# セキュリティを考慮したインタラクティブシステムのための アスペクト指向アーキテクチャの提案

2014SE031 石塚 雄太　2014SE049 北川 智久

指導教員：張 漢明

## 1　はじめに

近年、IT 技術の発展に伴い、証券取引やショッピングなど、人々の経済活動の大部分がインタラクティブシステムを用いて行われるようになった．インタラクティブシステムは人々の生活に欠かせないほど普及している．

インタラクティブシステムにおける代表的な脆弱性には、クロスサイト・スクリプティング（以下、XSS と呼ぶ）の脆弱性と SQL インジェクションの脆弱性がある．インタラクティブシステムにセキュリティ対策がなされていない場合、それらの脆弱性を悪用した攻撃によって、外部から不正な操作を行われたり、利用者の個人情報が漏えいするなど、人々の生活に甚大な被害が出る可能性がある．しかし、それらの脆弱性への対策がなされていないインタラクティブシステムが、数多く存在する．

Apache Struts2 など、セキュリティ機能を備えたアプリケーションフレームワークを用いてアプリケーションの開発を行えば、それらの脆弱性への対策を考慮する必要はない．しかし、他の方法でアプリケーションを開発する場合、セキュリティ対策について注意しながらアプリケーションを開発しなければならない．さらに、セキュリティ対策のためのコードがプログラムの各所に散らばってしまい、煩雑になる可能性がある．

本研究の目的は、セキュリティ対策について特に考慮することなくアプリケーションの開発を可能とするアスペクト指向アーキテクチャを提案することである．本研究室では、インタラクティブシステムのためのアスペクト指向アーキテクチャ（以下、CSA-ISys と呼ぶ）が提案されている [5]．それを基盤に、XSS の脆弱性と SQL インジェクションの脆弱性に対するセキュリティ機能を追加することで、アーキテクチャを再設計する．

目的の達成のため、XSS の脆弱性と SQL インジェクションの脆弱性への一般的な対策方法を調査した．そして一般的な対策方法が、CSA-ISys のどのモジュールに関わるのか検討した．結果として、これらのセキュリティ対策方法は、複数のモジュールに横断的に関わる関心事であることが分かった．この横断的関心事をアスペクトモジュールとして設計し、CSA-ISys に追加することで再設計した．この再設計したアーキテクチャを用いて簡単なウェブアプリケーションを設計し、提案方法の有用性を検証した．

CSA-ISys に組み込む際には、本研究室で提案されている PBR パターン [5] を用いることで、インタラクティブシステムの実行中に、セキュリティ対策方法の変更や追加を可能とする柔軟性を保証することができる．また、CSA-ISys と、我々が再設計したアスペクト指向アーキテクチャを比較した結果を考察した．

## 2　インタラクティブシステムにおけるセキュリティ対策

2017 年第 2 四半期 ソフトウェア等の脆弱性関連情報に関する届出状況 [4] によると、ウェブアプリケーションおよびスマートデバイスアプリケーションの届け出の割合が、全体の約半数を占めている．また、任意のスクリプトの実行が原因である割合が 38 %と他と比べて最も高い．したがって、ソフトウェアにおける脆弱性はウェブアプリケーションとスマートデバイスアプリケーションで発見されやすく、有害なスクリプトが実行されるような被害を受けやすいことがわかる．

### 2.1　ウェブアプリケーションにおける脆弱性

#### 2.1.1　クロスサイト・スクリプティングの脆弱性

XSS の脆弱性とは、主に入力フォームや URL に有害なスクリプトを埋め込むことのできる脆弱性である [2]．ユーザの入力に対して動的に HTML ページを生成するウェブアプリケーションにおいて、この脆弱性が存在する可能性がある．この脆弱性が存在するウェブアプリケーションは、以下の手順によって攻撃を受ける．

1. 攻撃者が、XSS の脆弱性への対策が施されていないウェブアプリケーションのテキスト入力欄などに有害なスクリプトを入力する．
2. 有害なスクリプトがウェブサーバに送信される．ウェブサーバが有害なスクリプトを受信し、そのままブラウザに返信する．
3. 有害なスクリプトが遷移先のブラウザ上で実行される．

