UMLによるプロテクションプロファイルのモデル化とその形式的検証

森本　祥一[†a)]　　程　京徳[†]

Modeling Protection Profiles by UML and Their Formal Verification

Shoichi MORIMOTO[†a)] and Jingde CHENG[†]

あらまし　プロテクションプロファイルは，様々な情報システムにおけるセキュリティ仕様書の雛形であり，ISO/IEC15408 という国際基準によりセキュリティが保証されている．しかしながら，扱う分野についての明確な規定がないことや，文書構造に問題があるため，実際にはほとんど利用されていない．本論文ではプロテクションプロファイルを効果的に利用できるよう，UML によるモデル化を提案する．このモデル化により，ISO/IEC15408 のセキュリティ基準に準拠した仕様記述を容易にする．また，モデル化したプロテクションプロファイルを用いて記述した仕様がセキュリティ基準を満たすかどうかを，定理証明技法とモデル検査技法により検証する技法を示す．

キーワード　ISO/IEC15408，コモンクライテリア，デザインパターン，定理証明，モデル検査

1. まえがき

近年のインターネットの普及を始めとする我々の社会におけるコンピュータネットワークの発展は，目覚ましいものがある．しかし，これらの発展に伴う利便性の向上の反面，コンピュータウィルスやコンピュータ犯罪による被害も，年々増加している [30]．今後も，IT 製品や情報システムに対して，より高い安全性が要求される．今やセキュリティ機能は情報システムにおいて不可欠である．このため，情報システムのセキュリティ仕様記述とその検証は重要な課題となっている．

セキュリティを十分に備えた情報システムを開発するためには，システムに必要なセキュリティ要求を的確に定義し，システムがそれらの要求を満たしているかどうかを確かめる必要があるが，一般的な基準が存在しないため，必要なセキュリティ要求は十分であるか，本当にそれらの要求を満たしているかを判断することは困難である．また，新しいシステムを開発するたびにこれらの作業を行うと，工数やコストもかかる．このような問題を解決するために，ある特定の分野の情報システムにおいて，どのようなセキュリティ要求を満たせばよいのかを定義したセキュリティ仕様書の雛形として，プロテクションプロファイルが提案された [6], [17]．プロテクションプロファイルは，情報システムのカテゴリー（OS，DBMS，電子商取引，ファイアウォール等）ごとにおけるセキュリティ要求のセットを定義している．

プロテクションプロファイルには，そのプロテクションプロファイルで取り上げている分野において想定される脅威，想定される脅威に対抗するためのセキュリティ対策方針，セキュリティ対策方針を実装するために必要なセキュリティ機能，等が定義されている．特に，必要なセキュリティ機能に関しては，IT 製品や情報システムのセキュリティ評価のための国際標準である ISO/IEC15408 [17] で定められたセキュリティ機能要件から引用している．

プロテクションプロファイルは，インターネット上からダウンロードして誰でも利用可能になっており，それ自身で ISO/IEC15408 の認証を受けることが可能である．プロテクションプロファイル登録のための ISO/IEC15292 [18] という国際基準も存在する．つまり，公開されているプロテクションプロファイルは，既に ISO/IEC15408 で定義されたセキュリティ基準を満たしている．しかしながら，これらのプロテクションプロファイルは実際にはあまり利用されていない．

† 埼玉大学大学院理工学研究科情報数理科学専攻，さいたま市
Information and Mathematical Sciences, Graduate School of Science and Engineering, Saitama University, 255 Shimookubo, Sakura-ku, Saitama-shi, 338–8570 Japan
a) E-mail: morimo@aise.ics.saitama-u.ac.jp

ISO/IEC15408 の認証を受けた製品・システムの公開セキュリティ仕様書 [7], [31] においても，ほとんど参照されていない．この理由は，以下に挙げられるようなことが考えられる．

プロテクションプロファイルで扱う分野についての明確な規定がないため，同じ分野，類似する分野のプロテクションプロファイルが多数存在し，どのプロテクションプロファイルを参照してよいか不明りょうである．あるプロテクションプロファイルで定義されているセキュリティ要求が，別のプロテクションプロファイルにおいて別の形で再定義されていたりする．

加えて，プロテクションプロファイルの文書構造に問題があるため，その中から必要な部分を見つけることが難しい．プロテクションプロファイルでは，前述のように脅威・対策・機能が述べられているが，これら以外の記述も多く含まれている．また，脅威・対策・機能の対応関係も相互に複雑に絡み合っている．更に記述が抽象化されているため，一見してどの部分を利用できるのか分かりにくい．

以上のように，プロテクションプロファイルは現状では非常に利用しにくいものとなっている．そこで本論文では，上記の問題を解決するために，プロテクションプロファイルの不必要，冗長である部分を排除し，必要な部分のみを整理した後，UML によりモデル化することを提案する．このモデル化により，プロテクションプロファイルの効果的な利用を可能にし，情報システムにおけるセキュリティ仕様書作成の負担の軽減が期待できる．加えて，モデル化したプロテクションプロファイルやこれらを用いた仕様記述・設計が実際に ISO/IEC15408 のセキュリティ基準を満たしているかどうかを形式手法により厳密に検証する技法を示す．

2. プロテクションプロファイルのモデル化

ソフトウェア仕様記述・設計では，システムが見通しよく記述・設計され，それらが他者によく伝わっていることが重要である．このため，システムの記述・設計を共通の認識で理解し合えるような表記法として，UML が提案された [24]．本研究では，意味的な統一を図るため，また汎用性の向上のため，ソフトウェアモデリングのデファクトスタンダードである UML でプロテクションプロファイルをモデル化した．

まず，プロテクションプロファイルの整理・分割についての方針を示す．次にモデル化の手順を示し，その手順に従った実際のモデル化の例を示す．

表 1　プロテクションプロファイルにおける脅威と対策方針の対応表
Table 1　Mapping between threats and security objectives.

	T.NOAUTH	T.REPEAT	T.REPLAY	T.ASPOOF
O.IDAUTH	X			
O.SINUSE		X	X	
O.MEDIAT				X
O.SECSTA				X
O.ENCRYPT		X		
O.SELPRO			X	

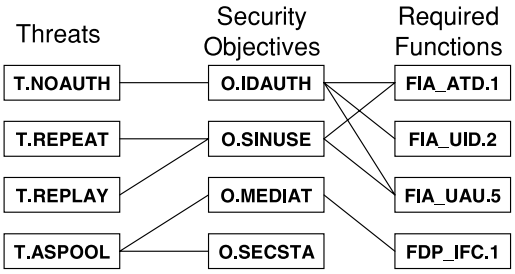

図 1　プロテクションプロファイルにおける各要素の関係
Fig. 1　Relations between the PP's contents.

2.1 整理・分割の方針

プロテクションプロファイルでは，表 1 のように，脅威と対策方針，対策方針と必要な機能の対応関係がまとめられている．T.NOAUTH や O.IDAUTH は，脅威や対策方針の名前であり，接頭辞 T が付いているものは脅威を，接頭辞 O が付いているものはセキュリティ対策方針を示す．同様に，セキュリティ対策方針とそれらを実現するためのセキュリティ機能要件の対応関係も，マッピングされる．これらの対応関係の一部を図で表現すると，図 1 のようになる．

この図 1 から分かるように，プロテクションプロファイルのそれぞれの要素は多対多の関係になっており，相互に複雑に絡み合っている．ここで，これらを整理，分割する方針を検討する．

Erich Gamma らが提案した GoF のデザインパターン [12] では,「問題」と「解法」をセットとして一つのパターンとして提供している.「問題」とはどのような場合にパターンを適用すべきかを記述したもので,「解法」とはどのようにそれらの問題を解決するかを示している．一方，プロテクションプロファイルにおける

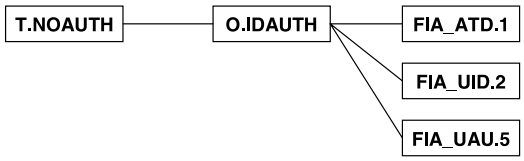

図 2　図 1 の分割例
Fig. 2　The example of divided Fig. 1.

