修 士 論 文

モデル検査技術によるセキュリティ検証へのアスペクト指向技術の適用

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

小坂 浩之

2009 年 3 月

修 士 論 文

モデル検査技術によるセキュリティ検証へのアスペクト指向技術の適用

指導教官　岸 知二 特任教授

審査委員主査　岸 知二 特任教授
審査委員　落水 浩一郎教授
審査委員　青木 利晃 特任准教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

0710029 小坂 浩之

提出年月: 2009 年 2 月

概 要

システムで生じるべきでない挙動を分析する手法にミスユースケースという考え方がある．ミスユースケースとして考えられたシナリオは，ソフトウェアの設計を脅かす攻撃シナリオである．本研究では，ソフトウェアの設計を攻撃シナリオに耐えうる設計に変更するために，攻撃緩和のシナリオ (ミティゲーションシナリオ) をアスペクトとして定義し，システムのモデルにウィーブする体系的な手法を提案した．また，ウィーブ後のシステムが攻撃シナリオを防ぐことを検証した．

目 次

第1章　はじめに

1.1　研究背景

近年，ネットワークに接続する機器が増加していることもあり，ソフトウェアのセキュリティの重要性が高まっている．例えば，セキュリティ要求工学の分野ではセキュリティ要件の分析手法が研究されており，その中にシステムにとって望ましくない状況を記述する「ミスユースケース」という考え方が研究されている．ミスユースケースは，システムに害を及ぼすユースケースであり，通常，ミスユースケースが持つシステムへの脅威に対して，それを緩和するユースケースを考える．攻撃を緩和するユースケースはソフトウェアが満たすべきセキュリティ要件である．

また，システムを開発する上で，セキュリティ要件を守るようなソフトウェア設計 (以後，セキュアな設計と呼ぶ) を行うことは重要であり，ソフトウェア工学の分野でもセキュアな設計を行うための体系的な手法に注目が集まっている．そのような背景から，本研究ではセキュアな設計を行うための体系的な設計手法を提案する．

セキュアな設計を行うために必要なセキュリティに関する機能はセキュリティ関心事と呼ばれ，その代表例として，認証やアクセス制御などの機能が挙げられる．これらの機能はシステムを横断するような機能であることから，システムの機能と分離して考えるアスペクト指向の技術の適用が考えられる．これによって，モジュール性・再利用性の向上が期待できる．本研究では，ミスユースケースを緩和するミティゲーションの機能をアスペクトとしてモデル化し，システムにウィーブすることでセキュアな設計のための体系的な手法を提案する．

1.2　本研究の位置付け

現代社会においてセキュリティの問題を考えることは非常に重要であるが，セキュリティを考慮すべき範囲は広大であるため，全てを検討することは難しい．セキュリティには，システムの安全性を高めるための技術 (認証，アクセス制御) やシステムの潜在的脅威 (不正アクセス，ウィルス) などの特徴がある．これら全ての脅威から防御方法を考えることは容易ではなく，システム開発において体系的な方法が求められている．

本研究では，ソフトウェア工学の観点からシステムが持つ潜在的な脅威からシステムを守るための手法を検討する．ソフトウェアの設計手法に焦点を当ててセキュアな設計手法の検討を行う．

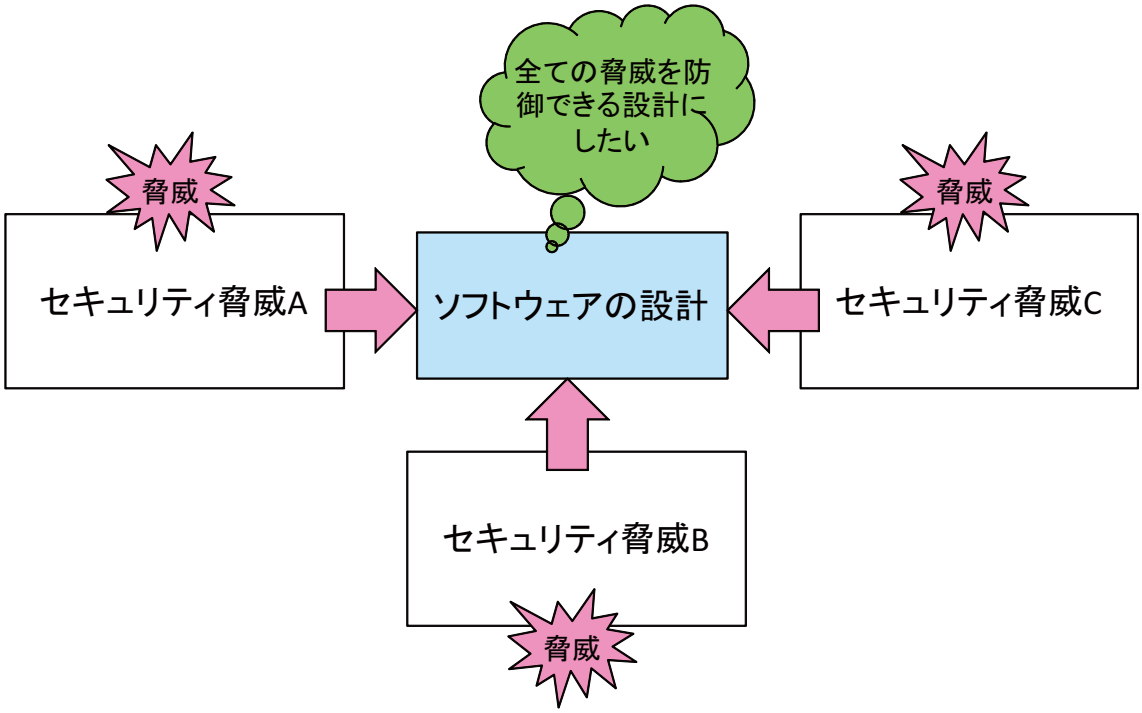


図 1.1: システムが持つ潜在的脅威のイメージ

1.3 論文の構成

本論文の構成は，２章で本研究を取り組むきっかけとなった関連技術を紹介し，３章で２章までに述べた内容を踏まえて本研究における目的を述べる．

４章ではセキュアな設計手法を提案するに当たっての設計思想を述べる．次に，提案手法と提案手法に用いるモデルについて述べる．

５章では本研究で提案するウィーブ手法について詳細な説明とその手順，また，ウィーブ手法を実装するに当たって適用した技術について述べる．

６章で，５章で提案したウィーバを電子投票システムの例題に適用させた例を紹介する．

７章で，本研究のまとめ，今後の課題を述べる．

第2章　関連技術

本章では，本研究を取り組むきっかけとなった関連技術及び、関連研究について述べる．

2.1　ミスユースケース

1章でも述べたように，ミスユースケース[1]とは否定的なアクターの観点から見たユースケースである．ミスユースケースはユースケースと相互に情報をやり取りし，ミスユースケース図に記述される．ミスユースケースは，システム開発時に利害関係者(stakeholder)の間で使用することができ，ミスユースケース図は脅威と危険要因の分析，システム設計，要求の抽出などに使われる．

図2.1はミスユースケースの記述例である．

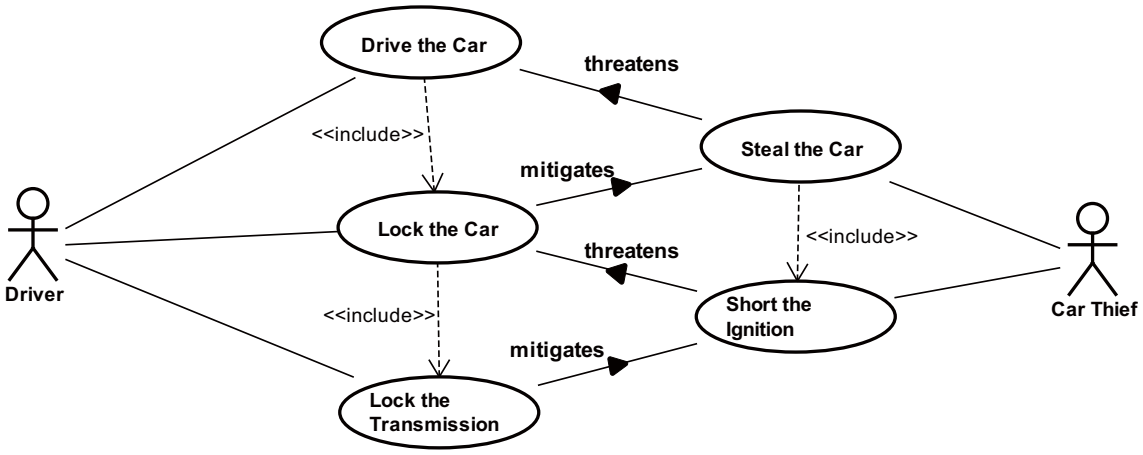


図 2.1: ミスユースケース図の例

ミスユースケースの考え方を提案しているIan Alexanderはミスユースケースとユースケースを一つの図で表している．図2.1から，”Driver”というアクターに対して，”Car Theif”という負のアクターの存在を明示的に理解することができる．また，左側の３つのユースケースに対して右側の２つのミスユースケースがあることがわかる．この図から，右側が左側を脅かす”threatens”という関係と，左側が右側を緩和する”mitigates”という

関係が成立し，ソフトウェアの開発時に，”mitigates”しなくていはいけない機能があることを示している．

2.2 関連研究

ここでは，セキュアな設計に関する体系的な手法を提案している関連研究 [2] を紹介する．

特徴

Jon Whittle らが提案している手法はミスユースケースを UML を用いた実行可能なモデルに変換し，作成した実行可能なモデルを使用して，セキュアな設計を行うための体系的な手法を提案していることが特徴である．

ミスユースケース自体は抽象度が高く理解しやすいが，それだけではテスト及び解析などに使うことが難しい．そこで，ミスユースケースがユースケースを脅かす「脅威」のシナリオを攻撃シナリオ，逆に「緩和」のシナリオをミティゲーションシナリオとして UML でモデル化し，ミティゲーションをシステムのシナリオにウィーブし，システムを攻撃に強い設計にする．そのセキュアな設計のための体系的な手法を提供している．

