2005－SE－150（4）
2005／11／29

アスペクト指向によるセキュリティ実装について

千葉　岳史　　伊藤　恵

公立はこだて未来大学大学院　　公立はこだて未来大学

オブジェクト指向を使ったソフトウェア開発において，システム機能の中で，ロギングや同期処理のような複数のクラスを横断する処理がある．それを１つのオブジェクトとして扱う単位が Aspect であり，それを技術化したのが Aspect 指向技術である．その Aspect という概念を，ソフトウェア開発に適用し，開発の効率と品質を向上させようと考えている．本研究では，Aspect 指向を活用したセキュリティ実装の提案を行う．ソフトウェア開発の中でも，セキュリティにおける関心が高まっており，容易な改良と把握が求められる箇所である．Aspect 指向を用いる事によって，アクセス制御などのクラスを横断する処理を，Aspect として扱える．そのように実装する事によって，Aspect 指向技術のソフトウェア開発における利用を提案し，考察していく．

About Making Software Security with Aspect-Oriented Technology

Takefumi Chiba ,　Kei Ito

Graduate School of Future University-HAKODATE , Future University-HAKODATE

There are crosscutting concerns such as logging or synchronization in Object-Oriented Software Development. The Aspect concepts is to treat such those as one object. And Aspect-Oriented technology is using and technifying the Aspect concepts. We think apply the concept of the Aspect-Oriented technology to Software Development. This study proposes building security functions into Software with Aspect-Oriented technology. Software Development is ever-increasing importance of security functions and that needs easy system improvement and understanding. Using Aspect-Oriented technology, we treat as the Aspect concepts that crosscutting concerns such as access control. After that we examine using Aspect-Oriented technology in Software Development.

1 はじめに

近年，ソフトウェアシステムのオープンソース化が増加している．例えば，Java，C＋＋といったオブジェクト指向型のシステム構築の増加である．そして，オブジェクト指向で取り扱い難い問題がわかってきた．それは，システム基本機能の中で、複数のクラスを横断する挙動(これを横断的関心事という)を捉えるのが困難である事である。それに対応するために開発されたのが、Aspect 指向技術である[3,4]．つまり，Aspect という概念を用いて，従来の Java プログラムを，独立性・組立て・再利用性において向上させる技術である。ソフトウェア開発において，Aspect 指向技術を適用させる研究は，様々な研究が行われている[1,2,5,6].

本研究では，その Aspect 技術をソフトウェア開発に用いて，オブジェクト指向におけるソフトウェア開発の改善とそれに伴う考察を行うものである．ソフトウェア開発工程の中でも，最近，関心が高まっているセキュリティの構築に主眼を置いている．それは，アクセス制御やログイン ID 管理といった横断的関心事を持ち，ソフトウェアにおいて重要度が高くかつ常に更新される事を前提の箇所である．Aspect 指向を活用したシステムの提案も数多く出ているが，セキュリティにおける実証を行うことにより，さらなるソフトウェア開発を改善しうる可能性を提案できると考える．

本研究の流れは，まず小規模のシステム構築を行う．それは，オブジェクト指向のみの構築であり，ソフトウェアの簡潔なシステム構成を持つものとする．それに対して，Aspect 指向を用いたシステム構築を行い，前者とのシステム上の違いを考察，評価を行う．それによって，Aspect 指向における利点，欠点を考察する．そして，Aspect 指向におけるソフトウェア開発の利用例と有用性を確認し，今後のソフトウェア開発に役立てる一例を提案できると考える．

2　関連研究

Aspect 指向技術については，これまでの研究で，大まかに Aspect の検証，適用，発見の分野で研究されている．具体的には，複数の Aspect が絡み合った際に相互に干渉を与え，正しい動作を不能とする可能性を発見するための技術に関する研究や Aspect を用いて簡潔に記述できるような横断要素として何があるか[1]，Aspect を用いてその横断要素を記述することがどのような影響をもたらすかという研究がある[2]．また，Aspect として書くべきでない要素，書くことで保守性をかえって悪化させるような要素に関する研究，オブジェクト指向で書かれた従来のプログラム内に分散した横断要素を発見し，Aspect を抽出する作業を機械的にサポートするための研究といったことがあげられる．