想定される脅威とは，ある情報システムの分野におけるセキュリティに関する問題であり，プロテクションプロファイルにおけるセキュリティ対策方針とは，それらの問題を解決するための解決策である．つまり，GoF のデザインパターンにおける問題と解法は，それぞれプロテクションプロファイルにおける想定される脅威とセキュリティ対策方針と一致する．この理由により，GoF のデザインパターンに倣ってプロテクションプロファイルにおける脅威と対策方針の対を一つの単位として分割することとする．例として，図 1 を脅威ごとに分割し，T.NOAUTH のみについて抽出すると，図 2 のようになる．

このように分割することで，プロテクションプロファイルの各要素は非常に単純になり，また脅威ごと，つまり機能的な単位ごとに独立して実装可能になる．よって我々は，一つの脅威に対する機能群を一つの単位とし，この単位でプロテクションプロファイルを分割した上でモデル化し，利用できるようにする．

2. 2　モデル化の手順

本論文で提案するプロテクションプロファイルのモデル化は，以下の手順で行う．

1. **2.1** で示した方針に基づき，脅威ごとに分割する．

2. 他のプロテクションプロファイルにおいて，同内容の脅威がないか，検索する．

3. 同内容の脅威が多数見つかった場合，それらの脅威に対する対策方針が要求している機能要件のうち，すべてに共通に要求されている機能要件を選択する．

4. 手順 3 で選んだ以外の機能要件については，特定の分野に依存する機能要件は選択せず，手順 3 で選んだ機能要件と依存関係がある機能要件や，セキュリティ保証の観点から明らかに必要であると思われる機能要件を選択する．

5. 手順 3 と 4 で選択したすべての機能要件をまとめ，オブジェクト指向分析・設計 [3] に基づいて UML により一つのパターンとして表現する．

我々は，上記の手順に従ってプロテクションプロファイルを以下に示すようなパターンとしてモデル化した．不正なユーザアクセスを防ぐ「認証パターン」，データの不正参照・改ざん・消去といった脅威に対する「データ保護パターン」や，物理的な媒体の損傷等によるデータの喪失に対抗する「データ保存パターン」，セキュリティ監査クラス FAU 等を用いた「システム監査パターン」等が挙げられる（表 2）．

表 2　モデル化の例
Table 2　The examples of modeling.

パターン名	必要なセキュリティ機能要件
認証パターン	FIA_ATD, FIA_UID, FIA_UAU, FIA_USB, FMT_MSA
データ保護パターン	FDP_ITT, FDP_ACC, FDP_ACF, FCS_COP
データ保存パターン	FPT_TDC, FPT_TRC, FDP_ETC, FDP_ITC, FPT_ITA, FPT_ITC, FPT_ITI

2. 3　モデル化の例

本節では，例として **2.2** で示した「認証パターン」のモデル化の過程を示す．

以下は，あるファイアウォールのプロテクションプロファイル [10] で述べられている脅威のうちの一つである．

> **T.NOAUTH** An unauthorized person may attempt to bypass the security of the TOE so as to access and use security functions and/or non-security functions provided by the TOE.

原文中の TOE（Target of Evaluation）とは，ISO/IEC15408 の評価を受ける対象システムの呼称である．T.NOAUTH は，権限のないユーザが，システムに対して攻撃をするかもしれないことを想定している．この脅威に対し，以下のようなセキュリティ対策方針が述べられている．

> **O.IDAUTH** The TOE must uniquely identify and authenticate the claimed identity of all users, before granting a user access to TOE functions. This security objective is necessary to counter the threat: T.NOAUTH because it requires that users be uniquely identified before accessing the TOE.

O.IDAUTH は，システムへのアクセスを許可する前に，厳密なユーザ認証を行わなければならない，と述べている．この対策方針を実現するためには，ISO/IEC15408 で定められているセキュリティ機能要件のうち，FIA_ATD，FIA_UID，FIA_UAU が必要だと述べられている．それぞれの原文を以下に示す．

FIA_ATD.1 User attribute definition
FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users:
a) indentity
b) association of a human user with the authorized administrator role
c) any other user security attributes to be determined by the Security Target writer(s)

原文中の TSF（TOE Security Functions）とは，対象システムのセキュリティ機能のことである．FIA_ATD は，原文中で列挙したセキュリティ属性をもたなければならない，という要件である．以下は FIA_UID の原文である．

FIA_UID.2 User identification before any action
FIA_UID.2.1 The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

FIA_UID は，システムにおけるあらゆるアクションよりも前に，ユーザ認証を行わなければならない，という要件である．以下は FIA_UAU の原文である．

FIA_UAU.5 Multiple authentication mechanisms
FIA_UAU.5.1 The TSF shall provide *password and single-use authentication mechanisms* to support user authentication.
FIA_UAU.5.2 The TSF shall authenticate any user's claimed identity according to the *rules describing how the multiple authentication mechanisms provide authentication.*

FIA_UAU は，認証方式として，パスワードによる認証と，単一使用認証を提供しなければならない，という要件である．また，それぞれの認証方式を，列挙したルールに従って使い分けることを要求している．ここでは，ルールについての引用は省略した．

一方で，ある DBMS のプロテクションプロファイル [27] では，以下のような脅威が述べられている．

T.ACCESS Unauthorised Access to the Database. An outsider or system user who is not (currently) an authorised database user accesses the DBMS. This threat includes: Impersonation - a person, who may or may not be an authorised database user, accesses the DBMS, by impersonating an authorised database user (including an authorised user impersonating a different user who has different - possibly more privileged - access).

この脅威に対する方針は，以下のように述べられている．

O.I&A.TOE The system shall identify and authenticate users prior to providing access to any TOE facilities.

これらの脅威と対策は，前述のファイアウォールのプロテクションプロファイルにおける T.NOAUTH と O.IDAUTH に一致する．同様に，同じ内容の脅威と対策が，文献 [8] をはじめとする多くのプロテクションプロファイルにおいて述べられている．つまり，大抵の分野の情報システムにおいて，T.NOAUTH や T.ACCESS のような脅威が想定されるということである．この事実をもとに，T.NOAUTH や T.ACCESS に代表される脅威に対抗し得るような機能に関するセキュリティ機能要件群を「認証パターン」と名づけ，モデル化することとする．

T.NOAUTH と O.IDAUTH のような脅威と対策方針が述べられているプロテクションプロファイルにおいては，FIA_ATD，FIA_UID，FIA_UAU が共通に要求されている．DBMS のプロテクションプロファイルでは，O.I&A.TOE を実装するために，これら以外に，FIA_USB，FMT_MSA，FMT_MTD，FTA_MCS が必要だと述べられている．これらのうち，FTA_MCS と FTA_TSE はデータベース特有のセッション機能に関するセキュリティ機能要件として要求されているため，認証機能のみについて考えた場合には，これらは不要である．また，いくつかのプロテクションプロファイルでは前述の要件以外に FIA_USB，FMT_MSA，FMT_MTD が要求されている．ここで，ひとまず共通に要求されている FIA_ATD，FIA_UID，FIA_UAU を認証パターンのために必要な機能として採用し，残りのセキュリティ機能要件が必要かどうか，検討する．FIA_USB の原文は，以下のとおりである．

FIA_USB.1 User-subject binding
FIA_USB.1.1 The TSF shall associate the appropriate user security attributes with subjects acting on behalf of that user.

IPA による ISO/IEC15408 の翻訳 [32] と付属書によると，原文中の subjects とは「利用者を代行して動作するサブジェクト」と定義されており，「サブジェクトが生成されたとき，そのサブジェクトは，その生成を起動した利用者を代行して動作する」とある．つまり FIA_USB は，認証されたユーザは，このサブジェクトを通して間接的にシステムを利用しなければならない，という要件である．ISO/IEC15408 で定義されているセキュリティ機能要件は，依存性という項目をもつ．機能要件同士に依存関係がある場合，依存性の項目の欄に，依存する機能要件名が明記される．FIA_USB は，既に採用した FIA_ATM と依存関係

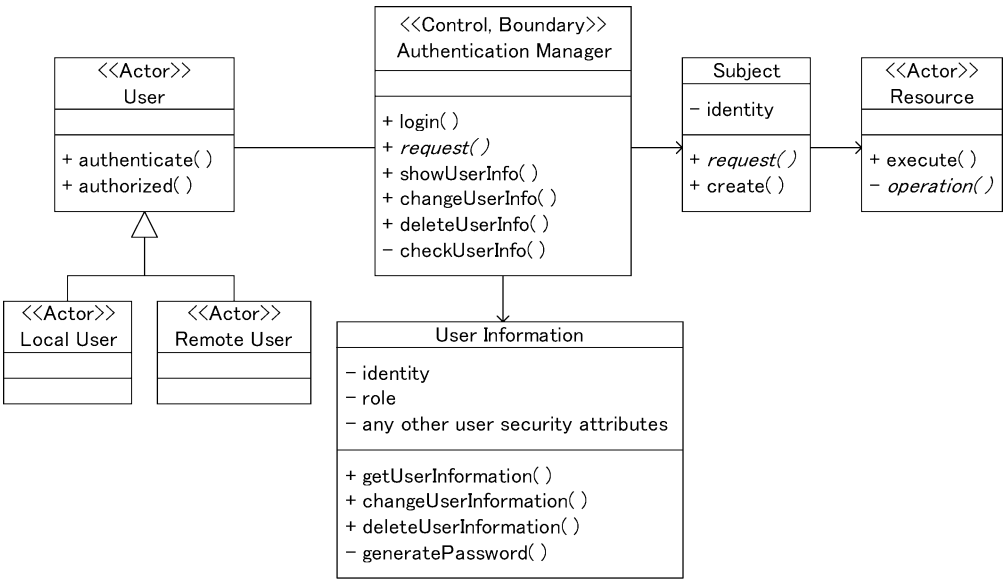

図 3　認証パターン：クラス図
Fig. 3　Authentication Pattern: The class diagram.

