Vol.2010-SE-168 No.17
Vol.2010-EMB-17 No.17
2010/6/2

ディペンダビリティ確保にむけたアスペクト指向技術動向

鷲崎弘宜[†1,†2] 高橋竜一[†1] 村上真一[†1]
大橋昭[†1] 吉岡信和[†2] 石川冬樹[†2]
久保淳人[†3] 山本里枝子[†4] 小高敏裕[†4]
錠尚史[†5] 鹿糠秀行[†6] 杉本信秀[†7]

ソフトウェアの開発にあたり可用性や保守性，セキュリティに代表されるディペンダビリティを確保するために有効なアスペクト指向技術の研究ならびに実践の動向について，文献や会議を中心とした調査結果を報告する．本調査において技術の適用対象として，主として Web アプリケーション・エンタープライズアプリケーションを扱う．

A Report on Trends in Aspect-Oriented Software Development Technology for Ensuring Dependability

Hironori Washizaki,[†1] Ryuichi Takahashi,[†1]
Shinichi Murakami,[†1] Akira Ohashi,[†1]
Nobukazu Yoshioka,[†2] Fuyuki Ishikawa,[†2]
Atsuto Kubo,[?] Rieko Yamamoto,[†4]
Toshihiro Kodaka,[†4] Hisashi Ikari,[†5]
Hideyuki Kanuka[†6] and Nobuhide Sugimoto [†7]

We report a result of a brief survey on progress in researches and practices in aspect-oriented software development (AOSD) technology for software dependability including availability, maintainability and security. Web/enterprise applications are the main target of the survey.

1. はじめに

組込みからエンタープライズに至るまで，ソフトウェアが社会基盤としての役割を増しつつある．特に，インターネットや関連技術の拡充に伴う新たな Web 活用の形としての Web2.0 は，ビジネスやコミュニティ，技術等の様々な面で基盤として期待されている[1)]．しかしながらオープン性・分散性の高さや変化の早さ，要求や構成の多様さなどに起因して，信頼性，可用性，安全性，セキュリティ，保守性等を総合したディペンダビリティの確保が難しく，真に「頼れる」Web の開発技術確立が社会的急務である．

Web アプリケーション開発におけるディペンダビリティの確保には，基盤であるサーバやクライアント環境への仕組みの作りこみ，基盤上のアプリケーションへの作りこみ，および，基盤とアプリケーション間・同士の安全確実な連携が重要である．しかし，修正や拡張の必要箇所が多岐に渡る問題や，モジュール間の組み合わせの複雑さがあり，効率的な確保は容易ではない．その解決策として，モジュール群へと散らばる処理（横断的関心事）やモジュール間の接続を新たなモジュール（アスペクト）へと局所化し，後の適当な時点においてウィーバと呼ばれる合成器により自動合成するアスペクト指向プログラミング（Aspect-Oriented Programming: AOP）[2)] が注目されている．

AOP の適用により，ディペンダビリティをソフトウェアの開発中や実行時において効率よく作りこむことが可能になることが期待され，様々な研究ならびに実践が始められつつある．また，AOP の考え方をソフトウェア開発の上流から適用し，開発の要求定義から分析設計を通じて実装・テスト・保守に至るまで関心事を分離するアスペクト指向開発手法

†1 早稲田大学
Waseda University
†2 国立情報学研究所 GRACE センター
GRACE Center, National Institute of Informatics
†3 青山メディア研究所
Aoyama Media Laboratory
†4 富士通研究所
FUJITSU LABORATORIES LTD.
†5 とめ研究所
TOME R&D Inc.
†6 日立製作所
Hitachi, Ltd.
†7 東芝ソリューション
Toshiba Solutions Corporation

Vol.2010-SE-168 No.17
Vol.2010-EMB-17 No.17
2010/6/2

（Aspect-Oriented Software Development: AOSD，例えば[3]）についても同様に，開発早期におけるディペンダビリティ要求実現方法のモジュール化と分離合成に有効な可能性があり，幾らかの研究ならびに実践が始められつつある．

そこで本報告では，ソフトウェアの開発にあたりディペンダビリティを確保するために有効なアスペクト指向技術の研究ならびに実践の動向について，文献や会議を中心とした調査結果を報告する．本調査において技術の適用対象として，主として Web アプリケーション・エンタープライズアプリケーションを扱う．

2. アスペクト指向

家の建築において，設備は部屋を横断する．例えば図 1 において，水道や電気などの設備は，間取り（アーキテクチャ）に対してそれぞれ異なる形で配置される．ここで，設備を OHP シートに分けて書けば，業者ごとの個別作業が可能であり，さらには重ね合わせることで一貫した全体プランが得られる．

アスペクト指向開発とは，このように家の建築においてアーキテクチャに対し直交する設備群を分離設計して最後に合成するプロセスを，同様にソフトウェア開発において実現するものである．具体的には，要求のうちで本質はオブジェクト等の従来のモジュールにより分離し，一方，横断的関心事はアスペクトにより分離する．このようにアスペクト指向は，要求がしばしばソフトウェア構造（間取り・アーキテクチャ）に対して横断する性質を的確に扱う．

