーラビリティが必要である．そして，第三に，セキュリティに関する問題が重要となってくる．本節では，そのようなアプリケーション分野における要求とゴールについて議論する．

3.1 モバイルコードアプリケーションの要求仕様書

IEEE の要求仕様書標準[5] にはソフトウェア要求仕様書 (SRS) について以下のように述べられている．

> ソフトウェア要求仕様書は，ある環境下において，ある機能を実行するソフトウェア製品，プログラムもしくはプログラムの集合のための仕様書である．

モバイルコードアプリケーションに関する SRS の場合，“ある環境” とは，前節で述べたモバイルコードの特性や機能，その配置そしてセキュリティポリシィと考えてよい．このような環境は，モバイルコードがあるアプリケーション内で，どのような機能を実行することが可能であるかを規定している．本稿では，与えられた環境下で実行することが可能なモバイルコードの機能の集合を，“可能機能 (enabled functions)” と呼ぶことにする．

3.2 ゴールと要求仕様

ソフトウェア要求仕様は，アプリケーションユーザーのゴールから導出されるものと我々は考える．しかし，一般には，アプリケーションに要求されるゴール全てが実現できるわけではない．我々は，アプリケーションが実現することと，ユーザーがゴールとしていることを区別するために，前者を “要求 (仕様)” と呼び，後者を “ゴール” と呼ぶ．本稿でのゴールは文献[6] でいうところの “あるべき” もしくは “理想的な” ゴールを指す．一方，本稿での要求は，ゴールを達成するための実現可能な機能の記述を指す．本稿ではゴールを達成するための機能の集合を “要求機能 (required functions)” と呼ぶことにする．一般に，ゴールと要求仕様は一致しない場合が多い．我々はこの不一致をありのままに記録し，将来における環境やゴール自身の変化に応じて要求仕様を変更する支援を行う．

モバイルコードアプリケーションにおいてセキュリティに関する問題は重要であるため，それらを明示的に扱う必要がある．セキュリティに関する一般的な問題分類をまとめた NFR 型[7] は，このような問題をトップダウンに認識する際に役立つ．我々の手法では，可能機能と要求機能の違いを認識することで，ボトムアップ的にセキュリティ問題を認識してゆく．ゴール分解の形式的な推論規則[8] は，ボトムアップ，トップダウン双方からセキュリティに関するゴール分解を行う際に役立つ．

4. ポリシィと要求仕様のトレードオフ分析法

図 3 に，本手法で利用する記述もしくは概念とそれらの関係を示す．この図を用いて，ポリシィと要求仕様のトレードオフ手法の手順を説明する．

本方法の主たる入力は，環境の記述と，ゴール階層である．環境は，前述のようにモバイルコードの特性，コードの配置そしてアプリケーションを実行する際の

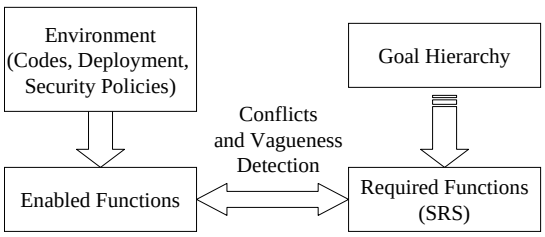

図 3 本手法で利用する記述もしくは概念とそれらの関係

セキュリティポリシィから成る．ゴール階層は，アプリケーションユーザーが目指していることを階層的に記述する．

本方法の主たる出力は要求機能 (required function) である．これがソフトウェア要求仕様書の主たる部分となる．それぞれの機能は，ある主体 (コードや人) が何をして良いかの許可として記述される．分析過程の最終段階において，要求機能は与えられた環境化で実行可能となっているが，ゴール階層とは必ずしも一致しなくなる場合がある．

分析を通して，非明示的もしくは認識されていなかったゴールが発見されるため，ゴール階層は，分析中に拡張されたり修正されたりする．環境も，もし可能ならば，ゴール階層に合うように修正される場合がある．結果として，ゴール階層は明確になり，環境も場合によってはゴールに合った形に修正されることになる．以下に分析手順を示す．

(1) 環境の記述とゴール階層を得る．

(2) 要求機能と可能機能を導出する:
最初のステップでは，要求機能をゴール階層から導出する．可能機能も同様に環境記述から導出する．それぞれの機能は，表形式で記述する．