にあると明記されているため，FIA_USB も必要な機能要件として採用する．

FMT_MSA は，セキュリティ属性の管理機能であり，セキュリティ属性のデフォルト値変更，問合せ，改変，削除の機能を備えなければならない，と述べている [6]．ファイアウォールのプロテクションプロファイルは，これらの機能は O.IDAUTH のための機能としては要求しておらず，他の対策の機能として要求している．一方で，その他のプロテクションプロファイルでは，T.NOAUTH と同様の対策のための機能として，FIA_ATM と併せて FMT_MSA が要求されている．既に採用した FIA_ATM によってセキュリティ属性が与えられるので，セキュリティ保証の観点から考えると，これらの管理機能もまた同時に備えるべきである．

FMT_MTD は，TSF データ管理機能である [6]．TSF データとは，例えばシステムログをとったときの，現在時刻や監査証跡である．今回は認証機能にのみ着目し，システム監査の機能については考慮していないため，不要である．

以上より，FIA_ATD，FIA_UID，FIA_UAU，FIA_USB，FMT_MSA を認証機能に必要なセキュリティ機能要件であるとみなし，図 3，図 4 で示すように，これらの要件をオブジェクト指向分析・設計に基づいて UML により表現する．

図 3 において，クラス User とその継承クラスは，システムにおけるユーザを表す．クラス Resource は，システムにおける種々のリソースを表す．これらのクラスは，実装範囲外の Actor クラスである．クラス Authentication Manager は，ユーザ認証とユーザ情報の管理を司る Control クラスであり，ユーザとのインタフェースである Boundary クラスでもある．クラス User Information は，ユーザ情報を保持するクラスである．クラス Subject は，ユーザがクラス Resource にアクセスするためのサブジェクトである．それぞれのクラスにおける斜字体で記述されたメソッド（request，operation）は，パターンを適用する対象システムごとに具体化する．

クラス User Information が，FIA_ATD によって要求されているセキュリティ属性を，属性としてもつ．同様に，FMT_MSA で要求されるセキュリティ属性管理機能を，メソッドとしてもつため，これらの要件は満たされる．クラス Resource は，クラス Subject のメソッド request によってのみアクセスされるため，FIA_USB もまた満たされる．

しかし，図 3 のみでは，FIA_UID と FIA_UAU は満たされていない．これらは動的な振舞いのため，静的構造図であるクラス図上では表現できない．よって，

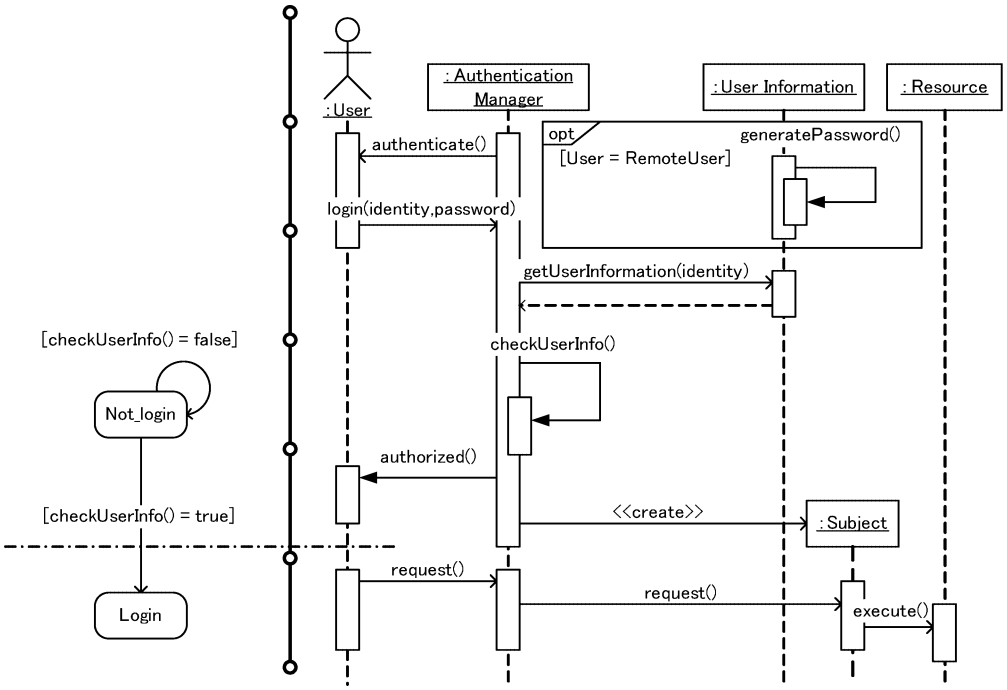

図 4　認証パターン：シーケンス図とステートマシン図
Fig. 4　Authentication Pattern: The sequence diagram and the state machine diagram.

シーケンス図上でこれらを表現した（図 4 右部分）．また，このシーケンス図上における User オブジェクトの状態の変化をステートマシン図で表した（図 4 左部分）．クラス Authentication Manager は，まず最初にユーザに対し authenticate と login によって認証を行う．ユーザ識別情報とパスワードの入力を要求し，ユーザはこれらを入力する．Authentication Manager は，User Information から getUserInformation により保持されているユーザの情報を得て，checkUserInfo によりユーザが入力した情報と比較する．入力された情報が正しければ，ログインを許可（状態 Login に遷移）する．同時に，User が Resource にアクセスするための Subject を生成する．ユーザは，この Subject を介して Resource に対する要求を実行する．入力された情報が不正な場合，ログインを許可しない（状態 Not_login を維持する）．以上のようなシーケンスにより，FIA_UID が満たされる．更に FIA_UAU は，ローカルユーザについてはパスワード認証で，リモートユーザについては単一使用認証で，それぞれ認証しなければならない，と要求している．この要求を満たすために，シーケンス図の記法の一つである Interaction Operator を使用した．opt という接頭辞のフレームに囲まれたシーケンスは，[ ] 内に記述された条件を満たしたときにのみ実行される．図 4 では，FIA_UAU で要求されているとおりに，ローカルユーザの場合は普通のパスワード認証を行い，リモートユーザの場合にのみ接続前に generatePassword により使い捨てパスワードを生成しておく，つまりワンタイムパスワードによる単一使用認証を行う [13].

以上の図 3 と図 4 の解説は，GoF のデザインパターンの記述における「構成要素」と「協調関係」に該当する [12]．UML は，それ単体のみではそれぞれのクラスや属性，メソッドがどのようなものであるか，判断することができない．よってこれらの解説と UML 図をセットとして提供する必要がある．図 3 と図 4，それらの解説により，前述の要求されたすべてのセキュリティ機能要件が満たされることが分かった．つまり，認証パターンは，T.NOAUTH，T.ACCESS に代表されるような脅威に対抗し得るということである．

3.　モデルの検証

2. では，UML を使ってプロテクションプロファイ

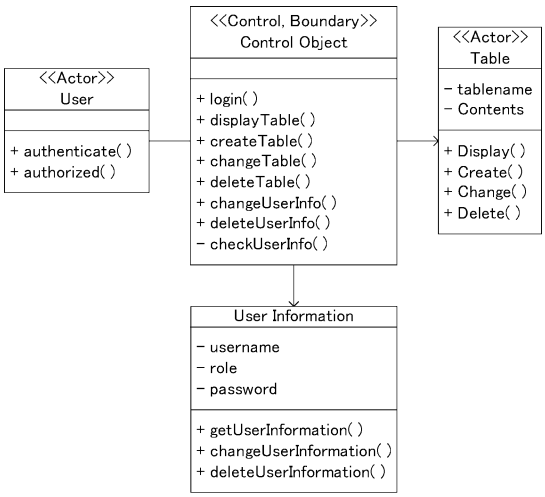

図 5　データベースシステムの設計例
Fig. 5　The first attempt at the design of the database.