例えば，河内らの研究[1]では，Aspect 指向の適用であり，石尾らの研究[2]では，適用と検証に当たると考える．どちらも Aspect 指向の利用によるプログラムの動作の向上を目指したものであると考えられる．それは，ソフトウェアの向上につながるものであり，本研究でも，同様にソフトウェアの品質向上に役立てるものとしたいと考えている．

3　検証するシステムの構築

3.1 Aspect 指向

システム構築にあたって，Aspect 指向技術について説明する．ここで使用していく Aspect 指向技術は，オブジェクト指向によるソフトウェア開発に利用するため，AspectJ[9]を使用している．そのため，AspectJ における基本概念や仕様に基づくものとする．

Aspect 指向の基本概念として，手続きや制約といった抽象化されたフレームワークを基にしたプログラミング言語は，最終的には複雑なシステムに対し適応できないとされている[12]．そして，プログラムされたシステムの挙動は様々な Aspect を持ち，それぞれが各々の形ともいうべきものを持っている．それを Aspect として捉え表現する事によって，複雑なシステムに対応させるようとするものである．具体的利用法としてよく引き合いに出されるものがあり，それはログアスペクトと言われ，発生したエラーをすべて記録したい，現在のネットワーク接続状態をチェックしておきたいといった動作を行うものである．

AspectJ の主な処理は，Aspect 仕様記述を Java 仕様記述に自動変換＆実行するものであり，Java 言語で記述されたプログラムと Aspect で構成されたプログラムを繋ぎ合わせるものである．

AspectJ における主な構成要素[12]として，「アスペクト」「ジョインポイント」「ポイントカット」「アドバイス」の 4 つが挙げられる。

はじめに，アスペクトであるが，Aspect 指向における意味では，横断的関心事を一まとめにしたものである．AspectJ では，ジョインポイントとアドバイスの組み合わせを指定するモジュール単位の事で，複数のポイントカットとアドバイスを指定できる．

次のジョインポイントは，プログラム実行時にアドバイスの実行を割り込ませることが可能なコード上の位置のことで，割り込み位置として，メソッド，コンストラクタの実行，フィールドへのアクセス，クラス静的初期化位置，インスタンス事前初期化位置 etc に限定される．

ポイントカットは，条件を指定することで，プログラム内に存在する全てのジョインポイントの集合から抽出される部分集合である．例えば，

call, execution(<メソッドパターン>)

get, set(<フィールドパターン>)

のように，前者は＜メソッドパターン＞が実行，または呼び出されたとき，後者は＜フィールドパターン＞が与えられた，セットしたときといった条件である．

最後にアドバイスは，指定されるポイントカットによって選択されるジョインポイントに差し掛かったときに実行されるコードの事である．それは

before(<コンテキスト定義>)

after(<コンテキスト定義>)

< --- >around(<コンテキスト定義>)

以上のように定義され，ポイントカットで指定された条件のときに，その状態から先に実行するのが before，あとで実行させるのが after であり，around は指定された部分に代わって処理を行うものである．

以下に示す図 1 は，以上で説明した 4 つの要素の関連図である．

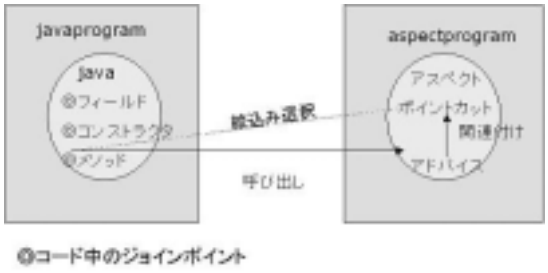

図 1. AspectJ の構成要素関連図

これら 4 つの要素は，AspectJ によって，どのように処理されているか．それは，Aspect 要素を含まない Java プログラムと Aspect によって記述されたプログラムが次の図 2 の様に混ぜ合わせられる[12].