(3) 要求機能と可能機能の違いを認識する:
同じ表形式で記述されているため，その違いを認識するのは容易である．

(4) 要求機能と可能機能の衝突を解消する:
本分析法では以下の二種類の解消法を提供する．

- 環境を修正する: 要求機能とゴールに合致するように環境を変更する．ポリシィはユーザー側のマシンにあるため，その修正は困難ではない．ポリシィの修正によって，他のモバイルコードが余計な機能を実行可能となる場合があるため，前述のツールを用いてそのような副作用をチェックする．コードの配置を修正することは簡単ではない．なぜなら現状のコード配置が他のアプリケーションやプロジェクトにも利用されている可能性があるからである．本稿ではモバイルコードそのものを修正し，その特性や機能は修正しないものと仮定している．
- 要求機能を修正する: 前述の理由で，環境が修正不可能な場合がある．そのような場合は，要求の一部を諦めることで，衝突の解消を行う．結果として，掲げていたゴールの一部が達成されない場合が生じる．本分析手法では，達成されないゴールの発生を食い止める術は無い．本手法では，単に，ゴールと縮小された要求機能とのギャップを記録することで，将来の環境やゴールの変化に備えるのみである．

(5) 要求機能とゴールの曖昧性を解消する:
可能機能と要求機能の差異から，本来記述されるべきにもかかわらず，記述されていない要求を発見することがある．そのような発見を通して，ゴール階層を修正もしくは拡張する．

(6) 環境やゴール階層が変更された場合，上記の手続きを繰り返し，要求機能を完成させる．

5. 例　　題

本節では，e ラーニングのためのアプリケーションを例題として，本分析法の有用性を示す．アプリケーションの要求の概要は以下の通りである．

> インターネット上のユーザーを対象とした e ラーニングアプリケーションを提供したい．それぞれの学習者は，我々のシステムを利用して問題を解き，解かれた問題はそれぞれに採点される．学習者は他の学習者に対してコメントをすることができ，それによって，学習の進行や動機付けを高めることができる．スケーラビリティのため，本アプリケーションは，非集中型の形態で実現しなければならない．問題の解答や採点などの機能は，経済的な理由から新規のコードは記述せず，関連会社の提供するモバイルコードを利用する．

上記の要求文に従い，本アプリケーションは CGI 等のサーバーサイド技術ではなく，モバイルコードを利用したクライアントサイドの処理を中心とするシステムとした．提供されるモバイルコードは以下の通りである．

- Staff.class 採点機能を含む．
- CoLearner.class 他ユーザーにコメントをする機能を含む．
- Learner.class 解答を記述し，採点結果やコメントを参照する機能を含む．

上記のコードは言及されている機能だけでなく，他の機能も同梱されており，コードを直接修正することができないため，セキュリティポリシィを利用して，それぞれのコードの振る舞いを制限することとした．このアプリケーションを利用する際，以下のようなファイル (データ) がユーザーサイドのマシンに保存されるが，それらへのアクセスを適切に制限する必要がある．

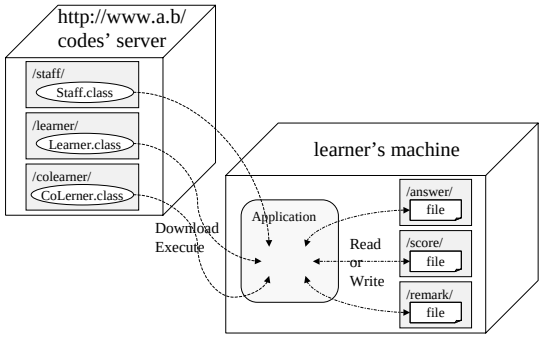

図 4 例題のアプリケーションをとりまく環境

```
grand codeBase "http://www.a.b/" {
     permission java.io.FilePermission "/", "read, write";
}
```

図 5 初期ポリシィ

表 1 初期ポリシィ下での可能機能 (enabled functions)

	Staff	CoLearner	Learner
Answer	read, write		
Score			
Remark			

- 質問に対する解答 (Answer)
- 採点結果 (Score)
- コメント (Remark)

単純化のために，本例題中では，1 つの問題，1 つの解答，そして 1 セットのコメントのみ扱う．このアプリケーションをとりまく環境は図 4 に示す通りである．

5.1 ポリシィを修正する

最初の例題では，要求に合致するようにポリシィを修正する方法を示す．加えて，修正を通して，ゴールの曖昧性を解消する方法も示す．

5.1.1 初期ポリシィ

前述のモバイルコードが実行可能となるために，図 5 に示すようなポリシィをアプリケーションに与えることとする．図 5 のポリシィは，前述のモバイルコードがユーザー側のマシン内のファイルを読み書きするのに十分な権限を与えるものである．このポリシィをもとに，表 1 に示すような可能機能を得る．