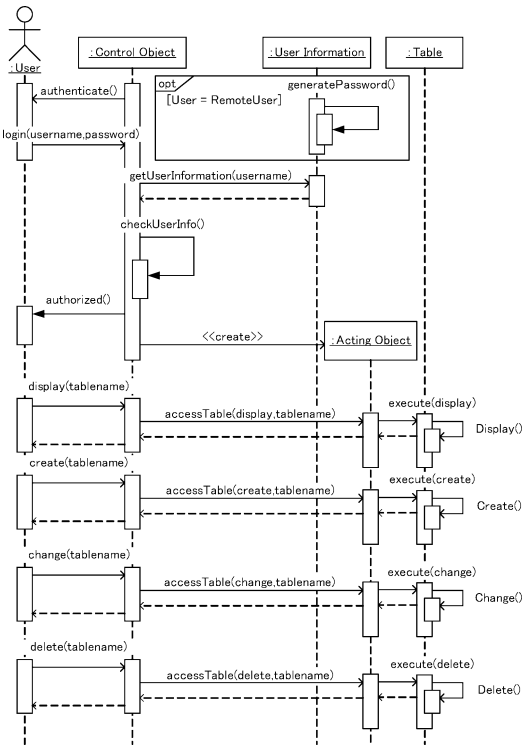

図 6　認証パターン適用後のシーケンス図
Fig. 6　The sequence diagram of the example after applying authentication pattern.

ルをモデル化した．しかし，UML は図の表記法を規定したものであり，内容の自由度は制限してない．したがってどのようなシステムでも記述でき，記述された内容の信頼性が保証されているわけではない．よって本章では，モデル化したプロテクションプロファイルとこれらを用いた仕様記述・設計が，実際にセキュリティ機能要件を満たしているかどうかを検証する方法を示す．例として，まず前述の認証パターンを使って簡単なシステムを設計し，次に，その設計例が，認証パターンのもととなったセキュリティ機能要件を満たしているかどうか，形式手法によって検証する．

3.1　パターンの適用例

以下に，単純なデータベースシステムの仕様を示す．

仕様 n（自然言語）

・ユーザはデータベースにログインする際にユーザ名とパスワードを入力する．

・システムは，登録されているユーザに，ユーザ情報の変更を許可する．

・ユーザは，一般ユーザと管理者権限をもつユーザに分類される．

・管理者権限をもつユーザは，データベース上に表の作成，変更，削除を許可される．

・一般ユーザは，表の参照のみ許可される．

・データベース自体は，既存のものを使用する．

この仕様 n に対して，図 5 のようなシステムを設計する．クラス Control Object は，ユーザ認証，ユーザ情報の管理，データベース上の表へのアクセスを制御する．クラス User Information は，ユーザ名 username，権限 role，パスワード password を属性としてもつ．クラス Table は，データベース上の表を表しており，表示，作成，変更，削除の機能を提供している．これらの機能は既存のデータベースを使う，と仕様 n に明記されているため，Actor クラスとした．必要最低限ではあるが，図 5 は仕様 n を満たしている．しかしながら，このような設計は，セキュリティに関する機能をほとんど考慮していないため，決して安全であるとは保証できない．仕様 n に，前述の T.NOAUTH や T.ACCESS といった脅威に対抗すべきだという要件が加えられた場合，図 5 をどのように変更すべきか再検討しなければならず，また対抗できるという根拠も示さなければならない．そのような場合に，前述の認証パターンを適用する．図 6 と図 7 は，図 5 に対して認証パターンを適用したものである．

図 5 と図 7 の違いは，ローカルとリモートの 2 種類のユーザが存在する点，クラス Table が直接アク

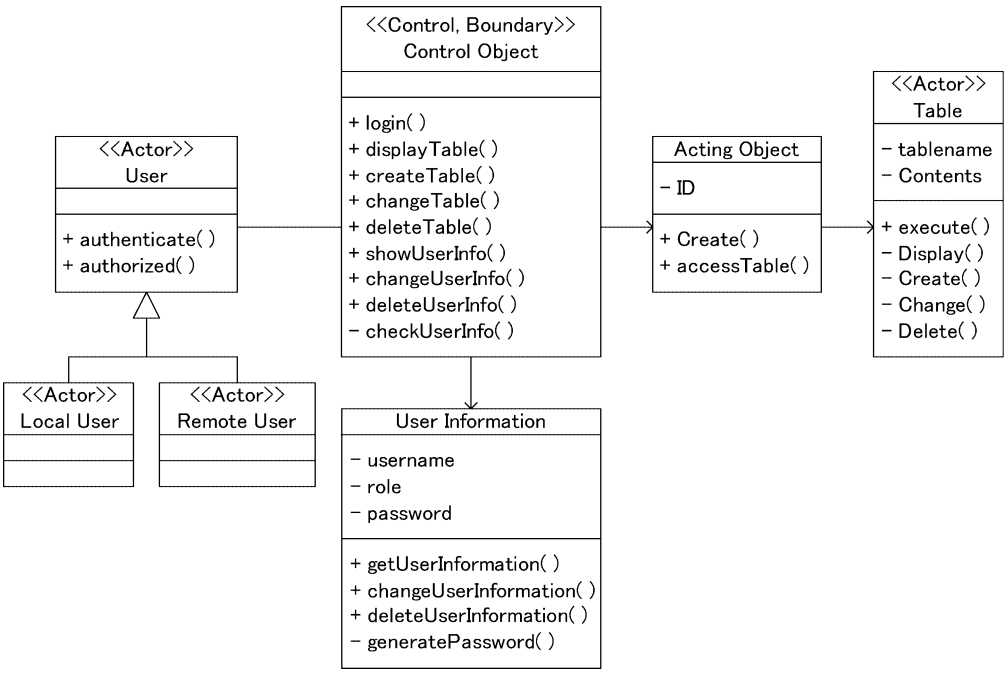

図 7　認証パターン適用後のクラス図
Fig. 7　The class diagram of the example after applying authentication pattern.

セスできなくなっている点である．クラス Table はクラス Acting Object のメソッド accessTable によってのみアクセスされる．データベースの表に対する表示，作成，変更，削除の要求は，図 6 で示すように accessTable の引数で操作の種類を渡すことにより，それぞれ実現している．

3.2　検 証 手 順

ISO/IEC15408 のセキュリティ機能要件は，抽象的な要件として記述されているため，システムの仕様が実際にそれらを満たしているかどうか，判断しがたく根拠も示しがたい．このため我々は，公開されている多数の認証済みセキュリティターゲット [7], [31] を調査して実際にセキュリティ機能要件が使われている実例から本質的な要件を解釈し，形式手法の検証の基準として利用できるようすべてのセキュリティ機能要件を形式化した [29]．動的な振舞い・時間的な制約については時相論理式 [5] で，静的・不変的な性質についてはソフトウェア信頼性の検証において実績のある Z 記法 [16], [26] で形式化した．我々は既に，これらの形式化した基準を用いて仕様が ISO/IEC15408 におけるセキュリティ機能要件を満たしているかどうか，形式的に検証する技法を提案した [21], [22], [35], [36]．

今回のモデルの検証にこの技法を用いるために，まず設計した UML を Z で形式化する必要がある．この形式化には，RoZ [11] を使用した．RoZ は，UML のクラス図と簡単な注釈を，Z で記述された仕様に変換するツールである．ただし，意味的な判断をせず機械的に変換するため，多少の修正は必要である．この RoZ を用いることにより，人手による恣意的な形式化を可能な限り防ぐ．また，Z に変換できない振舞い図についてはモデル検査 [5] で検証する．最近の研究では，モデル検査による UML の振舞い図の検証法が多数報告されている [19], [33]．

1. パターンを適用したクラス図を RoZ により Z に変換する．

2. Z で形式化したセキュリティ機能要件について定理証明で検証する．

3. パターンを適用した振舞い図をモデル検査ツール上で形式化する．

4. 時相論理で形式化したセキュリティ機能要件についてモデル検査で検証する．

以上のような手順で検証を行う．

3.3　定理証明による適用例の検証

3.1 の適用例が，前述の要求されている機能要件を

満たすかどうか，**3.2** の手順で検証する．以下は，図 7 のクラス図を RoZ によって Z 記法に変換し，一部修正した仕様である（紙面の都合上，検証の説明に必要な部分以外は付録に掲載した）．