この研究に用いられている技術を以下に記述する．これらの技術は既存の技術であるが，さらに Jon Whittle が提案した手法を適用し拡張を行っている．

- ユースケースモデリング
- アスペクト指向モデリング
- シナリオからのステートマシン合成

ユースケースモデリングは，ユースケースから相互作用図 (Interaction Overview Diagram) を作成するために Jon Whittle が拡張した手法である．ユースケースを 3 レベルの階層で分析し，ユースケースのシナリオを IOD に変換する．

アスペクト指向モデリングは，一般的にはアスペクト指向技術をモデリング手法に適用させることである．ここではアスペクト指向モデリングにグラフの変形規則を適用した．

シナリオからのステートマシン合成は，シナリオからステートマシンに変換する手法である．

提案手法に用いられるモデルは，相互作用図 (Interaction Overview Diagram) を拡張した拡張相互作用図 (Extended Interaction Overview Diagram) とステートマシン図である．図 2.2 は，ミティゲーションのシナリオをユースケースのシナリオにウィーブし，ウィーブ後のシナリオをステートマシン図に変換した後に，システムのステートマシン図上で攻撃が成功しないことをシミュレーションで確認するまでの一連の流れを表している．また、シミュレータにおいてはミスユースケースシミュレータ (MisUse Case SIMulator) を

作成してシナリオ作成からステートマシン図をシミュレーションするまでの体系的な手法を提案している．

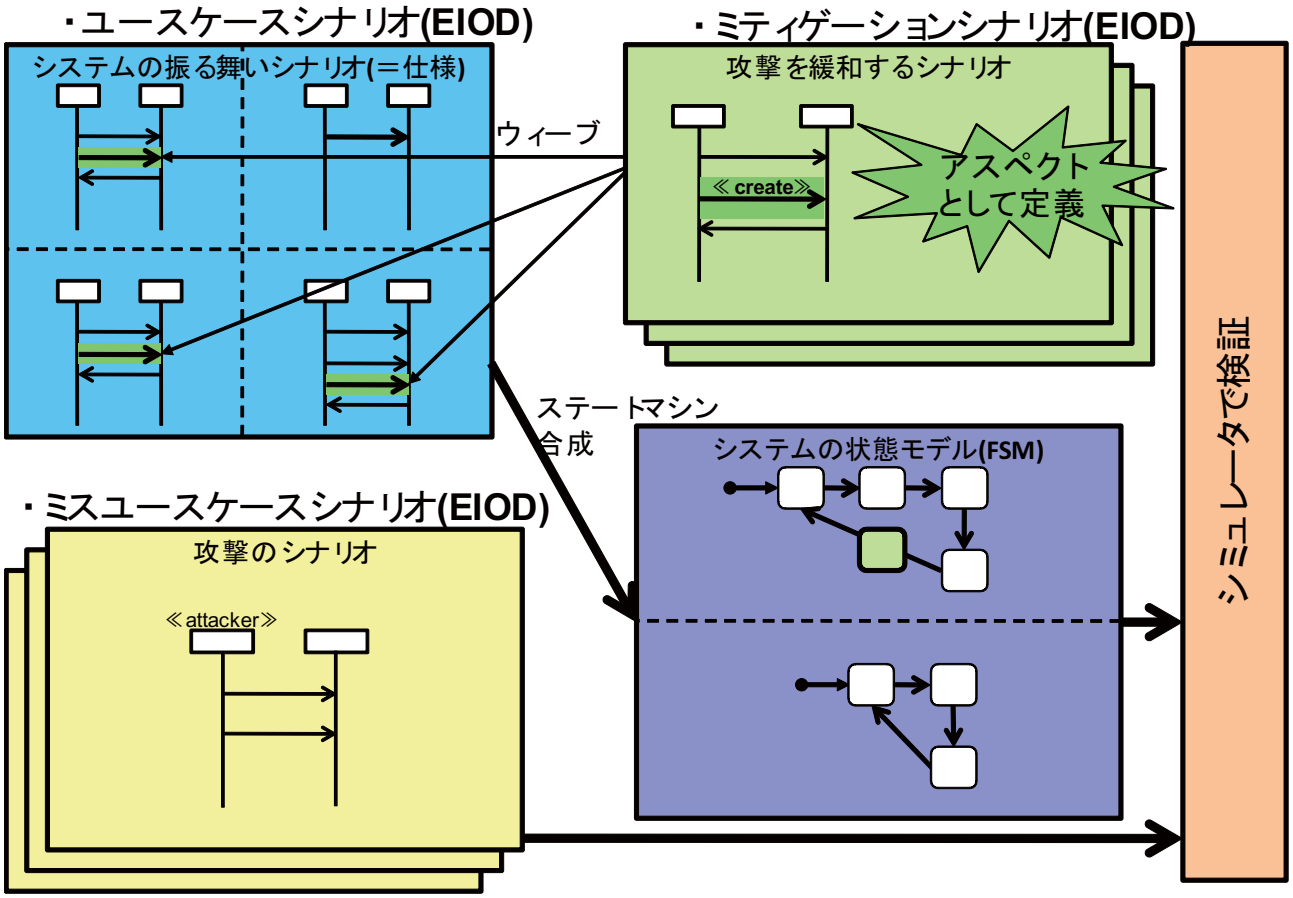


図 2.2: 関連論文の全体図

第3章　目的

本章では，関連研究の問題点に着目して，研究の目的を述べる．

3.1　本研究で解決したい問題点

関連研究では，ミスユースケースをミティゲーションシナリオとしてアスペクトで定義し，システムのシナリオにウィーブするセキュアな設計手法を提案していた．しかしこの手法だと，システムがシナリオレベルで定義されている場合のみにしか対応しないという問題点があった．

そこで，本研究ではよりセキュアなシステムを構築するために，システムの全ての振る舞いに対して，セキュアな設計が可能な手法を検討する．

本研究の目的を以下とする．

- システムの全ての振る舞いに対してセキュアな設計が可能な手法の考案

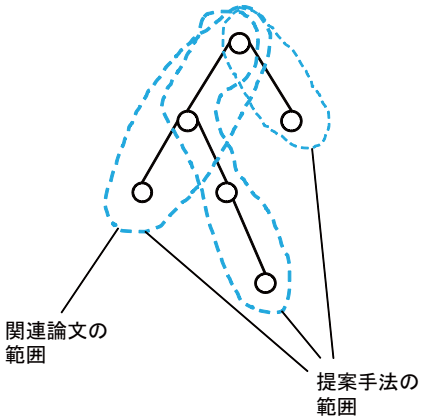


図 3.1: セキュアな設計の範囲のイメージ

図 3.1 は，関連研究の提案しているセキュアな設計手法の対象範囲と本研究で提案しようとしているシステムの全ての振る舞いに対応したセキュアな設計の対象範囲を示す．

第4章　セキュアな設計のための提案手法

本章では，セキュアな設計のための設計手法について述べるために，まず，既存技術のアスペクト指向技術について述べる．既存技術の特徴を踏まえた上で，提案する手法に必要な要件を述べる．そして，提案する手法の概要を説明し，提案するアスペクトモデルに関する定義，記法について述べる．

4.1　既存技術の紹介

4.1.1　アスペクト指向技術

一般的にアスペクト指向に関する議論には，横断的関心事の分離という概念が出てくる．ここでいう関心事とは，扱おうとしているシステム内に散在しているシステムの要素のことである．関心事を分離する理由は，システムの中心的な本体から関心事となる機能を分離させておくことで，システム自体の保守性，再利用性の向上が期待できるからである．

関心事として分離する機能の代表的なものに,「データの永続性」「ロギング」「セキュリティ」が挙げられる．

ここではロギングついて説明する (図 4.1)．どんなシステムにもログをとるという機能は重要である．しかし，ログをとるためには，ログをとりたいタイミングでログの機能を呼び出す必要があり，そのタイミングはシステム内の任意のモジュール中に存在する．ログをとる機能がシステム内の複数のモジュールを横断するため，システムの保守性、再利用性は低下する．このような複数のモジュールを横断するような機能を横断的関心事という．アスペクト指向は横断的関心事をいかに分離するかということに焦点を当てた考え方である．

4.1.2　アスペクト指向言語

次に，アスペクト指向技術を適用したアスペクト指向言語について述べる．代表的なものに AspectJ や JBossAOP などがある．これらの言語は JAVA で記述することを想定して作られたアスペクト指向言語で，分離された横断的関心事を，対象とするシステムにウィーブ (織り込む) ための機能を備えている．適切なウィーブが行われるためには，アスペクト側にどんな時に呼ばれるかという情報と，ウィーブされる側にどこにアスペクト

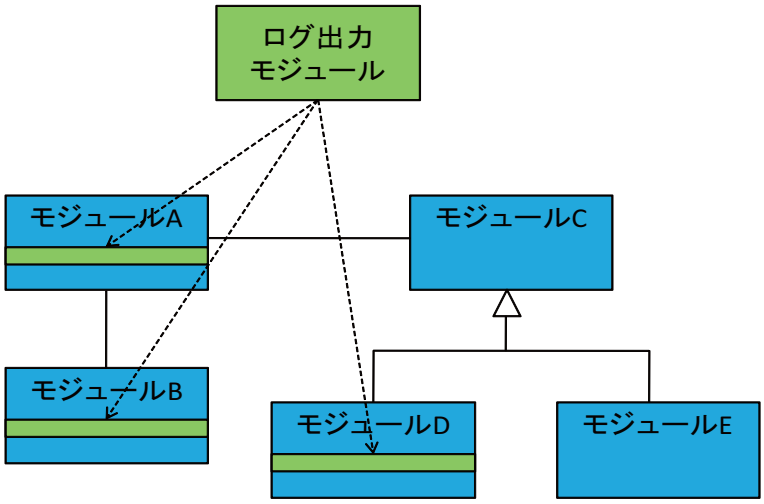


図 4.1: ロギングの例

が挿入可能かという情報を埋め込んでおく必要がある．そのための仕組みをジョインポイントモデルという．

4.1.3 ジョインポイントモデル

ジョインポイントモデルは以下の三つから構成される．

ジョインポイント

ジョインポイントは，アスペクトを織り込むことが可能なコード上のポイントである．プログラムの実行時にアドバイスの実行を織り込むことが可能なコード上の位置を指す．AspectJ や JBossAOP の場合，メソッドやコンストラクタの呼び出し位置などがジョインポイントとなる．