XSS 攻撃では、攻撃者が入力したスクリプトがブラウザ上で実行されるだけで、標的であるウェブアプリケーションに直接影響は出ない．攻撃者でない第三者が、メールなどに添付された XSS の脆弱性が存在するウェブアプリケーションへの URL にアクセスするだけで、その人物のコンピュータ上で有害なスクリプトが実行される．

#### 2.1.2　SQL インジェクションの脆弱性

SQL インジェクションの脆弱性とは、ウェブアプリケーションの入力画面で SQL 文を含む文字列を入力することで、データベースに有害な操作を行うことのできる脆弱性である [2]．データベースと連携したウェブアプリケーションにおいて、この脆弱性が存在する可能性がある．

データベースへの操作や問い合わせなどを行うプログラムに、パラメータとして細工された SQL 文が挿入されることで、データベースの改ざんや情報漏洩といった問題が発生する．

#### 2.1.3 脆弱性の根本的な原因

これら 2 つの脆弱性の根本的な原因は共通している．外部から入力された有害な文字列が、無害な文字列に変換されることなくブラウザやデータベースに送られることである．この問題を解消するためには、入力された有害な文字列を無害化する機能 [3] が必要である．

### 2.2 一般的な対策

XSS の脆弱性と SQL インジェクションの脆弱性への一般的な対策方法を調査した結果 [2] を以下に列挙した．

**XSS の脆弱性への対策**

- ウェブページに出力する全ての要素に対して、エスケープ処理を施す．
- 入力された HTML テキストから構文解析木を作成し、スクリプトを含まない必要な要素のみを抽出する．
- HTTP レスポンスヘッダの Content-Type フィールドに文字コード（charset）を指定する．

**SQL インジェクションの脆弱性への対策**

- SQL 文の組み立ては全てプレースホルダで実装する．
- 特別な記号をエスケープ処理する．
- ウェブアプリケーションに渡されるパラメータに SQL 文を直接指定しない．

2.1.3 節で述べた脆弱性の根本的な原因を踏まえると、XSS の脆弱性と SQL インジェクションの脆弱性に有効である対策はエスケープ処理である．

### 2.3 アプリケーション開発におけるセキュリティの問題

Apache Struts2 など、セキュリティ対策が考慮されたアプリケーションフレームワークを用いてアプリケーション開発を行えば、これらの脆弱性への対策を考える必要はない．しかし、他の方法でアプリケーションを開発する場合、セキュリティ対策について注意しながら開発しなければならない．さらに、セキュリティ対策のためのコードがプログラムの各所に散らばってしまい、煩雑になる可能性がある．

## 3 アスペクト指向アーキテクチャにおけるセキュリティ機能の実現

本研究の目的は、セキュリティ対策について特に考慮することなくアプリケーションの開発を可能とするアスペクト指向アーキテクチャを提案することである．アスペクト指向技術とは、横断的関心事を単一モジュールとすることでモジュール間の独立性を高める、オブジェクト指向を補う技術のことである．横断的関心事とは、オブジェクト指向や機能指向など、システムに対する支配的な関心事に基づいて分割された複数のモジュールに横断的に関係があり、支配的な関心事によるモジュール分割では 1 つのモジュールにまとめにくい関心事のことである．

### 3.1 インタラクシステムティブのためのアスペクト指向アーキテクチャ

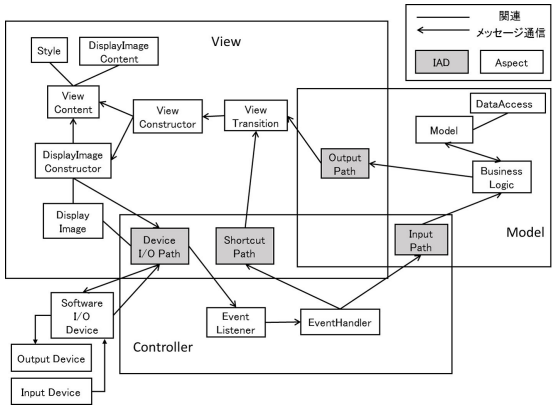


図 1 CSA-ISys

図 1 に、本研究室で提案されている CSA-ISys を示す．CSA-ISys は、インタラクティブシステムのアーキテクチャ中心開発において基盤となり、複数の既存のアーキテクチャをすべて説明可能とする、という設計思想に基づいて設計されたアーキテクチャである．CSA-ISys は、MVC アーキテクチャとその派生に基づいて設計されている．