仕様 z（Z 記法）

$[Char, Contents]$

$User ::= LocalUser \mid RemoteUser$

$Role ::= ordinary\_user \mid administrator$

$Operation ::= display \mid create \mid change \mid delete$

$Bool ::= TRUE \mid FALSE$

$String == \mathrm{seq}_1\ Char$

$Password == String;\ Name == String$

UserInformation

$username : User \pfun Name$

$password : User \pfun Password$

$role : User \pfun Role$

generatePassword

$\Delta UserInformation$

$u? : User$

$random\_password : Password$

$u? \in \mathrm{dom}\ password$

$username' = username$

$role' = role$

$password' = password \oplus \{(u? \mapsto random\_password)\}$

getUserInformation

$\Xi UserInformation$

$u? : User$

$output\_username! : Name$

$output\_password! : Password$

$output\_role! : Role$

$u? \in \mathrm{dom}\ username$

$u? \in \mathrm{dom}\ password$

$u? \in \mathrm{dom}\ role$

$output\_username! = username\ \ u?$

$output\_password! = password\ \ u?$

$output\_role! = role\ \ u?$

changeUserInformation

$\Delta UserInformation$

$u? : User$

$new\_username? : Name$

$new\_password? : Password$

$new\_role? : Role$

$u? \in \mathrm{dom}\ username$

$u? \in \mathrm{dom}\ password$

$u? \in \mathrm{dom}\ role$

$username' = username \oplus \{(u? \mapsto new\_username?)\}$

$password' = password \oplus \{(u? \mapsto new\_password?)\}$

$role' = role \oplus \{(u? \mapsto new\_role?)\}$

deleteUserInformation

$\Delta UserInformation$

$u? : User$

$u? \in \mathrm{dom}\ username$

$u? \in \mathrm{dom}\ password$

$u? \in \mathrm{dom}\ role$

$username' = \{u?\} \ndres username$

$password' = \{u?\} \ndres password$

$role' = \{u?\} \ndres role$

ControlObject

$Not\_login, Login : \mathbb{P}\ User$

$Not\_login \cap Login = \emptyset$

login

$login\_user?, u! : User$

$input\_username? : Name$

$input\_password? : Password$

$u! = login\_user?$

$LocalUserFlow \mathrel{\widehat=} login \gg getUserInformation$

$RemoteUserFlow \mathrel{\widehat=} login \gg generatePassword \fatsemi login \gg getUserInformation$

checkLocalUserInfo

$LocalUserFlow$

$judge! : Bool$

$login\_user? = LocalUser$

if $input\_username? = output\_username! \land input\_password? = output\_password!$

then $judge! = TRUE$ **else** $judge! = FALSE$

checkRemoteUserInfo

$RemoteUserFlow$

$judge! : Bool$

$login\_user? = RemoteUser$

if $input\_username? = output\_username! \land input\_password? = output\_password!$

then $judge! = TRUE$ **else** $judge! = FALSE$

$checkUserInfo \mathrel{\widehat=} checkLocalUserInfo \lor checkRemoteUserInfo$

authorized

$\Delta ControlObject$

$checkUserInfo$

$login\_user? \in Not\_login$

if $judge! = TRUE$

then $Login' = Login \cup \{login\_user?\}$

else $Login' = Login$

Table

$tablename : Name$

$contents : Contents$

Display
Ξ*Table*
. . .

Create
Δ*Table*
. . .

Change
Δ*Table*
. . .

Delete
Δ*Table*
. . .

execute
Display; *Create*; *Change*; *Delete*
kind? : *Operation*
---
if *kind?* = *display* **then** *Display*
else if *kind?* = *create* **then** *Create*
else if *kind?* = *change* **then** *Change*
else if *kind?* = *delete* **then** *Delete*
else Ξ*Table*

accessTable
execute
argument? : *Operation*
---
execute[*argument?*/*kind?*]

RoZ では，クラス図上のクラスとメソッドをそれぞれ Z の表記法であるスキーマとして変換する．スキーマとは，右側の線のない長方形で囲われたもので，システムの状態空間や操作を示す．この仕様 z について簡単に説明する．

冒頭の宣言部は，この状態空間には *Char*，*Contents*，*User*，*Role*，*Operation*，*Bool*，*String*，*Password*，*Name* という型（集合）が存在することを示している．RoZ は，Actor クラスであるクラス User もスキーマとして変換していたが，ユーザは実装の対象外であるので，この状態空間に存在する集合として与えるよう修正した．*User* は *LocalUser* か *RemoteUser* を取り得る．同様に，*Role*，*Operation*，*Bool* も列挙された値のいずれかをとる．*String*，*Password*，*Name* は *Char* 型のシーケンスである．クラス *UserInformation* に対応するスキーマ *UserInformation* は，要素として *username*，*password*，*role* をもつ．これらはそれぞれ *User* と，*Name*・*Password*・*Role* の全域関数である．クラス *UserInformation* のメソッドに対応するスキーマ *generatePassword* は，ユーザ *u*?に対応するパスワードを，生成したランダムパスワード *random_password* に置き換えている．スキーマ *getUserInformation*，*changeUserInformation*，*deleteUserInformation* は，クラス *UserInformation* のメソッドに対応し，それぞれユーザ *u*? に対応するユーザ名・パスワード・ロールの情報の取得，変更，削除を定義している．クラス *ControlObject* に対応するスキーマ *ControlObject* では，ユーザのログイン状態を表す *Login* と，非ログイン状態を表す *Not_login*（*User* の集合）を定義している．スキーマ *login* は，ログインしてきたユーザ *login_user*?にユーザ名 *input_username*? とパスワード *input_password*?の入力を促す操作を定義している．*u*!は，*login_user*?をパイピング [34] により，他のスキーマに渡すために便宜上定義した変数である．*LocalUserFlow* は，*login* でログインしてきたユーザの情報を *getUserInformation* から取得することをパイピングにより定義している．同様に，*RemoteUserFlow* は，まず *generatePassword* によりランダムパスワードを生成し，その後ログインしてきたユーザの情報を *getUserInformation* から取得することを定義している．*checkLocalUserInfo* は，ローカルユーザがログインしたときの入力されたユーザ名とパスワードが正しいかどうかチェックし，正しければ *TRUE* を，正しくなければ *FALSE* を出力する．同様に，*checkRemoteUserInfo* は，リモートユーザがログインしたときの入力情報のチェックを行う．RoZ は，クラス *ControlObject* のメソッド *checkUserInfo* を，一つのスキーマとして変換していたが，選択的にどちらかのフローをとる，ということを分かりやすく表すために，二つのスキーマの離接 [26] として定義するよう修正した．スキーマ *authorized* は，*checkUserInfo* の結果が *TRUE* であればユーザを *Login* 状態にし，*FALSE* であればログインを許可しない．クラス Table は Actor クラスであり，仕様 n より既存のデータベースの機能を利用するとあるので，その詳細は使用するデータベースに依存する．よってスキーマ *Table* の操作 *Display*，*Create*，*Change*，*Delete* については，スキーマ *Table* を操作するスキーマだということだけを定義し，詳細な形式記述については省略した．スキーマ *execute* は，入力 *kind*?によってスキーマ *Display*，*Create*，*Change*，*Delete* を使い分ける操作を定義している．クラス ActingObject のメソッドに対応するスキーマ *accessTable* は，間接的にスキーマ *execute*

を呼び出すことを定義している．

この仕様 z が，要求されているセキュリティ機能要件を満たすかどうか，前述の技法で検証する．以下は，我々が形式化した ISO/IEC15408 のセキュリティ機能要件のうち，FIA_UAU の定理である．

定理 FIA_UAU

$$\forall TSF' \mid users \bullet authentication\_mechanisms \land condition\_for\_authentication \bullet applied\_rules \land authorized\_state$$

$$\forall TSF' \mid users \bullet authentication\_mechanisms \land \neg\ condition\_for\_authentication \bullet applied\_rules \land not\_authorized\_state$$