ポイントカット

条件が指定された時に，コード内に存在する全てのジョインポイント集合から抽出される部分集合である．

アドバイス

プログラムの実行が，ジョインポイントに差し掛かった時に，実行される機能のことである．実行のタイミングとしては，ジョインポイントの事前 (before)，事後 (after)，呼び出し時に代わりに実行 (around) がある．

4.2 セキュアな設計のための設計思想

ミスユースケースから抽出された脅威のシナリオは，対象とするシステム内の複数のモジュールに散在する可能性があるため，脅威を緩和するシナリオをセキュリティの横断的関心事と捉えて，システムのモデルとは別にアスペクトとして定義する．定義したアスペクトはセキュリティの関心時であり，システムにウィーブすることで，脅威を緩和することが可能なセキュアなシステムの設計が構築出来る．

提案手法に必要なモデルを以下に定義する．

システムモデル 複数のシナリオを包含するようなモデルとしてステートマシン図を考える．

攻撃モデル ミスユースケースから抽出されるシナリオを UML のシーケンス図でモデル化する．

ミティゲーションモデル (アスペクトモデル) セキュリティの関心事をシステムモデルにウィーブ可能なアスペクトとしてシーケンス図でモデル化する．

4.3 アスペクトモデルをシステムモデルにウィーブするための要件

脅威を緩和するシナリオ (シーケンス図) をアスペクトとして定義する際に用いる概念と，シナリオをステートマシン図にウィーブするための概念を要件として纏める．

要件 1 シーケンス図のアスペクトが記述可能な範囲を定義し，シーケンス図が持つ要素をジョインポイントモデルと対応付ける必要がある．

要件 2 シーケンス図はポイントカットとして指定可能な範囲が限定されてしまうため，ポイントカットの指定をシーケンスで扱えるようにする．

要件 3 ウィーブする際にシーケンス図であるアスペクトモデルの要素とステートマシン図であるシステムモデルの要素は一対一には対応づかない．そのため，シーケンス図の要素がステートマシン図に与える影響の範囲を調査する必要がある．

また，要件 1，2 は関連研究でも登場している概念だが，本研究でのウィーブ手法を提案するに当たって再定義している．要件 3 は，関連研究から全くのオリジナルとなる概念で，本研究のメインとなる部分である．

上記の要件に対して，手法の提案を行う．

4.3.1 要件 1 に関する提案

シーケンス図が表現可能な範囲について考え，アスペクトとしてのシーケンス図が記述可能な範囲を定義する．

また，シーケンス図が持つ要素とジョインポイントモデルとの対応付けを行い，提案手法のアスペクトが持つジョインポイントモデルを定義する．

4.3.2 要件 2 に関する提案

メッセージのシーケンスをポイントカットとして扱うためには，複数のメッセージがポイントカットであることを明示的に記す必要がある．

例えばメッセージ Msg1 とメッセージ Msg2 のシーケンスをポイントカットとしたいと考えた時に，シーケンスポイントカットの種類は大きく分けて 2 種類考えられる．

1. 連続したポイントカットのシーケンス (Msg1 → Msg2)
2. 連続していないポイントカットのシーケンス (Msg1 →⋯→ Msg2)

これらの異なるシーケンスポイントカットに対応出来るような仕組みを考える必要がある．実現方法として，全てのメッセージを包含する「any フラグメント」という特別な概念を定義して，ポイントカットの間に挿入することで連続と不連続の区別が可能になる．

4.3.3 要件 3 に関する提案

アスペクトモデル中に記述されたジョインポイントモデルの情報によって，ウィーブ時のシステムモデルへの変更要素が決まる．よって，シーケンス図とステートマシン図との要素の対応づけを行う必要がある。ジョインポイントモデルに注目して，シーケンス図の要素が与えるステートマシン図への影響を調査し，ウィーブ手法を検討する．

4.4 アスペクトとしてのシーケンス図

4.4.1 アスペクトが記述可能な範囲の定義

シーケンス図は各ライフライン間で交わされるメッセージのやり取り (メッセージパッシング) をモデル化する手法である．このことから，シーケンス図をアスペクトで定義した時に表現可能なシステムのモデルはメッセージ通信 (チャネル通信) を伴うシステムに限定する．さらに，今回はメッセージの同期通信のみを扱うものとし，図 4.2 のような非同期通信のシーケンスには対応しない．

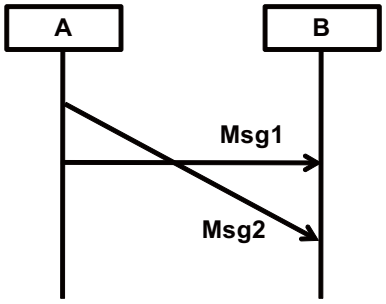


図 4.2: 想定しないシーケンス

4.4.2 シーケンス図で記述されたアスペクトの要素

シーケンス図を構成する要素は，大きく分けて以下の 3 つに分類される．

- ライフライン
- メッセージ
- フラグメント

フラグメントの要素に着目する．シーケンス図が扱うことの出来るフラグメントは複数存在し，その中でも，alt フラグメントはガード条件を持ち，ガード条件によって動作を分岐させることが可能な性質を持つ．この性質は，ガード条件によってウィーブメッセージを変化させることによって，攻撃緩和のシナリオが容易に作成出来ると予想される．よって，今回は alt フラグメントに対応させた手法を提案する．

alt フラグメントを用いたアスペクトの記述例は，本論文で用いる例題で見ることが出来る．

4.4.3 提案するアスペクトが持つジョインポイントモデル

シーケンス図をポイントカットとして扱うために，先に述べたアスペクト指向言語が持つような，ジョインポイントモデルを定義する．シーケンス図が持つ要素に対して，提案する手法のジョインポイントモデルを以下に述べる．

ジョインポイント シーケンス図上のメッセージの両端と交わるライフライン上のポイントを指し，矢印の元と先のセットをジョインポイントとする．

ポイントカット メッセージ又は二つのメッセージのシーケンスを指す．

アドバイス メッセージの前 (before) とメッセージの後 (after) とする．

シーケンス図で記述されるメッセージは，ポイントカット又は追加メッセージのどちらかである．シーケンス図上での実行がジョインポイントであるメッセージに差し掛かると，そのメッセージにポイントカットの情報が付加されているかを判定する．メッセージにポイントカットが付加されていた場合ジョインポイントが絞り込まれる．その後，絞り込まれたジョインポイントと追加メッセージとの前後関係からアドバイスが自動的に決定する．図 4.3 にジョインポイントが絞り込まれる様子を記述する．

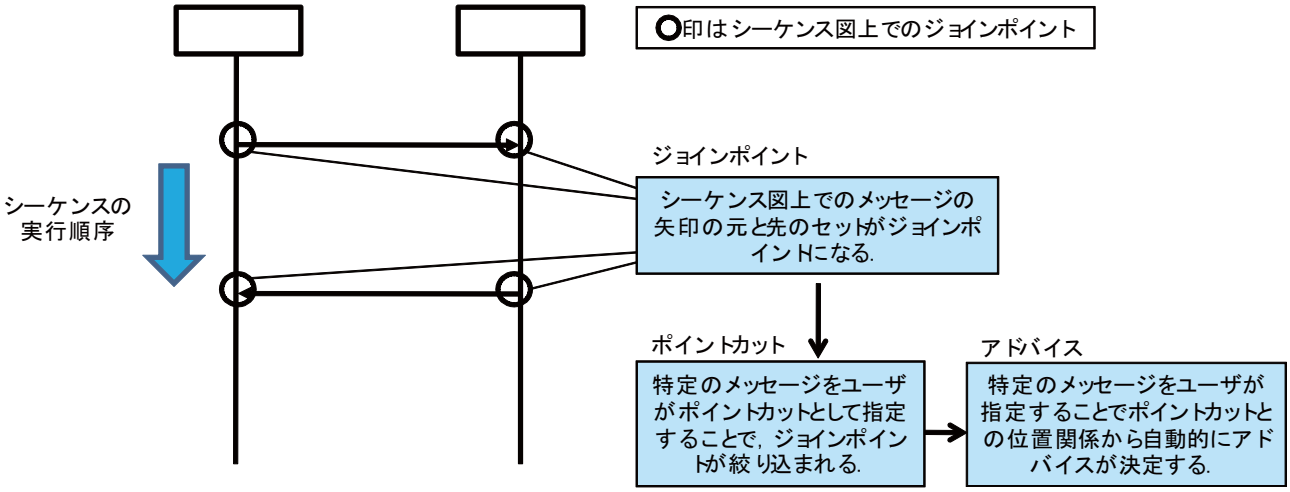


図 4.3: 提案手法のジョインポイントモデル

4.5 アスペクトに持たせる特別な概念

4.5.1 シーケンスポイントカット

提案手法で用いるポイントカットは二つのメッセージのシーケンスで表現される．例えばメッセージ p とメッセージ q という順番で実行されるメッセージがあるとする．これらを p → q(p が起こってから q が起こる) というメッセージのシーケンスと捉える．この二つのポイントカットをシーケンスポイントカットと定義する．また，シーケンスポイントカットによって絞り込まれるジョインポイントの領域は 4 箇所あり，メッセージ p の前後又は，メッセージ q の前後となる．この 4 箇所に追加したいメッセージを記述する．

4.5.2 any フラグメント

any フラグメントは特殊なフラグメントであるため，OMG が提供している UML の仕様には定義されていない．全てのメッセージに該当するフラグメントを any フラグメント