つまり，記述された Aspect プログラムは，AspectJ のコンパイラによって，通常の Java プログラムへと変換されて，その処理が実行されている．AspectJ における Aspect は，記述方法が違うだけで，実際の処理は，以下の図 2 で示すように Aspect を含まない Java プログラムで記述されたものと同じ動作をする事になる．

これは，AspectJ にて記述できるプログラムは，全て Java で記述する事ができるという事であるが，まだ開発が行われている最中のシステムであるので，万能ではなく問題点もいくつかあるが後に記述する．

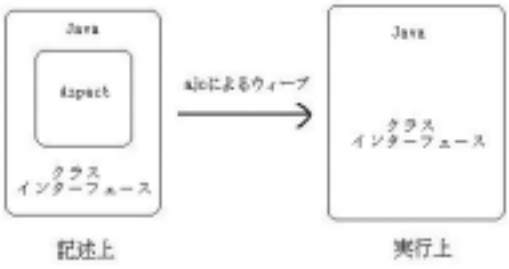

図 2. AspectJ によるウィービング

3.2 オブジェクト指向による実験システムの構築

評価基準とするオブジェクト指向によるシステム構築を行った．対象となるシステムは，Web アプリケーションとし，簡潔かつソフトウェアの根幹としての構成を持つように構築を行った．構築には，統合環境として eclipse[8]，アプリケーションサーバーとして Tomcat[10]，データベースとして MySQL[11]を利用した．

簡潔な Web アプリケーションを構築する事によって，ソフトウェアの規模や種類に捕らわれず，再利用性が高いものになると考える．また，今回はセキュリティに重点を置いているため，若干の構成は省略するものとして，構築を行っている．

今回実装される 1 例として，Web アルバム型アプリケーションを構築した．ここで参考にしたのは，北野 由希らによる著籍[13]よりプログラムを参考とした．

その構成されるシステムが持つ要素としては，ログイン，アクセス制御，データベース接続，簡潔なインタフェース，アップロード機能といった各々の処理である．それら要素は，アプリケーションの根幹とし得る部分である．そのシステム構成の概略は，以下の図 3 のとおりである．

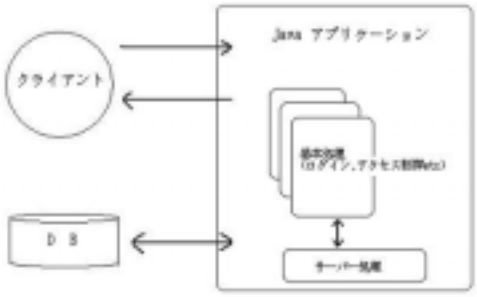

図 3. 基本システム構成図

オブジェクト指向によるシステム構築において，アクセス制御は，その処理ごとに記述を行わなければならない．今回参考にしたアプリケーションには，いくつかアクセス制御に対応するように処理を追加している．

3.3 Aspect 指向による実験システム構築

前述で構築したシステムに対して，Aspect 指向を用いたシステムの構築を行った．それは，オブジェクト指向を用いたシステムと基本的な構成要素は，同じ機能を持つという事であるが，Aspect 指向を利用する事によってシステム構築の改善を図るものである．

例えば，アクセス制御の処理の場合は，オブジェクト指向での実装では，各々のオブジェクトの処理に，細かく合わせながら記述を行わなければならないが，Aspect 指向を利用すると，必要なときに処理を呼び出す事によって，アクセス制御が可能になる．

この事の具体例が天野らの書籍[12]において，アクセス制御の示されている．これを，今回の Web アルバムのシステムに当てはめると，オブジェクトのみの構成では，

```
public class Album {
 public void Upload (FileItem item) {
  UPPermission permi =
   new UPPermission(item, userid);
  AccessCtrller.checkPermission(permi);
 ------
 }
}
```

以上のように，アルバムのアップデートの処理の箇所に直接アクセス制御を検査する記述をしなければならない．

しかし，Aspect 指向においては，この場合，アルバムの変更の処理が行われた時に実行するという条件で，処理をする事ができる．つまり，アルバムの操作箇所の記述とアクセス制御の記述を分離して，記述が可能になる．アクセス制御の部分だけを，記述しようとすると，

```
public aspect AccessCtl {
 pointcut AlbumCheck() :
   execution(void Album.Upload(FileItem));

  before(FileItem item) :
      AlbumCheck( ) &&args(item) {
       UPPermission permi =
         new UPPermission(item, userid);
       AccessCtrller.checkPermission(permi);
  }
}
```