CSA-ISys において、Model、View、Controller の各コンポーネントが IAD により協調している．IAD(Inter Aspect Description; アスペクト間記述）は、ポイントカットとアドバイス記述を用いて実装する．ポイントカットにはアドバイスをいつ実行するかの指定を記述し、アドバイス記述にはポイントカットで指定されたタイミングで実行する処理を記述する．

CSA-ISys を用いることで、アーキテクチャとアプリケーションの設計およびコードの理解、変更が容易になるだけでなく、ライブラリやミドルウェア等を、大きな粒度で変更する枠組みを提供することが期待できる．
しかし、CSA-ISys はセキュリティ機能を備えていないので、CSA-ISys を基盤に、XSS の脆弱性と SQL インジェクションの脆弱性についてのセキュリティ機能を追加することで、アスペクト指向アーキテクチャを再設計する．

### 3.2 追加するセキュリティ機能

我々は追加するセキュリティ機能にエスケープ処理を採用する．XSS の脆弱性における有害な文字列と、SQL インジェクションの脆弱性における有害な文字列は異なる．CSA-ISys において、XSS の脆弱性と SQL インジェクションの脆弱性で対策する箇所をそれぞれに検討する．

### 3.3 セキュリティ機能を追加したアスペクト指向アーキテクチャの設計

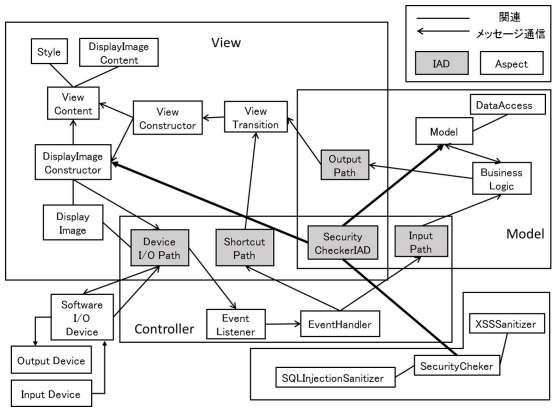


図 2 セキュリティ機能を追加したアスペクト指向アーキテクチャ

図 2 に、セキュリティ機能を追加したアスペクト指向アーキテクチャを示す．セキュリティ機能である *SecurityChecker* コンポーネントと、その IAD である *SecurityCheckerIAD* を CSA-ISys に追加した．*SecurityCheckerIAD* が *Model* と *DisplayImageConstructor* に付加される．*SecurityCheckerIAD* が *SecurityChecker* に通信を送り、*SecurityChecker* が必要なセキュリティ機能を実行する．

XSS の脆弱性のためのセキュリティ機能と、SQL インジェクションの脆弱性のためのセキュリティ機能は、Command パターン [1] で実装した．Command パターンとはデザインパターンの 1 種で、要求をオブジェクトとしてカプセル化することで、要求の種類によってクライアントをパラメータ化することができる．Command パターンを用いてセキュリティ機能を実装することで、「特定の脆弱性への対策」という処理をカプセル化し、処理の詳細を隠ぺいすることによりアプリケーションの安全性が高まる．また、Command パターンでは処理を一つのオブジェクトとしているので、新たな脆弱性に対するセキュリティ機能を追加する際は、セキュリティ機能を新たなオブジェクトとして扱い、追加すればよい．

### 3.4 アスペクト指向アーキテクチャの詳細化

CSA-ISys の具象アーキテクチャ [5] を、図 3 に示す．具象アーキテクチャとは、実現技術を選択し、参照アーキテクチャの構造を詳細化したものである．View コンサーン内の *MappingRule*[5] は、画面内部表現と具象表現との変換を実現するための変則規則を定義したものである．図 4 に、セキュリティ機能を追加した具象アーキテクチャを示す．