と定義する．

any フラグメントなしのシーケンスポイントカット　また，any フラグメントの有無で，シーケンスポイントカットの意味が異なる．図 4.4 のように記述された Msg1 → Msg2 のポイントカットは，連続したポイントカットのシーケンスと捉えられる．この場合のメッセージの挿入可能箇所は 3 箇所で，Msg1 → Msg2 のシーケンスの直前・直後又は，その間となる．≪ pointcut ≫がポイントカット，≪ create ≫が追加メッセージを表す．

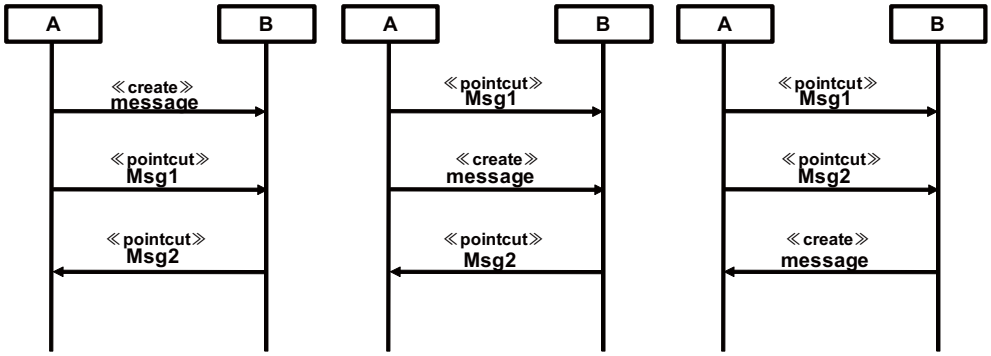


図 4.4: (any フラグメントなし) ミティゲーションモデルの例

any フラグメントありのシーケンスポイントカット　また，図 4.5 のように記述された Msg1 → Msg2 のポイントカットは，Msg1 と Msg2 の間に何らかのメッセージを含む連続しないポイントカットのシーケンスと捉えられる．この場合のメッセージの挿入箇所は 4 箇所で，メッセージ Msg1 の直前・直後又は，メッセージ Msg2 の直前・直後となる．

4.6　アスペクトモデルの記法

ここでは，提案手法に必要となるアスペクトモデルの記法の定義を述べる．

基本的にシーケンス図の要素に，特定のステレオタイプを付加することで，ポイントカットとウィーブする追加メッセージを表す．

4.6.1　ポイントカットの指定

ポイントカットとなるメッセージに対して≪ pointcut ≫のステレオタイプを付加する．また，通常のポイントカットの場合と，シーケンスポイントカットの 2 つの場合に対応し，それらは純粋に≪ pointcut ≫のステレオタイプの数で異なるものと判断される．

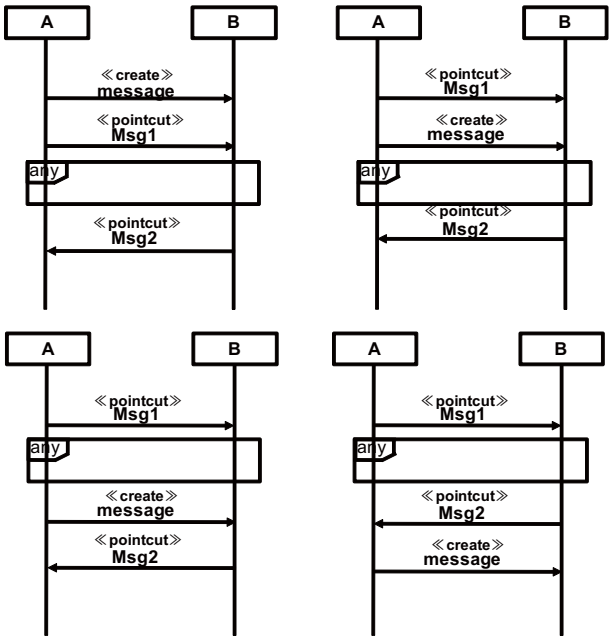


図 4.5: (any フラグメントあり) ミティゲーションモデルの例

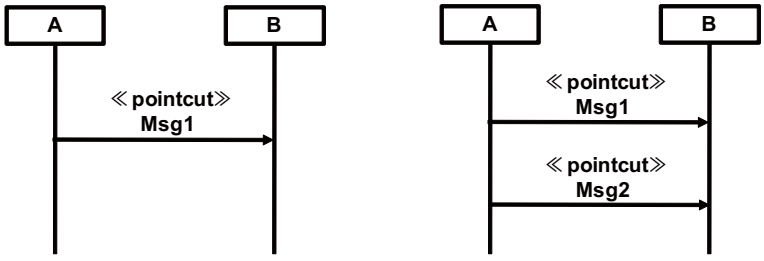


図 4.6: ポイントカットの指定

4.6.2　追加メッセージの指定

追加したいメッセージに対して≪create≫のステレオタイプを付加する.

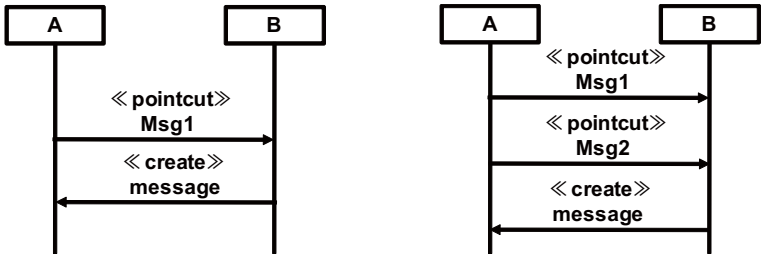


図 4.7: 追加メッセージの指定

4.6.3　追加ライフラインの指定

追加したいライフラインに対して≪create≫のステレオタイプを付加する.

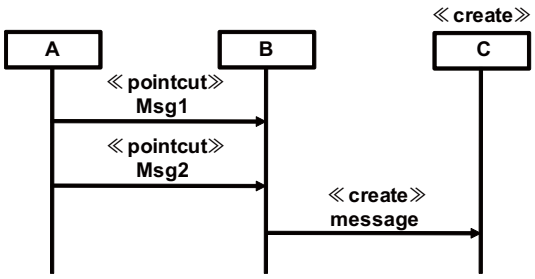


図 4.8: 追加ライフラインの指定

4.7　アスペクトモデルとシステムモデルの関係

アスペクトモデルをシステムモデルにウィーブする時に，以下に記述することに注意する必要がある.

4.7.1　メッセージ追加による影響

図 4.9 はシーケンス図で追加メッセージが指定された時，ステートマシン図に与える影響を示す．基本的にシーケンス図のライフラインが一つのステートマシン図と対応づくため，メッセージを追加すると，ステートマシン図上での影響箇所は二ヶ所になる.

これは，シーケンス図で記述されるメッセージは二つのライフラインにまたがっており，メッセージ自身が「送信元」と「送信先」の情報を持っているからである．

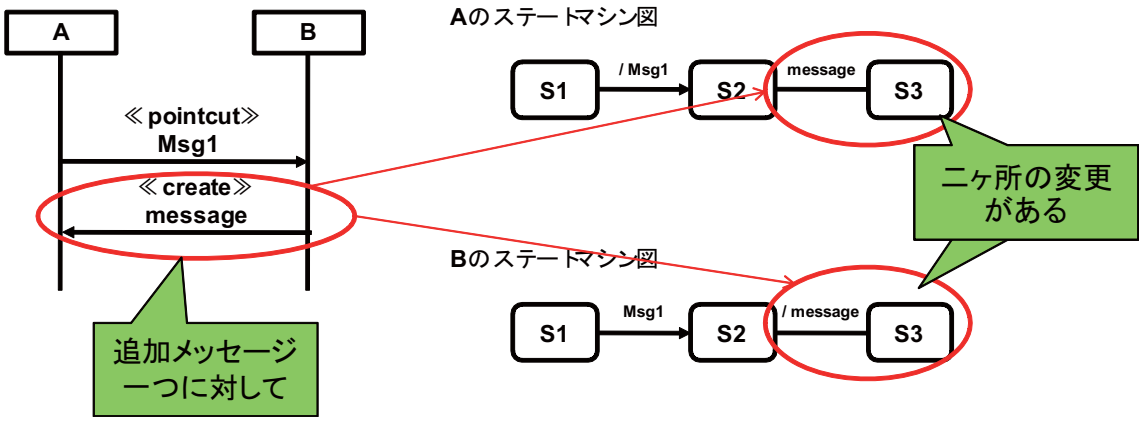


図 4.9: メッセージ追加によるステートマシン図への影響

4.7.2 ライフライン追加による影響

ライフラインの追加に関しては，シーケンス図のライフライン一つに対して，ステートマシン図が一つ出来るため，一対一に対応づけられる．

4.8 システムモデル

ここでは，ミティゲーションモデルのウィーブ先であるシステムモデルについて説明し，提案手法が想定するシステムについて述べる，

4.8.1 想定するシステム

提案手法はオブジェクト間でメッセージを通信して動作するシステムを想定している．メッセージを相手に送信することで通信がなされる，逆にメッセージを受信しそれに応じて適切な処理を行う．また，メッセージ通信の種類は同期通信を想定する．

システムモデルに必要なモデルは以下の 2 つである．

- システムのクラス図
- システムのステートマシン図

4.8.2 クラス図

メッセージ通信の通信路，及びメッセージの種類をクラス図で表現する．

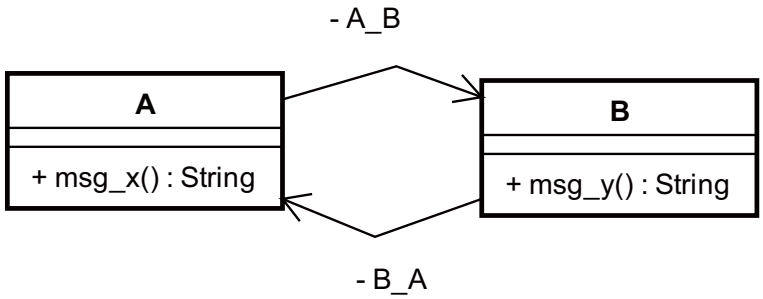


図 4.10: システムのクラス図

メッセージの通信路を関連に，メッセージはそれぞれが操作に記述する．

図 4.10 は，クラス (オブジェクト)A が持つメッセージは”msg_x”で A → B の向きにのみメッセージ送信が可能であることを意味している．また，クラス (オブジェクト)B から送られてくる”msg_y”を受信可能であることを意味している．逆も同様である．