という記述になる．これは，アルバムのア

ップデートが行われたときに，その変化が適用される直前にそのアップデートの許可があるかどうかをチェックするという処理になっている．実際の記述とは若干違うが，このように記述する事によって，アクセス制御処理が実装される．
以下の図 4 は，Aspect 指向を用いたシステムの概略図である．

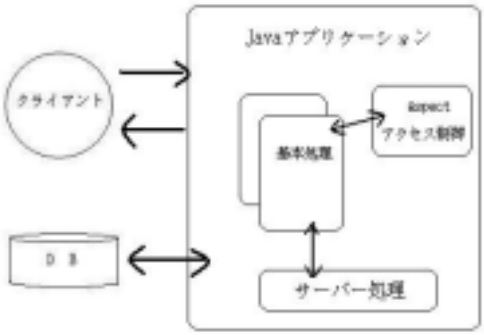

図 4. Aspect を適用したシステム図

以上の図のような構成をとり，前述のアップデートの例のように，このシステム構築は，アクセス制御の処理を分離して記述を行う．

このシステムには，アルバムのアップデートのほかに，削除や変更機能がある．同時に，それぞれがアクセス制限を行わなければ，アプリケーションとして成り立たないので，これらにも同様に適用している．

4 考察

対象となるシステム構築を一通り実装させたが，これらについて考察を行う．オブジェクト指向によるシステムと Aspect 指向によるシステムの両方を構築したわけだが，Aspect 指向技術が，システムのセキュリティにどの程度有効であるかを検討する．

そのシステムのセキュリティを行う処理を，Aspect 指向をもちいて行っているわけであるが，ここでの利点は，セキュリティ機能が 1 箇所まとめられている事である．それは，変更がたやすいという事でもある．セキュリティは常に，更新される事が前提となっているもので，変更のたびに，全てのセキュリティに関する部分を把握するのは，非常に大変な作業になる．ここで実装されたシステムのように，1 箇所であるなら，その作業量は明らかである．

しかし，実装した後は，そのように扱う事もできるが，実装する前は簡単にはいかない．そもそも，Aspect としてまとめるのは，容易ではないし，現状では，従来のように直接プログラムを記述するほうが，簡単に行える．

だが，今回はアクセス制御を完全に分離させたわけだが，実際のアプリケーションにおいて，完全に分離させる必要もない．

例えば，アクセス制御は，実装させているが，新たにアクセス制御を実装したいと考えた場合，その箇所に直接アクセス制御の処理を追加する事も可能であるが，そのアプリケーションを熟知していない限りは，容易な作業とはいえない．ここで，Aspect 指向を用いると，元のプログラムの全容を知っている必要はなく，追加したい箇所さえわかっていれば，その箇所に対応するように処理を追加できる．

Aspect 指向のみで実装させる利点は，確かにあるが，実際にそれを行おうとするのは，現実的と言えない．アプリケーショ

ンは，更新されることを前提として考えれば，追加していく処理を Aspect として実装する，つまりは，Aspect 指向を機能補助的に使用するというのが，より良い利用ではないだろうか．

よって，最終的には，アプリケーション開発において，アクセス制御のように分離すべき価値があるようなものは，Aspect として記述を行い，ほかの処理については従来の方法で実装を行う．そして，随時追加要素が出た場合は，Aspect にて実装を行うというのが，より良い開発方法と考える．以下の図 5 が，ここで提案するアプリケーションシステムの概要となる．

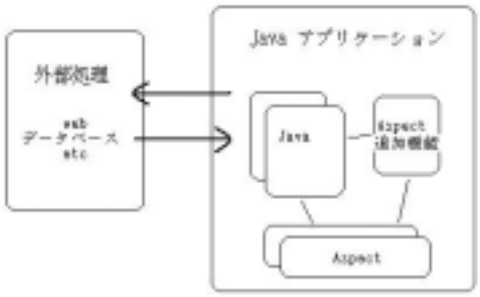

図 5. Aspect によるアプリケーション

また，この場合考えられる事として，Aspect 指向による追加機能が，ほかの処理機能と重なった場合，どちらが機能されるのか，またはどちらも機能されるか，されないかという点も注目されるだろう．その場合，アプリケーション機能が相殺され，機能しなくなるという問題が生じる可能性がある．それは，開発環境によってカバーが可能である範囲であるので，この研究では，対象とはしない．