5.1.2 アプリケーションのゴール

図 6 に示すゴール階層をもとに，このアプリケーションの要求機能を明確にする．この図において太線の楕円で表されるゴールが直接に要求機能と対応付いている．結果として，表 2 に示すような要求機能を得る．尚，表中の ‘rw’ は read and write の略記であり，‘r’ は read の略記である．

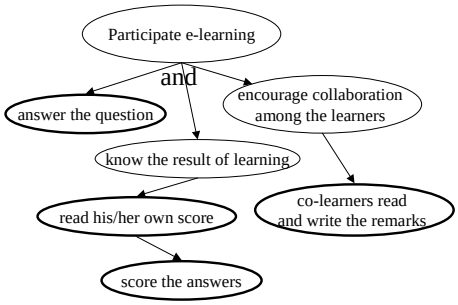

図 6 初期ゴール階層

表 2 初期要求機能

	Staff	CoLearner	Learner
Answer	r		rw
Score	rw		r
Remark		rw	

Conflicts with
enabled functions

5.1.3 ポリシィと要求の衝突

表 2 における ‘r’ は，“read only” を示すのではなく，“少なくとも読めること” を示す．よって，表 1(‘read,write’ となっている) との差異は，必ずしも，衝突を示しているわけではない．しかし，本分析手法では，手順の単純化のため，このような差異を “衝突の候補” としてまず拾い出し，その意味を個々に吟味することで，実際の衝突か否かを判断する．

ここでの例では，表 1 に示す可能機能下では，“スタッフが解答を記述する” ことと，“学習者がスコアを書き換える” ことを容認してしまう．それはゴールに反するため，表 2 の解釈を，“学習者のみが解答を読み書きする” かつ “スタッフのみがスコアを書き換える” に明確化する．この明確化された要求機能に合致するようにポリシィを図 7 のように変更し，要求機能と可能機能を一致させる．

同様に，表 2 における空欄全てが，後述の曖昧な要求であるとは限らないが，曖昧な要求である恐れはある．よって，空欄部分を曖昧な要求の候補として，表 1 の可能機能と比較して，個別に実際の曖昧な要求か否かを判断する．

5.1.4 要求の曖昧性解消

表 2 中のセルの一部は空欄である．我々の分析法では，このようなセルを曖昧性の兆しと見なす．本分析法における要求の曖昧性とは，記述しなければならないにもかかわらず記述されていないことを指し，いわゆる多義性とは異なる．表 2 中の空白セルを順に見当することで，我々は，表 3 に示すように要求機能を明確化した．尚，表 3 中の ‘-’ は, read write 双方とも

```
grand codeBase "http://www.a.b/staff/" {
permission java.io.FilePermission "/answer/*", "read";
permission java.io.FilePermission "/score/*", "read,write";
permission java.io.FilePermission "/remark/*", "read,write";
}

grand codeBase "http://www.a.b/learner/" {
permission java.io.FilePermission "/answer/*", "read,write";
permission java.io.FilePermission "/score/*", "read";
permission java.io.FilePermission "/remark/*", "read,write";
}

grand codeBase "http://www.a.b/colearner/" {
permission java.io.FilePermission "*", "read, write";
}
```

図 7 衝突解消後のポリシィ

表 3 明確化された要求機能

	Staff	CoLearner	Learner
Answer	r	-	rw
Score	rw	-	r
Remark	rw	rw	rw

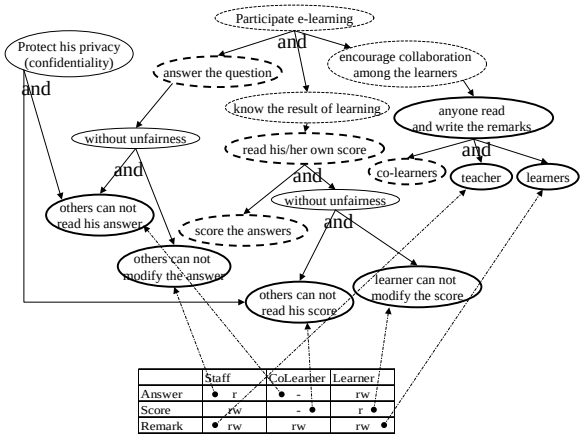

図 8 拡張されたゴール階層

許可されていないことを示す.