具象アーキテクチャにおけるいくつかのコンポーネントは、PBR パターン [5] を用いて記述した．PBR パターンとは、AspectJ のアドバイス記述を抽象化して設計された、動的再構成を目的とした自己適応のためのアーキテクチャパターンである．CSA-ISys においては、コード記述の規範の提示と大粒度でのコンポーネントの再利用を可能にしている．PBR パターンを用いることで、アスペクトの織り込みのためのコンポーネントと、セキュリティ機能のコンポーネントを分離することができるので、コンポーネントの変更を独立に行うことができる．また、インタラクティブシステムの実行中に、セキュリティ対策方法の変更や追加といった動的な再構成も可能とすることができる．したがって、セキュリティ機能を柔軟に織り込むことができる．それぞれの脆弱性への対策について、PBR パターンを用いて具象アーキテクチャを詳細化した結果を、図 5、図 6 に示す．

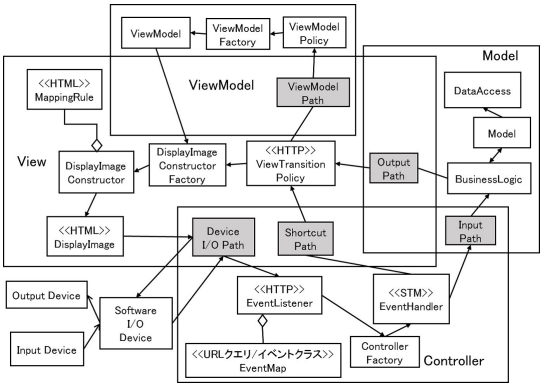


図 3 CSA-ISys の具象アーキテクチャ

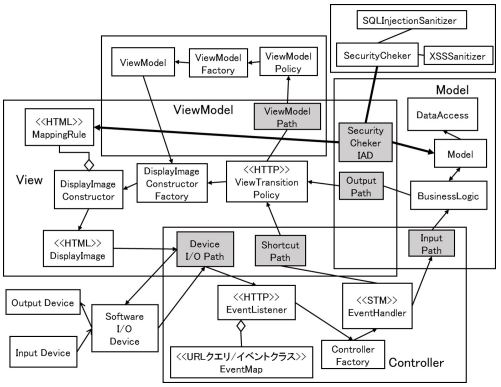


図 4 セキュリティ機能を追加した具象アーキテクチャ

図 5、図 6 に示す．

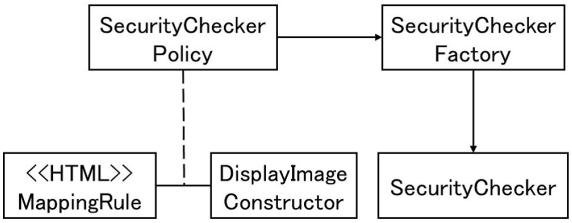


図 5 XSS の脆弱性への対策

XSS の脆弱性への対策の場合、*SecurityCheckerPolicy* が *Model* から *DataAccess* へのメッセージ通信を横取り

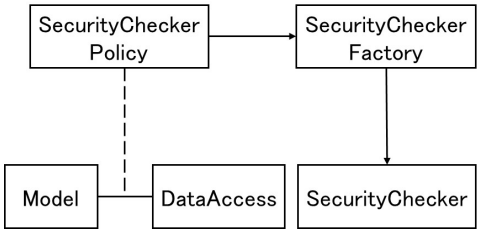


図 6 SQL インジェクションの脆弱性への対策

する．同様に、SQL インジェクションの脆弱性への対策の場合、*MappingRule* から *DisplayImageConstructor* へのメッセージ通信を横取りする．以降の構造は、XSS の脆弱性への対策と SQL インジェクションの脆弱性への対策の場合で共通している．*SecurityCheckerPolicy* が *SecurityCheckerFactory* を起動し、*SecurityCheckerFactory* が *SecurityChecker* のインスタンスを生成する．Command パターンで実装された *SecurityChecker* が、それぞれの脆弱性に対するセキュリティ機能を指定し、実行する．そして、脆弱性に応じた有害な文字列が無害化される．