4.8.3 ステートマシン図

各オブジェクトの振る舞いはステートマシン図で表現する．

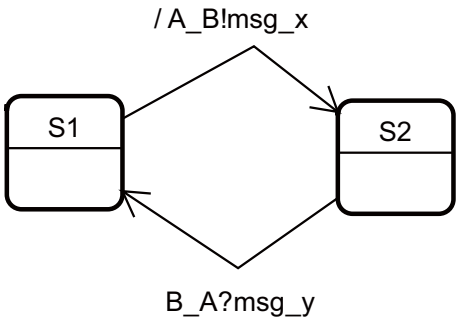


図 4.11: システムのステートマシン図 (A)

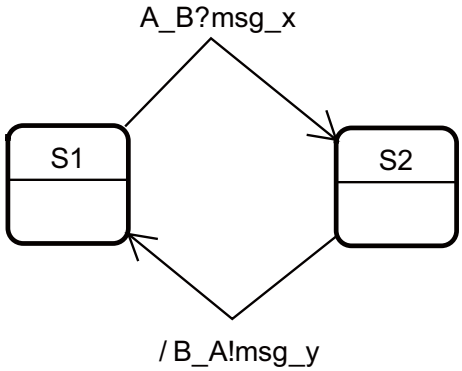


図 4.12: システムのステートマシン図 (B)

図 4.11 図 4.12 の例では，A の場合，メッセージの送信時は「A_B!msg_x」，受信時は「B_A?msg_y」と記述する．逆も同様である．

第5章　提案するウィーブ手法

本章では，本研究で提案するウィーブ手法の概要を述べ，提案手法の手順，提案手法の実現方法について述べる．次に，提案手法を用いて実装したモデルについて述べ，最後に，提案手法に用いた技術について述べる．

5.1　提案するウィーブ手法の概要

シーケンス図で定義したアスペクトモデルをステートマシン図で記述されたシステムモデルに，ウィーブする手法を提案する．

まず，シーケンス図で指定したポイントカットの位置が指し示す，ステートマシン図上のポイントを解析する必要がある．ここでは，ポイントカットを解析するためにモデル検査器 SPIN を用いた手法を提案する．SPIN は本来，ソフトウェアの設計の正しさを検証するためのツールであるが，その検証方法は，検証用記述言語 Promela で記述された各プロセスに対して，全ての状態を網羅的に探索するという特徴を持つ．この特徴を使って，ステートマシン図の全状態を SPIN に網羅的検査させ，その実行列の中に検出したいポイントカットのシーケンスがあるかを検査する．網羅検査によって，該当するシーケンスが見つかったならば，それは，アスペクトモデルで指定した，Promela 上でのポイントということになり，そのポイントに「ポイントカットを指し示すラベル」を挿入することで，Promela 上でのポイントカットの位置を知ることができる．

また，この手法を用いる際にシステムモデルを Promela で記述するが，ステートマシン図と Promela は意味論的にほぼ等価なモデルであるため，特に問題ないものと考える．

次に，SPIN によって解析され,「ポイントカットを指し示すラベル」が挿入された Promela に対して，挿入されたラベルの情報を元に追加要素のウィーブを行う．

5.2　ウィーブの全体図

提案するウィーブ手法の全体図を図 5.1 に示す．

図中の四角はデータを，丸は処理を表す．

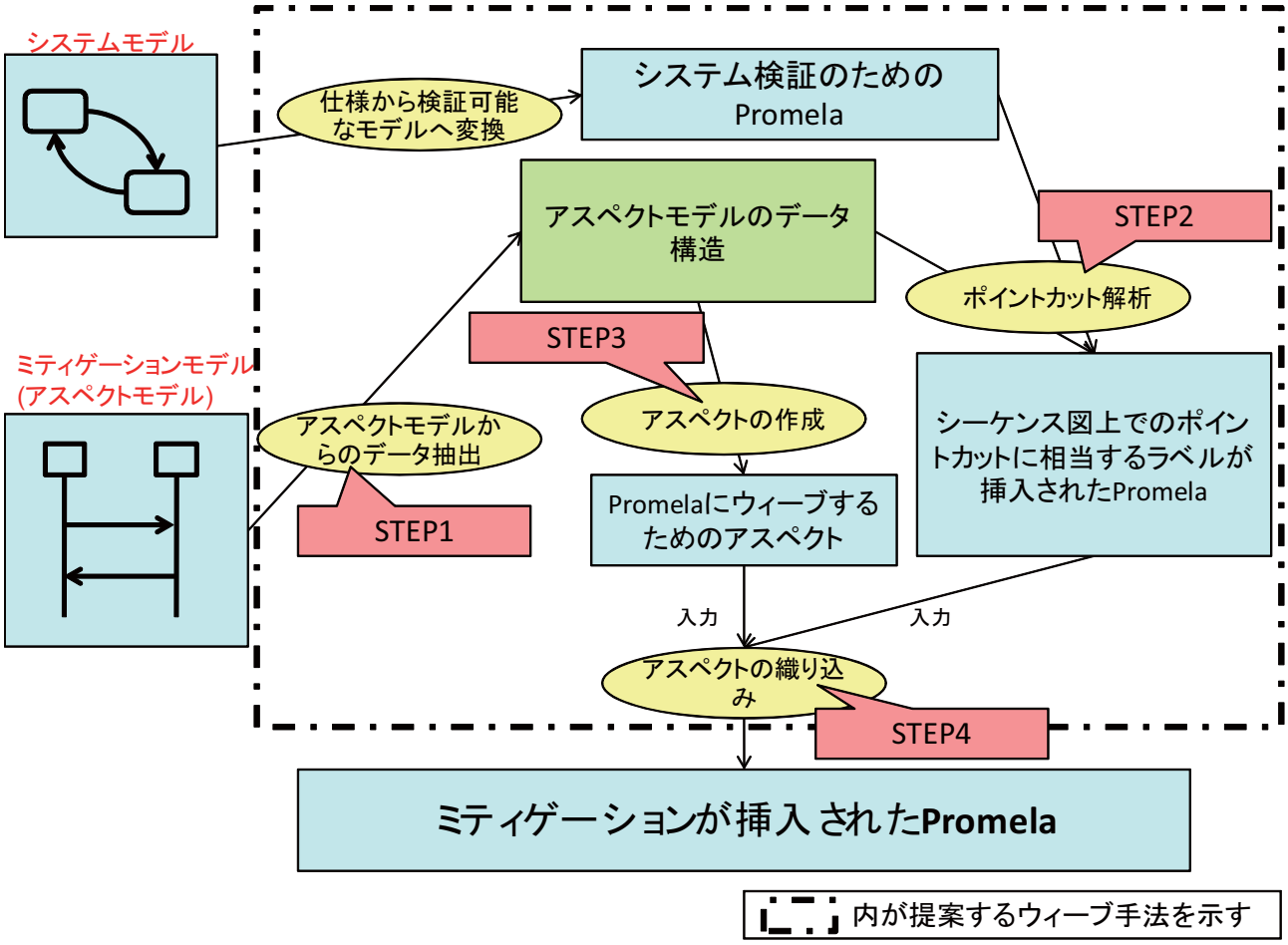


図 5.1: 提案するウィーブ手法の全体図

5.3　ウィーブ手法の実装

ウィーブの全体図で示した各 STEP について検討する．

5.3.1　STEP1 ウィーブに必要な情報を抽出

シーケンス図で記述されたアスペクトモデルが持つデータ (ライフライン，メッセージ，フラグメントなど) を抽出する．抽出の際にシーケンス図の上から順番にメッセージ単位でデータを抽出する．抽出したデータは Message クラスとする．

また，ライフラインの追加，変数の追加についてはノートに記述することにしている．記述方法は「create:ライフライン名」「var:変数名」と定義しており，Nortation クラスとしてデータを抽出する．

アスペクトモデルから抽出したデータ構造を図 5.2 に示す．

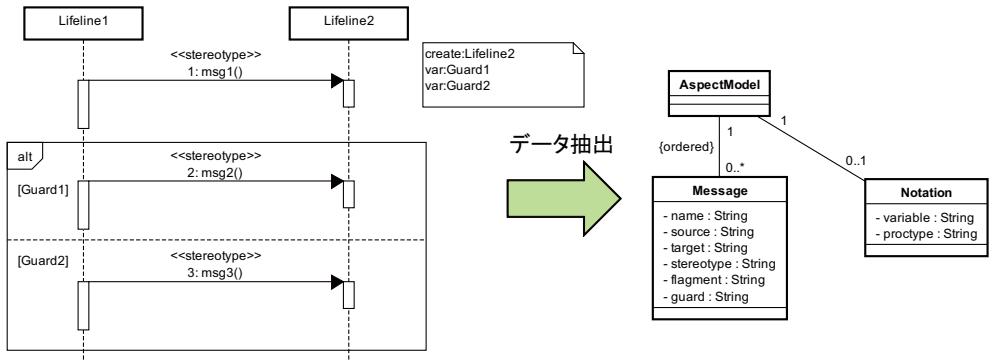


図 5.2: アスペクトモデルから抽出されたデータ構造

5.3.2　STEP2 ポイントカット解析

Promela にポイントカットを指し示すラベルを挿入することによって，システムモデル上のポイントカットの箇所を特定する．

ここで用いるラベルは 2 種類ある．それは，ポイントカット解析前に，ポイントカット解析用に挿入するラベルと，ポイントカット解析後に，ポイントカットであることを指し示すために挿入するラベルである．

1. ポイントカットになる可能性のあるメッセージ全てに挿入するラベル (ポイントカット解析用のラベル)

2. ポイントカットを指し示すメッセージに挿入するラベル (ポイントカットを指し示すラベル)