5.1.5 ゴール階層の更新

表 3 のように得られた要求機能をもとに，ゴール階層を図 8 のように拡張する．図中の破線および点線の楕円は既存のゴールを示す．太線のゴールは前述のように直接に要求機能と関係するゴールを示す．図 8 に示すように，要求機能と，それに関係するゴール階層内のゴールとの間を点線でつなぐことで，ゴールと機能間の追跡可能性を確保する．

5.2 要求の一部を破棄する

次の例題は，ポリシィにあわせて，要求の一部を破棄する場合を示す．我々の分析法ではあるべき例題と修正した要求との間の追跡可能性を保持することで，その要求が破棄された理由や方法を容易に知ることができる．

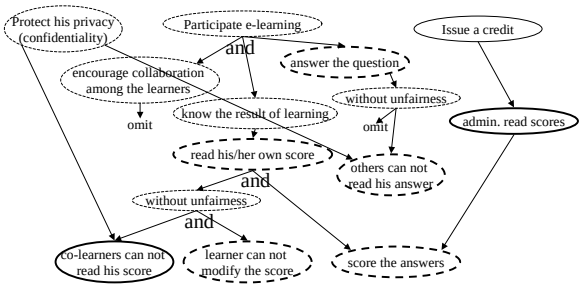

図 9 ビジネス拡大後のゴール階層

表 4 図 9 をもとにした要求機能

	Teacher	Admin.	CoLearner	Learner
Answer	r	-	-	rw
Score	rw	r	-	r
Remark	rw		rw	rw

5.2.1 要求とゴールの追加

ビジネス拡大のため前述の e ラーニングのアプリケーションに以下の要求を追加した.

> 提供する e ラーニングコースそれぞれについて単位認定の機能を追加する．将来，ここでの単位はある大学での講義受講単位と互換性をとる予定である.

この文書をもとに，図 9 に示すようにゴール階層を修正した．点線の楕円で表現されたゴールは既存のゴールであり，書面の都合，一部は省略した．図 8 における “他者のスコアを読めない” を，図 9 では，“他の学習者のスコアを読めない” に修正した．単位を発行するためには，管理者 (Admin.) に相当する人物が各学習者のスコアを参照しなければならないが，管理者も他者に含まれてしまうためである．表 4 に，図 9 のゴール階層をもとにした要求機能を示す．表 3 では，単に “staff” と分類していた主体を，表 4 では，“teacher” と “admin.” に分割した.

5.2.2 コードのパッケージングに関する制約

表 4 に示すような新しい要求機能を認識したが，この機能要求を現環境下では充足できないことがわかった．その理由は，teacher と admin. の機能が，1 つのモバイルコード Staff.class によって提供されており，本例題中では，コードの分割が不可能なためである．結果として，表 5 に示すような可能機能しか現環境下では提供できないことがわかる．表 5 の可能機能と，表 4 の要求機能は明らかに衝突しているが，前述のように環境を変更できないため，表 5 の可能機能を新しい要求機能として採用し，本来の要求機能の一部を破棄することにした.

5.2.3 妥協に関する追跡可能性

環境に関する制約によって要求機能の一部を破棄したため，アプリケーションユーザーのゴールの一部は

表 5 パッケージングの制約による可能機能

	both in Staff.class			
	Teacher	Admin.	CoLearner	Learner
Answer	r	r	-	rw
Score	rw	rw	-	r
Remark	rw	rw	rw	rw

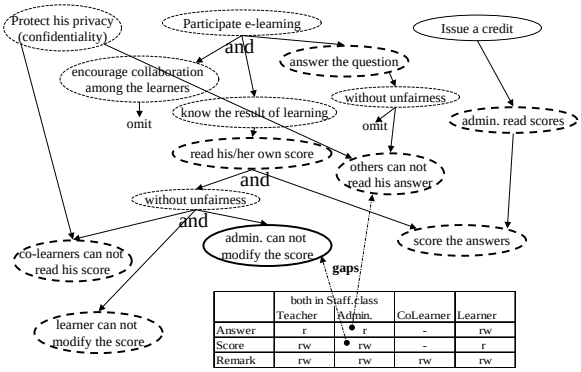

図 10 ゴール階層と妥協された要求機能の関係

達成されなくなる．具体的には，“他者が解答を参照できない” と “管理者はスコアを変更できない” というゴールが達成されない．図 10 に，ゴール階層と妥協された要求機能の関係を示す．ゴール階層を上向きにたどることで，より上位のゴールである “プライバシーを保護する”，“自分のスコアを (正しく) 参照する” 等のゴールが完全には達成されないことがわかる．