プライムの付いたスキーマは，何らかの操作によって変化した後のスキーマを表す．上記の二つの式のうち，上の式はシステムのセキュリティ管理機能 *TSF* により，利用者 *users* が認証メカニズム *authentication_mechanisms* によって，認証される条件 *condition_for_authentication* を満たした場合，適用ルール *applied_rules* に従って認証される（*authorized_state* になる）ことを意味する．下の式は，逆に認証される条件 *condition_for_authentication* を満たしていない場合は，認証されないことを意味する．

これらの定理は，いかなる仕様においても利用できるよう汎用性をもたせた雛形として定義してあるので，使用する際に検証対象の仕様に適合するよう，それぞれの構成要素を ISO/IEC15408 の定義に従って具体化しなければならない．仕様 z では，*TSF* は認証を司る *ControlObject* に対応するため，上記の式の *TSF* の部分を *ControlObject* に置き換える．同様に，**3.1** の適用例の認証メカニズム *authentication_mechanisms* は，パスワード認証と単一使用認証である．認証パターンでは，認証する機能自体はどちらの場合も同じ機能を使い，特に単一使用認証のときはワンタイムパスワードを生成する，というメカニズムをとったため，ローカルユーザ，リモートユーザいずれの場合も *authentication_mechanisms* は *checkUserInfo* となる．認証される条件 *condition_for_authentication* は，*checkUserInfo* の出力 *judge!*が真 *TRUE* となる場合である．*applied_rules* は，ローカルユーザの場合は保持してあるパスワードは変化せず，リモートユーザの場合はスキーマ *generatePassword* によってパスワードが変化するので，それぞれ *password' = password* と *password' ≠ password* となる．認証された状態 *authorized_state* は，あるユーザを u とすると，変化後のユーザ u' が集合 *Login* に含まれることが条件となる．それぞれ定理 FIA_UAU と対応する部分を置き換えると，以下のような式になる．

$$\forall\ ControlObject' \mid \forall\ u : User \mid u = LocalUser \bullet checkUserInfo \land judge! = TRUE \bullet password' = password \land u' \in Login$$

$$\forall\ ControlObject' \mid \forall\ u : User \mid u = RemoteUser \bullet checkUserInfo \land judge! = TRUE \bullet password' \neq password \land u' \in Login$$

$$\forall\ ControlObject' \mid \forall\ u : User \mid u = LocalUser \bullet checkUserInfo \land \neg\ judge! = TRUE \bullet password' = password \land u' \notin Login$$

$$\forall\ ControlObject' \mid \forall\ u : User \mid u = RemoteUser \bullet checkUserInfo \land \neg\ judge! = TRUE \bullet password' \neq password \land u' \notin Login$$

上記の四つの式のうち，上二つの式は正しいユーザ名とパスワードを入力すると，ローカルユーザ，リモートユーザそれぞれが FIA_UAU で定めた認証メカニズムの適用ルールに従って正しく認証される，ということを意味し，下二つの式は正しいユーザ名とパスワードを入力しない場合は認証されないことを意味する．このように具体化した定理を，Z/EVES [25] により検証する．Z/EVES は，GUI による Z のエディタ機能が充実しており，またインタラクティブな定理証明機能による強力な検証ツールである．Z/EVES では，Z で記述した仕様と，Z における法則や公理を前提条件とし，入力した定理が導出できるかどうかを判定する．定理を導出できた場合には *true* を出力する．*true* が出力されるということは，その仕様上で入力した定理が成り立つということである．上記の四つの定理の場合は，*true* が出力される．つまり仕様 z 上で定理 FIA_UAU が成り立つということである．

以下は，FIA_USB の定理である．

定理 FIA_USB

$$\forall \Delta Resource \bullet a\_resource\_access\_operation$$

Δ はスキーマの変化を示す（Δ*Schema* は変化前 *Schema* と変化後 *Schema'* を含む）．この式は，システム上のリソース *Resource* のすべての変化はサブジェクトの操作 *a_resource_access_operation* によって起こることを意味する．つまり，*Resource* は *a_resource_access_operation* によってのみアクセス

される，ということである．仕様 z 上では，*Resource* は *Table* に，*a_resource_access_operation* は *accessTable* に対応するため，具体化した定理は以下のようになる．

$$\forall \Delta Table \bullet accessTable$$

この定理についても Z/EVES により *true* が出力される．*Table* は *ActingObject* の *accessTable* によってのみ変化させられる，つまり仕様 z は定理 FIA_USB を満たす，ということである．

以下は，FMT_MSA の定理である．

定理 FMT_MSA

$\exists$ *query_function* $\bullet \Xi$ *security_attributes*
$\exists$ *modify_function* $\bullet \Delta$ *security_attributes*
$\exists$ *delete_function* $\bullet \Delta$ *security_attributes*

この定理は，セキュリティ属性 *security_attributes* を問合せ・変更・削除する各管理機能が確かに存在することを意味する．仕様 z 上では，*security_attributes* はセキュリティ属性をもっている *UserInformation* に，*query_function*，*modify_function*，*delete_function* は *UserInformation* がもつ各セキュリティ属性管理機能に該当する．よって，具体化した定理は以下のようになる．

$\exists$ *getUserInformation* $\bullet \Xi$ *UserInformation*
$\exists$ *changeUserInformation* $\bullet \Delta$ *UserInformation*
$\exists$ *deleteUserInformation* $\bullet \Delta$ *UserInformation*

これらの定理についても，*true* が出力される．セキュリティ属性 *UserInformation* の問合せ *getUserInformation*，変更 *changeUserInformation*，削除 *deleteUserInformation* の機能が存在する，つまり仕様 z が定理 FMT_MSA を満たすことを検証できた．

3.1 の例の場合，クラス User Information が指定されたセキュリティ属性をもつため，FIA_ATD を満たすことは自明である．よって，検証は省略する．

3.4 モデル検査による適用例の検証

FIA_UAU の要件については，シーケンス図上で表現したが，Z のスキーマによっても表現可能であったため，定理証明によって検証することができた．一方で，FIA_UID については，シーケンス図にのみ表れる動的な要件であり，Z で表現することが困難なため，モデル検査により検証を行う．今回の例の場合，検証対象の仕様が単純なため，システムの仕様を最も単純に，自然に，分かりやく記述できるとされているモデル検査ツール NuSMV [4], [23] を用いた [2]．NuSMV は，状態遷移系として記述した仕様上で，時相論理式として記述した動的な振舞いが成り立つかどうか検証できる．以下の記述は，図 4 のステートマシン図を参考に，**3.1** の例を状態遷移系として NuSMV 上で記述したものである．

```
MODULE main
VAR
  input_username:boolean;
  input_password:boolean;
  User:{Login,Not_login};
  operation:{nothing,login,display,
                    create,change,delete};
DEFINE
  CheckUserInfo:=
          input_username & input_password;
ASSIGN
--input_username の遷移系
  init(input_username):={0,1};
  next(input_username):={0,1};
--input_password の遷移系
  init(input_password):={0,1};
  next(input_password):={0,1};
--User の遷移系
  init(User):=Not_login;
  next(User):= case
    operation!=login     :User;
    CheckUserInfo :Login;
    1                    :User;
    esac;
--operation の遷移系
  init(operation):=nothing;
  next(operation):= case
    User=Login      :{nothing,display,
                      create,change,delete};
    User=Not_login :{nothing,login};
    1               :nothing;
    esac;
```

NuSMV では，init にそれぞれの変数の初期状態を記述し，next にそれぞれ変数がどのように変化していくかの遷移を記述する．ユーザ名とパスワードを

表す変数 *input_username* と *input_password* は，正しいものが入力されるとは限らないので，非決定的に 0（正しくない）か 1（正しい）をとる．「非決定的にとる」とは任意のタイミングでいずれかの値をとる，ということを意味する．ユーザの状態を表す変数 *User* は，認証前状態 *Not_login* と認証後状態 *Loign* をとる．初期状態は *Not_login* で，以後は *operation* = *login* のときに *CheckUserInfo* が真である，つまりユーザがログインを試みたときに，入力された情報が正しければ *Login* に遷移し，それ以外の場合は現状維持とする．ユーザが行える操作を表す変数 *operation* は，*User* が状態 *Not_login* では何もしない *nothing* かログインを試みる *login* をとり，状態 *Login* では何もしない *nothing*，表の表示，作成，変更，削除に該当する *display*，*create*，*change*，*delete* を非決定的にとる．

FIA_UID は，システムにおけるあらゆるアクションよりも前に，ユーザ認証を行わなければならない，という要件であった．つまり，ユーザは正しく認証されるまで，いかなるアクションも実行できない，ということである．この条件を，LTL の時相論理式で記述すると，以下のようになる．