## 4 事例検証

我々がセキュリティ機能を追加して再設計した CSA-ISys を用いて、Twitter を想定したウェブアプリケーションを開発した．アプリケーション開発において、JSP とサーブレットを用いている．開発したアプリケーションの主な機能として、ユーザが入力した文字列をデータベースに格納し、データベースから該当する文字列を検索するなどの操作ができる．以下、開発の際に選択した実現技術を示す．

- プログラミング言語：Java
- 具象表現形式：HTML5
- 外部との通信プロトコル：HTTP

再設計したアーキテクチャを検証する観点を、次に示す．

- 開発したウェブアプリケーションにおいて、追加したセキュリティ機能が正しく動作できているか．
- 開発したウェブアプリケーションが、アスペクト指向に基づいてセキュリティ機能を実装できているか．

検証した結果、上記の観点を満たしたウェブアプリケーションを作成することができたので、アーキテクチャの有用性を確認できた．

## 5 考察

CSA-ISys に基づいてアプリケーション開発を行う場合、*EventListener*、*BusinessLogic*、*ViewConstructor* などにセキュリティ機能を追加する必要があるが、セキュリティ機能に関するコードが煩雑になってしまった．再設計した CSA-ISys に基づいて設計すると、*Model* と *DisplayImageConstructor* に追加するだけで良いので、プログラムの煩雑さが解消できることがわかった．

PBR パターンと Command パターンを用いずにセキュリティ機能を実装する場合、セキュリティ機能をオブジェクトとして扱うことができなくなり、ウェブアプリケーションの各コンポーネントにセキュリティ機能のためのプログラムが散らばってしまうので、容易に追加することが難しくなる．再設計した CSA-ISys に基づいて設計すると、新しいセキュリティ機能を追加するとき、*SecurityCheckerIAD* を織り込むコンポーネントを指定することと、*SecurityChecker* に新しいセキュリティ機能のためのオブジェクトを追加するだけでよいので、セキュリティ機能を容易に追加できることが確認できた．

## 6 おわりに

XSS の脆弱性と SQL インジェクションの脆弱性は、現代において広く普及しているインタラクティブシステムにおける代表的な脆弱性である．これらの脆弱性対策がなされていないフレームワークやライブラリなどを用いてアプリケーションを開発する場合、セキュリティ対策に注意しながら開発する必要があるので、ソフトウェア開発の観点から非効率である．

本研究では、CSA-ISys にセキュリティ機能を追加することで再設計した．追加するセキュリティ機能が、アスペクト指向アーキテクチャを実装する段階でどのモジュールに横断的関心事として関わるのか検討した．我々が再設計したアスペクト指向アーキテクチャに基づいてアプリケーションを開発すれば、セキュリティ対策について特に考慮する必要がなくなる．また、アスペクト指向に基づいてセキュリティ機能を横断的関心事として捉えているので、セキュリティ機能のためのコードが煩雑にならない．

今後の課題は、XSS の脆弱性と SQL インジェクションの脆弱性以外の脆弱性へのセキュリティ機能の追加を行うことで、設計したアスペクト指向アーキテクチャを拡張することである．

## 参考文献

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides,“オブジェクト指向における再利用のためのデザインパターン”, SoftBank Creative, 1999.

[2] IPA，安全なウェブサイトの作り方 `http://www.ipa.go.jp/files/000017316.pdf`， 2015.

[3] IPA，セキュアプログラミング講座 `http://www.ipa.go.jp/files/000059838.pdf`，2016.

[4] IPA 技術本部 セキュリティセンター，JPCERT コーディネーションセンター： ソフトウェア等の脆弱性関連情報に関する届出状況 `https://www.ipa.go.jp/files/000060913.pdf`，2017.

[5] 江坂篤侍，野呂昌満，沢田篤史,“インタラクティブシステムのための共通アーキテクチャの設計” ソフトウェア工学の基礎 (日本ソフトウェア科学会 FOSE2017)，pp. 129-134，2017.