また，STEP2 は以下の手順で行う．

1. Promela にポイントカット解析用のラベルを挿入
2. システムの実行列にポイントカットが存在するかを SPIN で網羅検査
3. Promela にポイントカットを指し示すラベルを挿入

以下に，その手順を示す．

Promela にポイントカット解析用のラベルを挿入

例えばシステム全体の実行シーケンスが「msg1 → msg2 → msg1」で，ポイントカットが「msg1 → msg2」だったとすると (図 5.3)，実行シーケンスの最初の 2 つのメッセージがポイントカットに相当するメッセージであるが，最後のメッセージ msg1 はポイントカットにはならない．しかし，msg1 はポイントカット「msg1 → msg2」のメッセージの一部であるため，ポイントカットになる可能性のあるメッセージと解釈されて，ポイントカット解析用のラベルが挿入される．

アスペクトモデルを図 5.3 に示す．解析用のラベルを挿入したイメージを図 5.4 に示す．

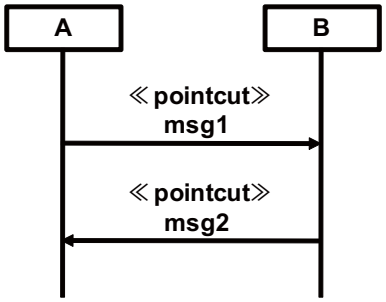


図 5.3: アスペクトモデル (msg1 → msg2 のポイントカットの指定時)

SPIN による網羅検査

SPIN を用いてポイントカットのシーケンスを探索するために，NEVER CLAIM を作成して「p → q → r → s」のシーケンスについて検査した．システムの実行列に該当するシーケンスが存在すれば，その該当箇所を反例として出力する．また，このシーケンスの意味を図 5.5 で記述する．

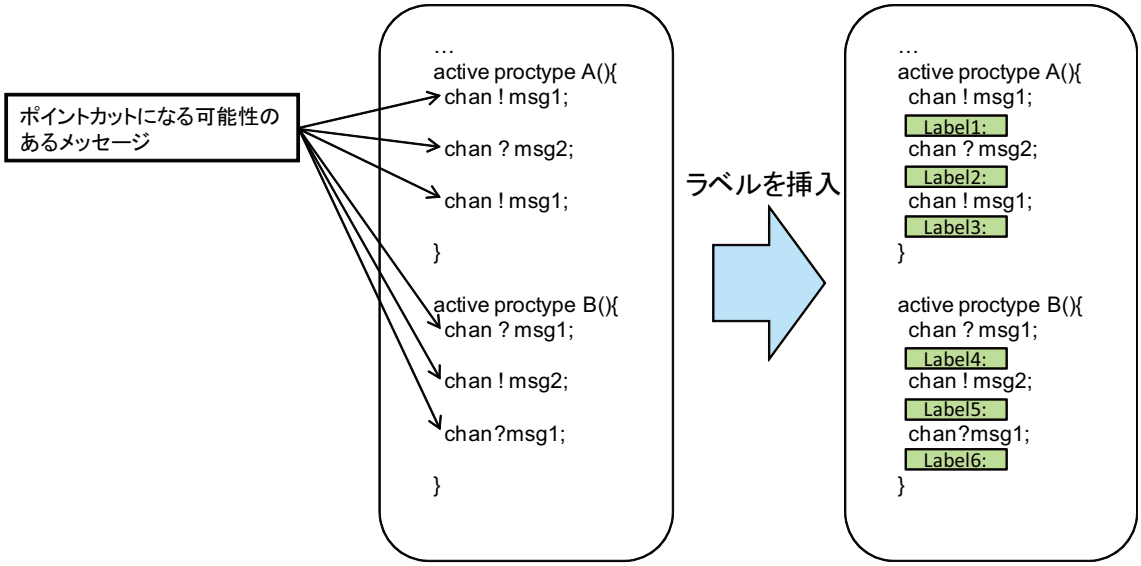


図 5.4: ポイントカットになる可能性のあるラベルの挿入

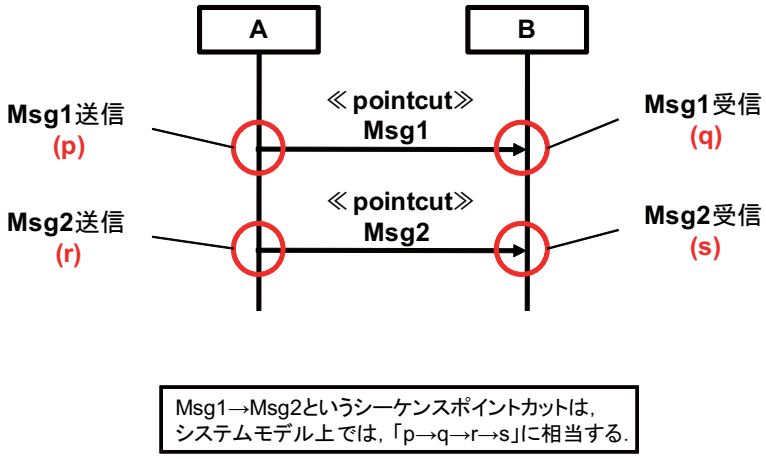


図 5.5: 検査対象のシーケンス

```
never { /*シーケンスポイントカット解析用*/
T0_init:
if
:: p -> goto T0_S1
:: !p -> goto T0_init
fi;
T0_S1:
if
:: q -> goto T0_S2
:: !q -> goto T0_S1
fi;
T0_S2:
if
:: r -> goto T0_S3
:: !r -> goto T0_S2
fi;
T0_S3:
if
:: s -> goto accept_all
:: !s -> goto T0_S3
fi;
accept_all:
skip
}
```

Promela にポイントカットを指し示すラベルの挿入

図 5.6 は図 5.4 からさらにポイントカットを絞り込んだ結果である．

5.3.3 STEP3 アスペクトの作成

ここでは，STEP2 で解析用のラベルが挿入された Promela に対してウィーブするアスペクトを生成する．ここでのアスペクトの記述方法は，ウィーブ手法に依存して決定する．今回は Promela へのウィーブをサポートしてある Aspect Promela ツールを用いるため，Aspect Promela が入力可能なアスペクトを生成する．具体的なアスペクト記述は，本論文で用いる例題で見ることが出来る．

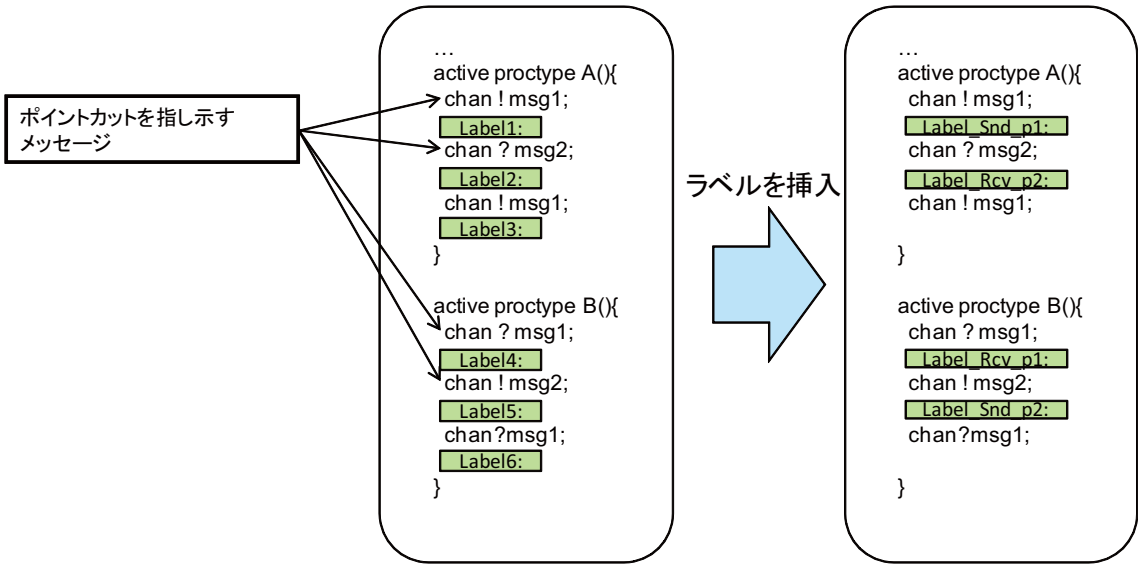


図 5.6: ポイントカットを指し示すラベルの挿入

5.3.4 STEP4 アスペクトの織り込み

ここでは，挿入したラベルを目印にして，Aspect Promela を用いて，STEP2 で得た Promela に STEP3 で得たアスペクトをウィーブする．

STEP1〜4 までを，提案するウィーバとする．
提案手法の実装に関して，使用したツールを簡単に説明する．

5.4 実装に使用したツール

JUDE Professional
提案手法に必要なモデルを記述するために，一般的に良く知られているツールとして UML モデリングツール JUDE Professional を用いた．

SPIN
システムモデル上のポイントカットを解析するために，モデル検査器 SPIN を用いた．また，SPIN はウィーブ後のシステムモデルを検証する時も使用する．

Aspect Promela
システムモデルにアスペクトをウィーブする時に，Promela に必要なコードをウィーブするツールが必要であった．Promela へのウィーブは，本研究室の OB である大野が作成した検証支援ツール Aspect Promela を用いた．

第6章　例題への適用

本章では，提案するウィーブ手法を電子投票システムの例題に適用させ，そのウィーブの有効性について述べる．

6.1　例題の概要

適用する例題には，簡易版の電子投票システムを使用した．この例題は関連研究でも用いられている例題で，実際に米国で問題になったDievold社製の電子投票機”Accu Vote”が持つセキュリティ脆弱性を指摘した論文に基づく事例である．この電子投票機はスマートカードを使用して投票を行う．投票機に挿入したスマートカードが認識されたら投票を開始することが出来る．指摘されている問題点としては，カードが容易に偽造出来るという点であり，この問題について様々な場所で議論がなされている．