定理 FIA_UID

$$\Box\ (\neg\ \mathit{any\_action\_occur}\ \mathrm{U}\ (\mathit{condition\_for\_authentication} \rightarrow \mathrm{X}\ \mathit{authorized\_state}))$$

時相論理式 □ p は，遷移系で常に p が成り立つことを，p U q は，q が成立するまで p が成り立つということを，p → X q は，p が成り立つならば次に q が成り立つことを意味する [5]．つまり上記の式は，常に，ユーザが認証される条件 *condition_for_authentication* を満たし認証状態 *authorized_state* になるまで，あらゆるアクション *any_action_occur* が成立しないという意味である．このように，FIA_UID のような時間的な制約を表すセキュリティ機能要件については，LTL で形式化した．

前述の NuSMV のコード上では，認証される条件 *condition_for_authentication* は，ログインを試みる，つまり *operation* が *login* をとり，かつ正しいユーザ名とパスワードが入力される，つまり *input_username* と *input_password* が真となることであり，認証状態 *authorized_state* は *User* が *Login* に遷移することである．また，何らかのアクションが起こる *any_action_occur* とは，ログイン以外の操作をとる，つまり *operation* が *display, create, change, delete* となる場合である．つまり具体化した式は以下のようになる．

$$\Box\ (\neg\ (\mathit{operation}{=}\mathit{display} \vee \mathit{operation}{=}\mathit{create} \vee \mathit{operation}{=}\mathit{change} \vee \mathit{operation}{=}\mathit{delete})\ \mathrm{U}\ ((\mathit{CheckUserInfo} \wedge \mathit{operation}{=}\mathit{login}) \rightarrow \mathrm{X}\ \mathit{User}{=}\mathit{Login}))$$

上記の式は，NuSMV により *true* と判定される．FIA_UID を満たすことは，図 6 からほぼ自明であるが，モデル検査により形式的に検証することができた．

以上のように，認証パターンを用いて設計した例は，確かに要求されているすべてのセキュリティ機能要件を満たすことが検証できた．

3.1 で示したように，プロテクションプロファイルをモデル化することにより，効果的に利用することができる．その結果，ISO/IEC15408 で定義されたセキュリティ基準を満たしたシステムの設計が可能である．これは，本節の例と同様，**3.2** で示した手順で，我々が形式化したセキュリティ機能要件 [29] を用いて一般的な定理証明技法とモデル検査技法によって形式的に検証することが可能である．

4. 関 連 研 究

UML とセキュリティに関して，以下のような先行研究がある．SecureUML [20] では RBAC（Role Based Access Control）に，文献 [9] では MAC（Mandatory Access Control）に基づくアクセス制御によりセキュリティを保証する．authUML [1] では，ユースケース図のみにおいてアクセス制御によりセキュリティを保証する．UMLsec [14], [15] では，UML の記法を拡張し独自に形式的なセマンティクスを与えることでセキュリティを保証する．これらの表記や検証は，ツールに依存する．また，これらは外部の客観的な基準によるセキュリティを保証するものではない．

これらと比較し，本論文で提案したパターンは，表記・検証ともにツールに依存しない．また，ISO/IEC15408 という現存する一般的・客観的な基準，つまり国際標準によるセキュリティを保証することを目標とした．開発者は，利用者に対して，システムにおいてどのようなセキュリティ対策が施されているのかを，利用者が理解しやすい形で具体的に示すことができる．更に，ISO/IEC15408 には，大きく分けて 11 分野のセキュ

リティ機能要件が定義されており，アクセス制御はこれらのうちのほんの一部でしかない．セキュリティ監査・通信・暗号サポート・利用者データ保護・資源利用・セキュリティ管理・プライバシー等といった幅広い範囲のセキュリティを提供することが可能である．

5. 考　　察

本論文で提案したモデル化により，プロテクションプロファイルを機能的な単位ごとに分割し，定めた方針に基づいて必要なセキュリティ機能要件を識別した後，UML によるパターンとして提供するため，必要な機能ごとに必要な部分のみの利用が容易になり，解釈も統一される．加えて，モデル化したプロテクションプロファイルを使用した仕様記述・設計は，ISO/IEC15408 によりセキュリティが保証されている．これは本論文で示した検証の手順によって，形式的に検証することができる．

しかしながら，プロテクションプロファイルをモデル化するには，その他の数多くのプロテクションプロファイルを参照し，それぞれの分野や機能の相互関係やセキュリティを保証することを考慮に入れながらモデル化しなければならないため，時間と労力を要する．この問題を解決するために，我々は様々な分野におけるプロテクションプロファイルと要求されているセキュリティ機能要件の依存関係を調べ，それらを格納しプロテクションプロファイルのモデル化等に利用できるようにするデータベースを提案した [37].

また，検証を行う場合，検証対象の形式化が課題となる．提案したパターンを用いて設計したシステムを開発者自らが形式化しなければならないが，どう形式化してよいか分からないということもあり得る．また，形式化する際の解釈や記述の相違により検証が困難になってしまう可能性がある．このため本研究では，明確で読みやすく直感的に理解しやすい，人間が使用する際の使いやすさが考慮されているという点で，Z 記法を採用した [34]．Z では，集合論や一階述語論理に基づき，対象システムの仕様をシステムの状態と操作という観点からとらえ，これらの状態と操作をスキーマとして表現する．このスキーマや集合を用いて，基準を一般化し雛形として形式化しやすいという点も，Z を採用した理由の一つである．また，提案したパターンは UML で表現されているため，RoZ を用いて形式化することができる．RoZ を使用することで，形式化の困難さの軽減や，形式化した結果をある程度統一することを試みた．Z で記述，検証できない部分については，モデル検査ツールで記述・検証した．UML のステートマシン図は，ほぼ機械的に状態遷移系として形式化できる．また，静的側面のみを Z で，動的側面のみをモデル検査ツールでそれぞれ形式化することで，それぞれの側面の形式化を単純にし，また正確に形式化することを試みた [36].

6. む　す　び

本論文では，情報システムのセキュリティ基本設計書の雛形であるプロテクションプロファイルを，想定される脅威ごとに分割し，UML によるパターンとしてモデル化することを提案した．また，モデル化したプロテクションプロファイルによる仕様記述・設計が，セキュリティ基準を満たすことを形式手法を用いて検証する手順を示した．更に検証例を示すことにより，正しくモデル化できていることを実証した．

しかしながら，モデル化したプロテクションプロファイルを使用するためには，UML の知識が不可欠である．よって我々は現在，モデル化とその利用を簡単にするための機能や，形式化と検証をサポートする機能を併せ持つツールを開発している．

また近年，PVS や Expander，Agda，μCRL など，ハイブリッドな検証ツールが提案されており，様々な検証環境において定理証明とモデル検査の環境が統合されつつある [28]．このような現状を受け，我々は現在，形式化した ISO/IEC15408 の評価基準をどのような検証環境においても利用できるよう検討している．本論文では，Z と時相論理式により記述した ISO/IEC15408 の評価基準を Z/EVES と NuSMV により検証したが，我々が形式化した基準はツールに依存しないため，Z と時相論理式が扱えるその他のツールでも検証することが可能である．また，Z は一階述語論理で表現されており，また時相論理式は主要なモデル検査ツールで検証が可能である．以上のような点を踏まえ，形式化した基準の拡張を行っている．

加えて，我々は公開されている認証済みセキュリティターゲット [7], [31] やプロテクションプロファイル [8] を多数検証することで形式化した ISO/IEC15408 の基準の正しさを確認しているが，今後も引き続きこの作業を行うことで形式化した基準の妥当性・正当性を実証，洗練していく．そして，更に多くのモデル化を行って，それらをより実規模な問題に適用し有効性を実証する．

謝辞 本研究の草分けとしてISO/IEC15408 Part2セキュリティ機能要件の形式化に従事して下さった重松真二郎氏，並びに本論文の原稿について貴重な御意見を下さった査読委員の方々，編集委員の方々に深く感謝致します．

文　献

[1] K. Alghathbar and D. Wijesekera, "authUML: A three-phased framework to analyze access control specifications in use cases," Proc. 2003 ACM Workshop on Formal Methods in Security Engineering (FMSE'03), pp.77–86, Washington D.C., USA, 2003.

[2] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, and P. McKenzie, Systems and Software Verification - Model-Checking Techniques and Tools, Springer-Verlag, 1999.