無論，現実には管理者がスコアを不正改竄するということは考え難いため，ここでの変更はそれほど重大ではない．しかし，あくまで現要求機能はユーザーのゴールとは完全に一致していないことを記録することで，要求に関する正しい解釈を与えることができると思われる．このような理由から，本分析法は，たとえ要求機能が変更されても，ゴール階層は変更せず，ゴール階層と要求機能のギャップを明示的に記録する方針をとっている．

6. おわりに

本稿では，Java モバイルコードのセキュリティポリシィと Java アプリケーションのトレードオフ分析手法を提案した．本方法を利用することで，本来のゴールを完全には達成できないにしても，実現可能な要求仕様を得ることが可能となる．加えて，どのゴールが破棄されたかを明確に把握することができ，将来の環境変化に対応して要求変更を行う際の指針を残している．手法の一部は，我々の開発したセキュリティポリシィチェッカー&ジェネレータによって半自動的に行うことが可能である．

ポリシィを含む環境とゴールとの衝突回避のために，本手法では，ゴールの達成を一部諦めるか，環境を変更するかどちらかの方針をとる．しかし，現実には，要求機能を満足し，現環境下で適切に動作可能な別の (モバイル) コードを利用するという第三の選択肢もありうる．(モバイル) コードに関する仕様のカタログ等が完備されていれば，この第三の方針を選択することも可能である．この際，仕様マッチ[9] の技術が適切なコード選択を行う際に役立つと思われる．

本稿では，Java モバイルコードを用いたアプリケーションのみ扱っているが，一般的なモバイルコードについて，この手法を拡張して行きたい．我々は日々の生活において，詳細が不明なプログラムやライブラリをダウンロードし，それほど注意も払わずに利用している．そういった意味からは，我々の手法はモバイルコードだけでなく，プログラムやライブラリ一般に対して適用できる可能性があると思われる．このような拡張のためには，より詳細なポリシーを記述できる言語が必要となるが，たとえば，Ponder[10] 等の利用が見当できると思われる．

セキュリティ要求に関する研究の重要性は，文献[11] で議論されている．そこでは 6 項目の問題提起がされており，そのうち半数の問題中でセキュリティポリシィが言及されている．以下ではこの文献の流れにそって関連研究を概観する．

最初に，アンチ要求と呼ばれる要求について述べる．これは，不正侵入者等に代表される悪意あるユーザーの要求を示すものである．アンチ要求を発見するために，ミスユースケースや否定的シナリオが用いられる[12][13]．しかし，これらを使ったとしても，アンチ要求を想起することはそれほど容易ではないように思われる．我々の分析法では，ある環境下で実際にできてしまうことを明確にすることで，アンチ要求の洗い出しが容易であると思われる．ゴール指向要求分析の手法はこの分野でも適用されており[14]，非機能要求 (NFR) の分類辞書もアンチ要求を洗い出す助けになると思われる．

次に組織におけるセキュリティポリシィについて議論する．文献[15] では，セキュリティポリシィの抽象度を以下の 4 レベルに分類している; 組織要求，コンピュータポリシィモデル，アクセス制御モデル，実装モデル．組織レベルの研究は文献[16][17] 等で述べられており，そこでもゴール指向分析が利用されている．一方，我々の分析法は，下層レベルのセキュリティポリシィに言及したものである．文献[11] では，以下のことが言及されている．

> セキュリティポリシィに関する議論のほとんどは問題領域ではなく，解法の領域から発生しているが，そのようなことでは，セキュリティポリシィに関する系統的な探求を促進しない．

我々の手法も解法の領域に大きく依存しているが，それでも，セキュリティポリシィに関する系統的な探求に貢献すると思われる．モバイルコードを利用したサービス構築の分野では，動的にサービスを構築できることなどからもわかるように，実装と要求が密接に関連付いている．よって，この分野に限っていえば，解法の領域における議論は，要求の探求に直接貢献するものと思われる．無論，モバイルコードのセキュリティに関しては既に幅広い研究が行われているが[18]，要求仕様との関係を議論したものはほとんど見られない．