本論文では，この事例に関する例題として簡易版の電子投票システムを作成し，提案するウィーブ手法の有効性を検討する．

6.2　適用する例題の説明

例題の流れは，投票者 (Voter) がスマートカードを投票機 (VotingMachine) に挿入し，スマートカードレコード (SmartcardRecord) がカードを認識する．カードが認識された後，投票を行うというものである．投票の流れを以下に示す．

1. Voter が VotingMachine にカード挿入する (insertcard)
2. VotingMachine が SmartcardRecord にスマートカードの有無を問い合わせる (authenticate)
3. SmardcardRecord が VotingMachine にスマートカードが有ることを知らせる (authenticated)
4. VotingMachine が Voter に投票の許可をする (voteAllowing)
5. Voter が VotingMachine に投票する (submit)

6.3 例題の仕様

例題の仕様を以下に示す．これらのモデルは，投票処理 (insertcard から submit まで) が繰り返されるモデルである．

6.3.1 クラス図

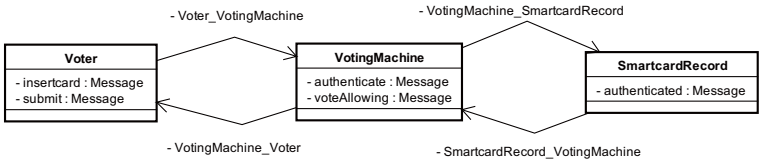


図 6.1: 例題の仕様

6.3.2 ステートマシン図

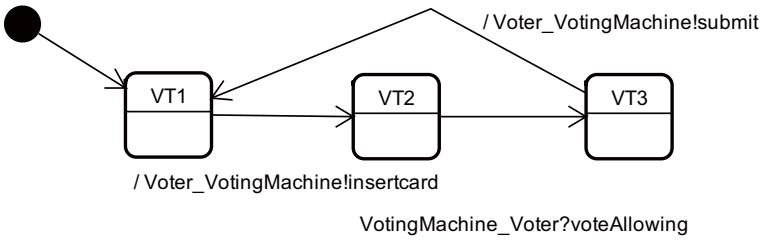


図 6.2: 例題の仕様 (Voter)

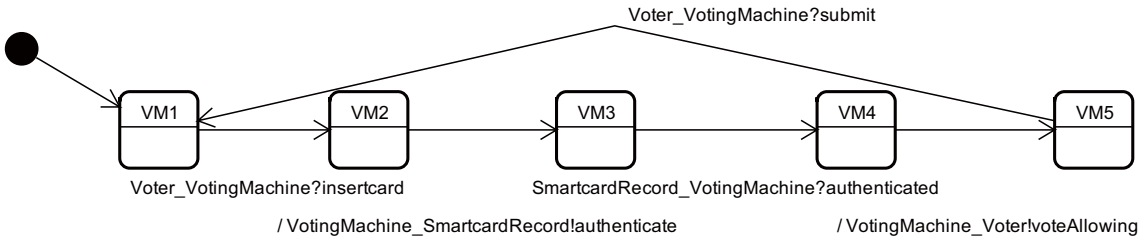


図 6.3: 例題の仕様 (VotingMachine)

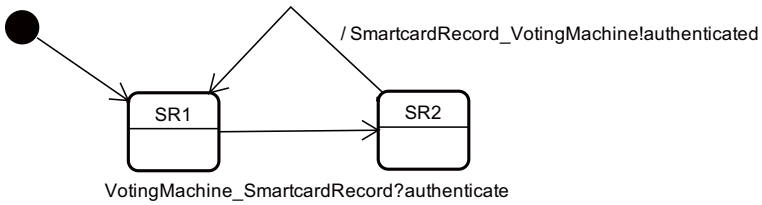


図 6.4: 例題の仕様 (SmartcardRecord)

6.4 攻撃モデル

この仕様では，同じ Voter が何度も投票することが可能であることが容易に理解できる．攻撃のシナリオに複数回の投票を考える．攻撃モデルを図 6.5 に示す．

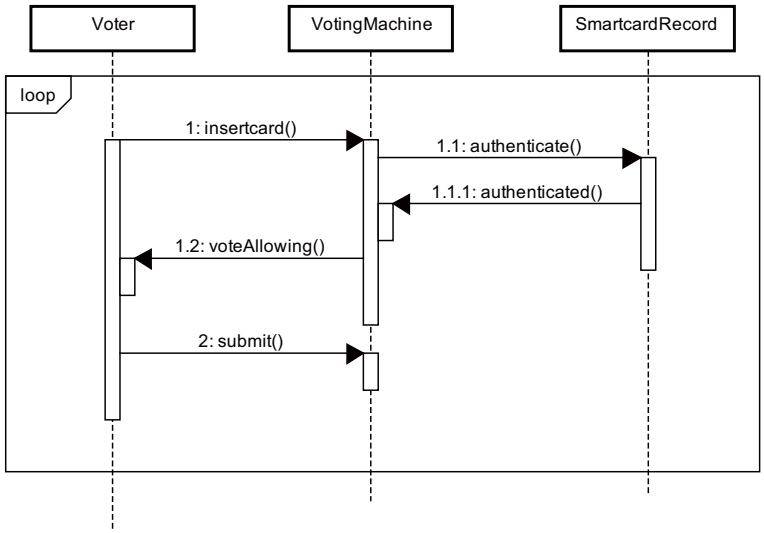


図 6.5: 攻撃モデル

6.5 ミティゲーションモデル

攻撃シナリオに対して，攻撃を緩和するミティゲーションモデルを作成する．今の仕様では，投票時に投票回数を制限する仕組みがないため，複数回の投票が可能になっている．そこで，投票回数をカウントするシナリオをミティゲーションモデルとして作成し，システムのステートマシン図にウィーブすることを考える．図 6.6 にミティゲーションモデルを示す．

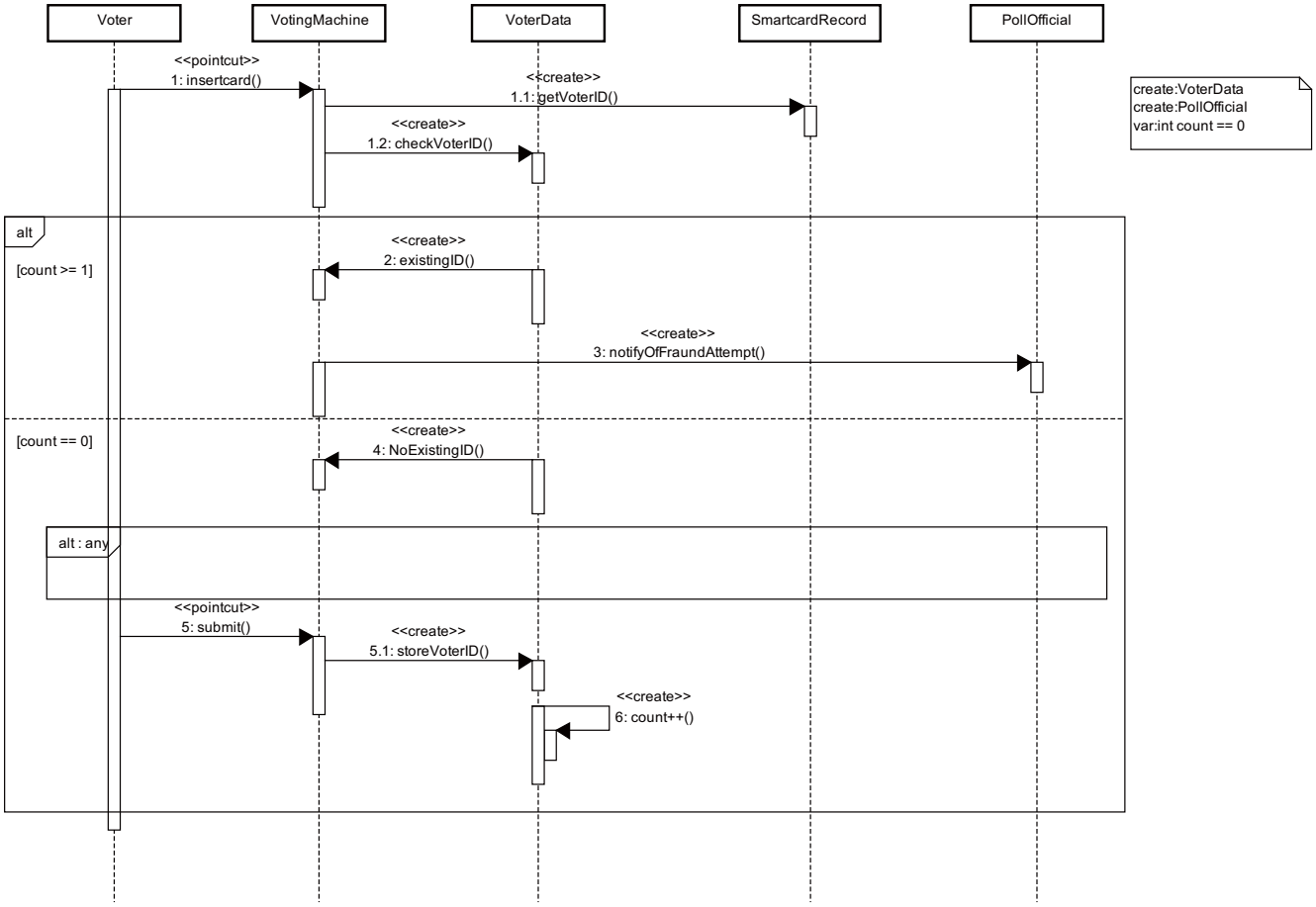


図 6.6: ミティゲーションモデル

6.5.1 ミティゲーションのウィーブ

作成したミティゲーションモデルを Promela にウィーブした.

ウィーブ前の Promela とウィーブ後の Promela を，ソースコード 6.1 とソースコード 6.2 に示す.

また，ソースコード 6.2 中のウィーブによって追加された箇所の行数を示す.