[3] G. Booch, R.A. Maksimchuk, M.W. Engel, A. Brown, J. Conallen, K.A. Houston, R. Martin, and J.W. Newkirk, Object-Oriented Analysis and Design with Applications, 3rd ed., Addison Wesley Professional, 2004.

[4] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: A new symbolic model verifier," Proc. Computer Aided Verification: 11th International Conference CAV'99, pp.495–499, Trento, Italy, 1999.

[5] E. Clarke, O. Grumberg, and D. Peled, Model Checking, MIT Press, 2000.

[6] Common Criteria Org, "Common criteria for information technology security evalution version 2.2 revision 256," 2004.

[7] Common Criteria Org, "Evaluated product files," http://www.commoncriteriaportal.org/public/files/epfiles/

[8] Common Criteria Org, "Protection profile files," http://www.commoncriteriaportal.org/public/files/ppfiles/

[9] T. Doan, S. Demurjian, T.C. Ting, and A. Ketterl, "MAC and UML for secure software design," Proc. 2004 ACM Workshop on Formal Methods in Security Engineering (FMSE'04), pp.75–85, Washington DC, USA, 2004.

[10] K. Dolan, P. Wright, R. Montequin, B. Mayer, L. Gilmore, and C. Hall, U.S. Department of Defense Traffic-Filter Firewall Protection Profile for Medium Robustness Environments, National Security Agency, 2001.

[11] S. Dupuy, Y. Ledru, and M. Chabre-Peccoud, "An overview of RoZ: A tool for integration UML and Z specifications," Proc. 12th Conference on Advanced information Systems Engineering (CAiSE'2000), pp.417–430, Stockholm, Sweden, 2000.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object Oriented Software, Addison Wesley Longman, 1998.

[13] N. Haller and C. Metz, "A one-time password system," RFC 1938, Internet Engineering Task Force, 1996.

[14] J. Jürjens, "UMLsec: Extending UML for secure systems development," Proc. UML 2002 — The Unified Modeling Language : 5th International Conference, pp.412–425, Dresden, Germany, Springer-Verlag, 2002.

[15] J. Jürjens, Secure Systems Development with UML, Springer-Verlag, 2005.

[16] ISO/IEC 13568 Standard, "Information technology — Z formal specification notation — syntax, type system and semantics," 2002.

[17] ISO/IEC 15408 Standard, "Information technology — security techniques — evaluation criteria for IT security," 1999.

[18] ISO/IEC 15292 Standard, "Information technology — security techniques — protection profile registration procedures," 2001.

[19] D. Latella, I. Majzik, and M. Massink, "Automatic verification of a behavioural subset of UML statechart diagrams using the spin model checker," Formal Aspects of Computing, vol.11, no.6, pp.637–664, 1999.

[20] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-based modeling language for model-driven security," Proc. UML 2002 — The Unified Modeling Language : 5th International Conference, pp.426–441, Dresden, Germany, Springer-Verlag, 2002.

[21] S. Morimoto, S. Shigematsu, and J. Cheng, "A formal method for verifying security specifications based on international standard ISO/IEC 15408," Supplement of the IEEE-CS 2005 International Conference on Dependable Systems and Networks, pp.62–63, Yokohama, Japan, 2005.

[22] S. Morimoto, S. Shigematsu, Y. Goto, and J. Cheng, "A security specification verification technique based on the international standard ISO/IEC 15408," Proc. 21st Annual ACM Symposium on Applied Computing, Dijon, France, 2006.

[23] NuSMV, http://nusmv.irst.itc.it/

[24] Object Management Group, "UML 2.0 Superstructure Specification," 2004.

[25] ORA Canada, "Z/EVES," http://www.ora.on.ca/z-eves/welcome.html

[26] B. Potter, J. Sinclair, and D. Till, An Introduction to Formal Specification and Z, 2nd ed., Prentice-Hall, 1996.

[27] H. Smith, J. DeMello, and S. Pannifer, Database Management System Protection Profile, Oracle Corporation, 2000.

[28] YAHODA, "Verification tools database," http://anna.fi.muni.cz/yahoda/

[29] 埼玉大学大学院理工学研究科情報数理科学専攻先端情報システム工学研究室，“ISEDS：ISO/IEC15408 セキュリティ機能要件の形式記述,” http://www.aise.ics.saitama-u.ac.jp/
[30] （財）インターネット協会，インターネット白書 2005，インプレス，2005.
[31] 独立行政法人情報処理推進機構，“認証製品リスト,” http://www.ipa.go.jp/security/jisec/cert_list200504.html
[32] （独）情報処理推進機構，“情報技術セキュリティ評価のためのコモンクライテリアパート 2: セキュリティ機能要件,” http://www.ipa.go.jp/security/jisec/evalbs.html
[33] 中島　震，“オブジェクト指向ソフトウェアのモデル検査,” 第 2 回ディペンダブルソフトウェアワークショップ DSW2005 論文集，日本ソフトウェア科学会研究会資料シリーズ，no.35, pp.1–12, 2005.
[34] 水野忠則，プロトコル言語，カットシステム，1994.
[35] 森本祥一，重松真二郎，程　京徳，“ISO/IEC15408 の形式化に基づく情報セキュリティ仕様の形式的検証法,” 先進的計算基盤システムシンポジウム SACSIS2005 論文集，IPSJ Symposium Series, vol.2005, no.5, pp.201–202, 2005.
[36] 森本祥一，重松真二郎，後藤祐一，程　京徳，“ISO/IEC 15408 に基づく定理証明とモデル検査を用いた情報セキュリティ仕様の検証技法,” 第二回システム検証の科学技術シンポジウム予稿集，pp.12–23, 2005.
[37] 森本祥一，堀江大輔，程　京徳，“ISO/IEC15408 に基づく情報セキュリティ要求管理データベース,” 日本データベース学会論文誌 DBSJ Letters, vol.4, no.3, pp.13–16, 2005.

付　　録

形 式 仕 様

本文中で省略した形式仕様を以下に示す.

changeUserInfo

$ControlObject$
$changeUserInformation$
$change\_user? : User$

$change\_user? \in Login$
$changeUserInformation[change\_user?/u?]$

deleteUserInfo

$ControlObject$
$deleteUserInformation$
$delete\_user? : User$

$delete\_user? \in Login$
$deleteUserInformation[delete\_user?/u?]$

showUserInfo

$ControlObject$
$getUserInformation$
$display\_user? : User$
$display\_username! : Name$
$display\_password! : Password$
$display\_role! : Role$

$display\_user? \in Login$
$getUserInformation[display\_user?/u?]$
$display\_username! = output\_username!$
$display\_password! = output\_password!$
$display\_role! = output\_role!$

display

$input\_operation?, argument! : Operation$

$input\_operation? = display$
$argument! = input\_operation?$

create

$input\_operation?, argument! : Operation$

$input\_operation? = create$
$argument! = input\_operation?$

change

$input\_operation?, argument! : Operation$

$input\_operation? = change$
$argument! = input\_operation?$

delete

$input\_operation?, argument! : Operation$

$input\_operation? = delete$
$argument! = input\_operation?$

$displayTable \mathrel{\widehat=} display \gg accessTable$

$createTable \mathrel{\widehat=} create \gg accessTable$

$changeTable \mathrel{\widehat=} change \gg accessTable$

$deleteTable \mathrel{\widehat=} delete \gg accessTable$

ActingObject

$ID : \mathbb{N}$

（平成 17 年 7 月 15 日受付，11 月 4 日再受付）

森本　祥一　（正員）

平 11 埼玉大・工・情報システム卒．平 13 同大大学院理工学研究科博士前期課程了．同年日本電気航空宇宙システム株式会社入社．平 15 同社退職．平 18 埼玉大大学院理工学研究科博士後期課程情報数理科学専攻了，博士（工学）．ソフトウェア工学，情報セキュリティ工学に関する研究に従事．情報処理学会，日本ソフトウェア科学会，日本データベース学会，ACM 各会員．

程　　京徳

昭 57 中国清華大学計算機科学技術系卒．平元九州大大学院工学研究科博士後期課程情報工学専攻了，工博．同年同大学工学部情報工学科助手．平 3 同大学助教授．平 8 同大大学院システム情報科学研究科情報工学専攻教授．平 11 埼玉大大学院理工学研究科情報数理科学専攻教授．ソフトウェア工学，知識工学，情報セキュリティ工学に関する研究に従事．情報処理学会，日本ソフトウェア科学会，日本データベース学会，ACM，IEEE-CS，IEEE-SMC，IEEE 各会員．