5　今後の課題

Aspect 指向におけるアプリケーション開発において，問題点としては横断対象のクラスが揃わないとテストしにくいという事があげられる．これは，Aspect 指向が提案されたころから言われていることで，Aspect 指向の特性上からの問題である．これに対して，解決策が提案されれば，さらなる向上が見込まれるが，難しい問題だろう．

また，テストができるとしたら，Aspect を結合する前後でクラスをテストするべきか，いつテストを開始するべきか，Aspect だけをテストするようなイベント列を生成するテストケースは書けるのかという問題もある．

今回のセキュリティにおいては，ログイン時のパスワードやデータベースにアクセスするときの情報は，安全に管理されるべきであり，それらをアプリケーション内でも厳重に管理できれば，セキュリティの向上になるのではないかと考える．

また，アプリケーション機能としては，簡潔な要素であるので，随時検討し追加していく予定である．そして，考察におけるアプリケーション例にて開発を行った場合に伴って，さらなる Aspect 指向におけるセキュリティ実装への提案も行いたい．

今回の取り扱う範囲ではないが，この研究において使用した AspectJ であるが，これは eclipse プラグインのものを使用した．しかし，まだ開発が進められているものであるので，実用できる範囲がまだ狭いという問題がある．今後の仕様変更に伴い，さらなる検証を続ける予定である．

6 おわりに

オブジェクト指向によるアプリケーション構築と，それに Aspect 指向技術を適用したものの構築を行った．特にセキュリティにおける実装を中心に行ったわけだが，それらによって，Aspect 指向におけるセキュリティ実装の 1 例を提案した．Aspect 指向を用いる事によって，アクセス制御をより簡潔にまとめる事ができ，改良も行い易くなった．アプリケーション開発における，Aspect 指向を用いたセキュリティ実装の有用性を提案できたと考える．引き続き，セキュリティに対しての適用実験を行っていく予定である．

参考文献

[1] 河内一了,増原英彦: アスペクト指向言語におけるデータフローポイントカットの提案, http://www.graco.c.u-tokyo.ac.jp/ppp/projects/dflow/index-j.html

[2] 石尾 隆 ,楠本 真二,井上 克郎(大阪大学大学院基礎工学研究科): アスペクト指向プログラミングのプログラムスライス計算への応用, 情報処理学会論文誌，Vol.44，No.7, pp.1709-1719, 2003 年 7 月

[3] 吉見隆洋．Aspect-Oriented ProgrammingInfoTab: http://www.race.u-tokyo.ac.jp/~yoshimi/AOP/index-j.html

[4] AOPatDSL 97, Aspect Oriented Programming, Kiczales,G., Irwin,J.,Lamping,J.,Loingtier,J..M.,Lopes, C.,Maeda,C., Mendhekar,A.,DSL '97 - First ACM SIGPLAN Workshop on Domain-Specific Languages, 1997

[5] Miguel-P-Monteiro.Instit-Politecnico de-Castelo-Branco,Avenida-do-Empresario,Portugal-JoaoM. Fernandes .Universidade do Minho, Braga Portugal:Towards a catalog of aspect-oriented refactorings, Proceedings of AOSD 2005, pp.111-122, Chicago, Illinois, March 2005

[6] Jan Hannemann , Gail C. Murphy , Gregor Kiczales.University of British Columbia : Role-Based Refactoring of Crosscutting Concerns, Proceedings of AOSD 2005, pp.135-146, Chicago, Illinois,March 2005

[7] Java, http://www.java.com

[8] eclipse, http://eclipse.org/

[9] AspectJ, http://eclipse.org/aspectj/

[10] Tomcat, http://jakarta.apache.org/

[11] MySQL, http://www.mysql.com/

[12] 天野まさひろ,鷲崎弘宣,立掘道昭:AspectJ によるアスペクト指向プログラミング入門,ソフトバンクパブリッシング株式会社出版(2004)

[13] 石田 聡,北野 由希,杉之原 亮,谷口 寿俊,細畠 啓史,吉村 智史: Eclipse ではじめる Web アプリケーション開発,工学社出版(2005)