1 行目 メッセージの追加

7-10 行目 チャネル (通信路) の追加，変数の追加

15-20 行目 メッセージ通信の追加

25-27 行目 メッセージ通信の追加

41 行目 メッセージ通信の追加

48-64 行目 proctype の追加

ソースコード 6.1: ウィーブ前の Promela

```
1  mtype = {insertcard, authenticate, authenticated, voteAllowing, submit_};
2
3  chan Voter_VotingMachine = [0] of {mtype}
4  chan VotingMachine_Voter = [0] of {mtype}
5  chan VotingMachine_SmartcardRecord = [0] of {mtype}
6  chan SmartcardRecord_VotingMachine = [0] of {mtype}
7
8  active proctype VotingMachine(){
9  do
10   ::Voter_VotingMachine?insertcard;
11     VotingMachine_SmartcardRecord!authenticate;
12     SmartcardRecord_VotingMachine?authenticated;
13     VotingMachine_Voter!voteAllowing;
14     Voter_VotingMachine?submit_;
15 od;
16 }
17
18 active proctype Voter(){
19 do
20   ::Voter_VotingMachine!insertcard;
21     VotingMachine_Voter?voteAllowing;
22     Voter_VotingMachine!submit_;
23 od;
24 }
25
26 active proctype SmartcardRecord(){
27 do
28   ::VotingMachine_SmartcardRecord?authenticate;
29     SmartcardRecord_VotingMachine!authenticated
```

```
30 od;
31 }
```

ソースコード 6.2: ウィーブ後の Promela

```
1  mtype={insertcard, authenticate, authenticated, voteAllowing, submit_,
       getVoterID, checkVoterID, existingID, nofifyOfFraudAttempt, NoExistingID,
       storeVoterID}
2
3  chan Voter_VotingMachine=[0] of {mtype};
4  chan VotingMachine_Voter=[0] of {mtype};
5  chan VotingMachine_SmartcardRecord=[0] of {mtype};
6  chan SmartcardRecord_VotingMachine=[0] of {mtype};
7  chan VotingMachine_VoterData=[0] of {mtype};
8  chan VoterData_VotingMachine=[0] of {mtype};
9  chan VotingMachine_PollOfficial=[0] of {mtype};
10 int count=0;
11
12 active proctype VotingMachine() {
13 do
14 ::Voter_VotingMachine?insertcard;
15   VotingMachine_SmartcardRecord!getVoterID;
16   VotingMachine_VoterData!checkVoterID;
17   if
18    ::VoterData_VotingMachine?existingID;
19      VotingMachine_PollOfficial!nofifyOfFraudAttempt;
20    ::VoterData_VotingMachine?NoExistingID;
21      VotingMachine_SmartcardRecord!authenticate;
22      SmartcardRecord_VotingMachine?authenticated;
23      VotingMachine_Voter!voteAllowing;
24      Voter_VotingMachine?submit_;
25      VotingMachine_VoterData!storeVoterID;
26      count++;
27   fi;
28 od;
29 }
30
31 active proctype Voter() {
32 do
33 ::Voter_VotingMachine!insertcard;
34   VotingMachine_Voter?voteAllowing;
35   Voter_VotingMachine!submit_;
36 od;
37 }
38
39 active proctype SmartcardRecord() {
40 do
41  ::VotingMachine_SmartcardRecord?getVoterID;
42  ::VotingMachine_SmartcardRecord?authenticate;
43    SmartcardRecord_VotingMachine!authenticated;
44 od;
45 }
46
```

```
47
48 active  proctype VoterData() {
49 do
50   ::VotingMachine_VoterData?checkVoterID;
51     if
52       ::count>=1->
53         VoterData_VotingMachine!existingID;
54       ::count==0->
55         VoterData_VotingMachine!NoExistingID;
56         VotingMachine_VoterData?storeVoterID;
57     fi;
58 od;
59 }
60
61 active  proctype PollOfficial() {
62 do
63   ::VotingMachine_PollOfficial?nofifyOfFraudAttempt;
64 od;
65 }
```

6.5.2 検証性質

検証したい性質は攻撃モデルに依存して決定される．今回は「常に count 変数が 1 以下である」という検証性質に対して検証を行った．

NEVER CLAIM 及び，変数置き換えのマクロを以下に記述する．

NEVER CLAIM

```
never {    /* !( [] p ) */
T0_init:
if
:: (! ((p))) -> goto accept_all
:: (1) -> goto T0_init
fi;
accept_all:
skip
}
```

マクロ

```
#define p (count<=1)
```

6.5.3 評価結果

検証の結果，投票回数を表す変数 count が 0 又は 1 の値しかとらないことが確認できた．

6.5.4 考察

提案するウィーブ手法を例題に適用して，ミティゲーションをウィーブされたシステムモデルでは攻撃が成功しないことを検証した．

結果，想定するウィーブが出来たことを確認した．

第7章　まとめ

本研究では，関連研究 [2] に着目して，対象とするシステムの全ての振る舞いに対して，セキュアな設計が可能な手法の考案を行うことを目的とした．

関連研究ではシステムのモデルをシナリオ (シーケンス図) としていたが，本研究では複数のシナリオを包含するモデルとしてステートマシン図を使用することとし，シーケンス図で記述されたアスペクトを指定したステートマシン図上の全ての箇所にウィーブする手法を提案した．提案手法の実現に当たり，シーケンス図の要素がステートマシン図の要素と一対一に対応づけられないことから，シーケンス図上のポイントカットが指す領域が，ステートマシン図上どの領域に相当するのかを調査する必要があった．

よりセキュアな設計を行うために，アスペクトで指定したポイントカットが，ステートマシン図のどの部分に相当するのかを調べ，該当する箇所全てに攻撃緩和のシナリオを挿入する必要があった．そこで，ステートマシン図の全状態を網羅的に調べるツールとして，モデル検査器 SPIN を用いた網羅探索を行った．SPIN を用いた網羅探索によって，システムの実行列の中から特定のポイントカットのシーケンスが出現する箇所を解析することができた．ステートマシン図上でのポイントカットが絞り込むことが出来，その絞り込まれた箇所に対して攻撃緩和のシナリオをウィーブを行った．

そして，提案するウィーブ手法の有効性を確認するために，本手法を電子投票システムの例題に適用して，システムに攻撃緩和のシナリオをウィーブした．

その結果，想定通りのウィーブを行うことができ，さらにウィーブ後のモデルが攻撃を防ぐことが SPIN による検証実験で確認できた．

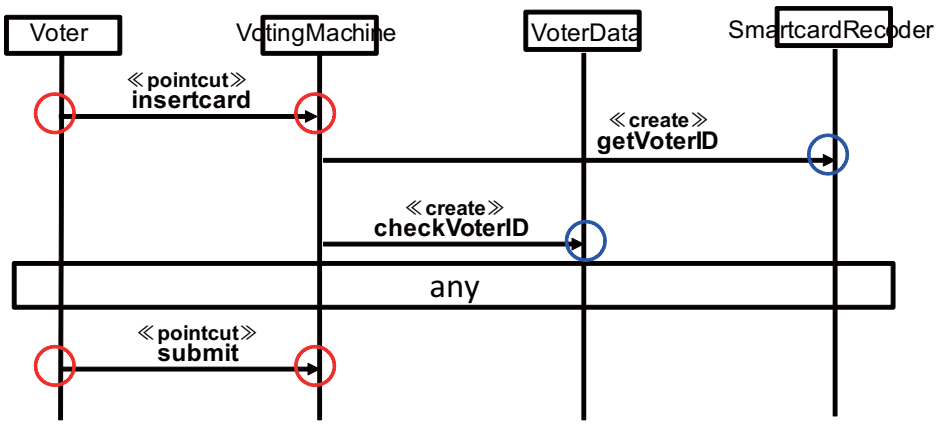
7.1　今後の課題

7.1.1　SPIN によるポイントカット探索法

本研究による，ステートマシン図へのウィーブはステートマシン図の全状態を SPIN に網羅探索させて，アスペクトモデルで指定したポイントカットを解析することによって実現した．しかし，この探索で発見できるポイントカットがステートマシン図上で表わされるポイントカットの全てではない．

例題に使ったアスペクトモデル (図 7.1) を例にとって説明する．

アスペクトモデルによって指定されたポイントカットは,「insertcard → submit」のシーケンスであった．そのため，提案する手法ではその二つのメッセージのシーケンス (正確



SPINを用いた解析によって，赤丸部分にラベルを挿入することが出来た．
しかし，赤丸部分のラベルから青丸部分のメッセージ受信のタイミングは知ることが出来ない．

図 7.1: ポイントカットの解析時の問題点

には，それぞれのメッセージの送信元，送信先を含む 4 つのシーケンス) を SPIN によって探索させた．この解析方法によって発見できる箇所は，ポイントカットのメッセージを送信又は受信しているライフラインのみである．

よって，その他のライフライン上に受信メッセージを≪create ≫した場合は，その受信タイミングを知ることが出来ない．今回は，この問題点に対する対策法として，該当するメッセージが≪create ≫された場合は，必ず訪れる場所全てに受信メッセージを挿入するという方法をとった．この方法は正確なミティゲーションの挿入とは言えないため，他の解決方法を考える必要があると考える．

一つの案として，ライフライン上でメッセージ受信をする箇所全てに状態名をつけて，全てのメッセージにポイントカットを与えるという方法が考えられる．

この方法によって，的確な場所に攻撃を緩和するシナリオが挿入されるものと考える．

第8章　終わりに

謝辞

本論文を執筆するに当たって，多大なるご指導を賜りました，岸知二特任教授，青木利晃特任准教授に感謝申し上げます。また，本研究を進めるに当たって，様々な議論に応じてくださった，岸研究室，青木研究室の皆様に感謝いたします．ありがとうございました．

参考文献

[1] I. Alexander, "Misuse cases: use cases with hostile intent," IEEE Software, vol. 20, pp. 58–66, Jan./Feb. 2003.

[2] J.Whittle, D.Wijesekera, M.Hartong, "Executable Misuse Cases for Modeling Security Concerns", ICSE pp. 121–130, 2008.

[3] 大野真一郎, "モデル検査のためのアスペクト指向でのモデル記述支援環境", ソフトウェア工学研究会, vol. 48, No. 5, 2007.