すでに述べたが，ゴール指向分析は，セキュリティ要求を分析する際に幅広く利用されている[14)17)19)]．我々自身も拡張型のゴール指向分析法である AGORA を提案している[20]．本稿の分析では，ステークホルダ間の衝突を扱わないため，単純なゴール指向分析法を利用しているが，そのような衝突を扱う拡張を今後は行い，ゴール階層の記法も AGORA を採用してゆく予定である．

謝　　辞

本研究の一部は，2003(平成 15) 年度 文部科学省科学研究費補助金 (若手研究 (B) 課題番号 15700028) の支援を受けている．ここに記して謝意を表す．

参 考 文 献

1) Kaiya, H., Furukawa and Kaijiri, K.: Security Policy Checker and Generator for Java Mobile Codes, *Engineering Information Systems in the Internet Context (EISIC)*, IFIP TC8/WG8.1, Kluwer Academic Publishers, pp. 255–264 (2002).
2) van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour, *RE'01*, pp. 249–263 (2001).
3) Thorn, T.: Programming languages for mobile code, *ACM Computing Surveys*, Vol. 29, No. 3, pp. 213–239 (1997).
4) JODE decomplier homepage: `http://jode.sourceforge.net/` (2003).
5) : IEEE Recommended Practice for Software Requirements Specification (1998). IEEE Std 830-1998, ISBN 0-7381-0332-2 SH94654(Print).
6) Kavakli, E.: Goal-Oriented Requirements Engineering: A Unifying Framework, *Requirements Engineering*, Vol. 6, pp. 237–251 (2002).
7) Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J.: *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishers (2000).
8) van Lamsweerde, A., Darimont, R. and Letier, E.: Managing Conflicts in Goal-Driven Requirements Engineering, *IEEE Transactions on Software Engineering*, Vol. 24, No. 11, pp. 908–926 (1998).
9) Zaremski, A. M. and Wing, J. M.: Specification Matching of Software Components, *ACM Trans. Software Eng. and Methodology*, Vol. 6, No. 4, pp. 333–369 (1997).
10) Damianou, N., Dulay, N., Lupu, E. and Sloman, M.: The Ponder Specification Language, *Proc. of Policy 2001: Workshop on Policies for Distributed Systems and Networks*, pp. 18–39 (2001).
11) Crook, R., Ince, D., Lin, L. and Nuseibeh, B.: Security Requirements Engineering: When Anti-requirements Hit the Fan, *IEEE Joint International Requirements Engineering Conference, RE'02*, Essen, Germany, pp. 203–205 (2002).
12) Alexander, I.: Initial Industrial Experience of Misuse Cases in Trade-Off Analysis, *IEEE Joint International Requirements Engineering Conference, RE'02*, Essen, Germany, pp. 61–68 (2002).
13) Sindre, G. and Opdahl, A. L.: Templates for Misuse Case Description, *REFSQ'2001 Proceedings* (2001).
14) Chung, L.: Dealing with Security Requirements during the development of information systems, *CAiSE'93 proceedings* (1993).
15) Thomas, R. K. and Sandhu, R. S.: Conceptual Foundations for a model of Task-based Authorizations, *Proc. of IEEE Computer Security Foundation Workshop VII* (66-79(ed.)) (1994).
16) Anton, A., Earp, J., Potts, C. and Alspaugh, T.: The Role of Policy and Stakeholders Privacy Values in Requirements Engineering, *RE'01 proceedings*, pp. 138–145 (2001).
17) Anton, A., Earp, J. and Reese, A.: Analyzing Website Privacy Requirements Using a Privacy Goal Taxonomy, *IEEE Joint International Requirements Engineering Conference, RE'02*, Essen, Germany, pp. 23–31 (2002).
18) Rubin, A. D. and Daniel E. Geer, J.: Mobile Code Security, *IEEE Internet Computing*, Vol. 2, No. 6, pp. 30–34 (1998).
19) van Lamsweerde, A. and Letier, E.: Handling Obstacles in Goal-Oriented Requirements Engineering , *IEEE transaction of Software Engineering*, Vol. 26, No. 10, pp. 978–1005 (2000).
20) Kaiya, H., Horai, H. and Saeki, M.: AGORA: Attributed Goal-Oriented Requirements Analysis Method, *IEEE Joint International Requirements Engineering Conference, RE'02*, pp. 13–22 (2002).