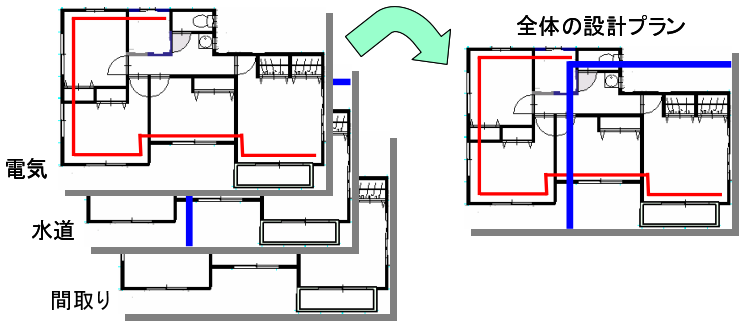

図 1 家の間取りと設備の直交性

2.1 アスペクト指向の広がり

アスペクト指向プログラミング（Aspect Oriented Programming: AOP）とは，要求を本質と横断的関心事の集合として分離して捉え，結合ルールによって 1 つのプログラムへと纏め上げるプログラミング手法である（図 2）．アスペクト指向ソフトウェア開発（Aspect Oriented Software Development: AOSD）とは，その AOP における仕組みを上流工程にまで応用したものである．

今日においてアスペクト指向が扱う領域は，要求工学からアーキテクチャ・モデリング，プログラミング，形式手法，実行基盤・ミドルウェアまで多岐にわたる．例えば要求工学におけるアスペクト指向の適用は Early Aspects と呼ばれ，要求段階における横断的特性の識別や表現を扱う．要求分析に特化した手法[26] や，設計まで含めた方法論[3),5)] などの提案がある．

2.2 アスペクト指向プログラミングの詳細

AOP の基本用語を，図 3 を用いて以下に説明する．

- アスペクト（Aspect，側面）：ポイントカットとアドバイスの組み合わせを指定するモジュール
- ジョインポイント（Joinpoint，結合点）：アドバイスの実行を割り込ませ可能なコード上の決まった位置
- ポイントカット（Pointcut，点の絞込み）：プログラム中の全ジョインポイント集合から，部分集合を得るための絞込み条件
- アドバイス（Advice，助言・挿入コード）：スレッドの実行が，ポイントカットによっ

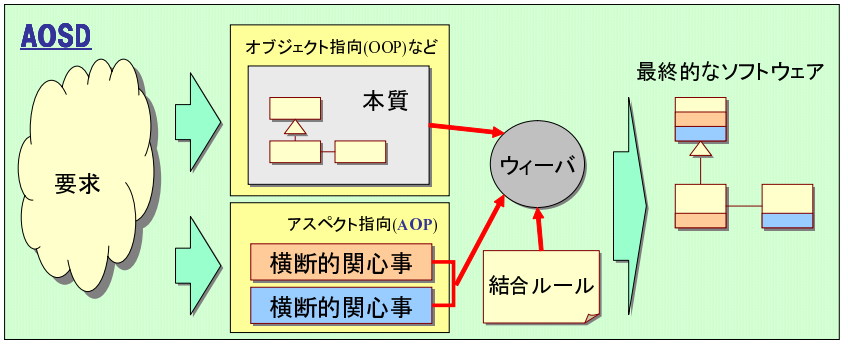

図 2 AOP と AOSD

Vol.2010-SE-168 No.17
Vol.2010-EMB-17 No.17
2010/6/2

て選択されたジョインポイントに到達したとき実行されるコード

- ウィーブ（Weave，織込み）：アドバイスを各モジュールに埋め込むこと

AOP では，プログラムの実行を割り込み可能なジョインポイントの連続として捉える．ジョインポイントは，例えばメソッドの呼び出しやフィールドの参照といった処理の時点である．それらの集合の中で，ポイントカットによりあらかじめ割り込みたい部分集合を指定しておき，その絞り込まれたジョインポイントにスレッドの実行が到達した際に実際に割り込ませる処理を，アドバイスとして記述する．そのポイントカットとアドバイスの対応付けを格納するモジュールの単位がアスペクトである．

3. ディペンダビリティ

ディペンダビリティ（Dependablity）の定義は文脈に応じて様々にあるが，狭義には可用性や耐故障性を含む実行サービスの正しさに関する品質であり，広義には，可用性およびその影響要因としての信頼性や保守性等の能力や程度を意味する[6]．従ってディペンダビリティは実行時における信頼性を広く捉えるため，(機密性や保全性等の）セキュリティがディペンダビリティの一部として捉えられることも多い．

本稿では，広義のディペンダビリティを扱い，ソフトウェアシステムの信頼性や可用性，セキュリティ，保守性に代表される実行時の品質や影響を受ける品質，関連する品質を特に扱う．それぞれの定義は例えば下記のように捉えられる．

- 信頼性: 指定された条件下で利用するとき，指定された達成水準を維持するソフトウェア製品の能力[7]
- 可用性: ソフトウェア製品が正常に稼働し続ける能力，確率[8]
- セキュリティ: 許可されていない人又はシステムが情報又はデータを読んだり，修正したりすることができないように，及び許可された人又はシステムが情報又はデータへのアクセスを拒否されないように，情報又はデータを保護するソフトウェア製品の能力[7]
- 保守性: 修正のしやすさに関するソフトウェア製品の能力[7]

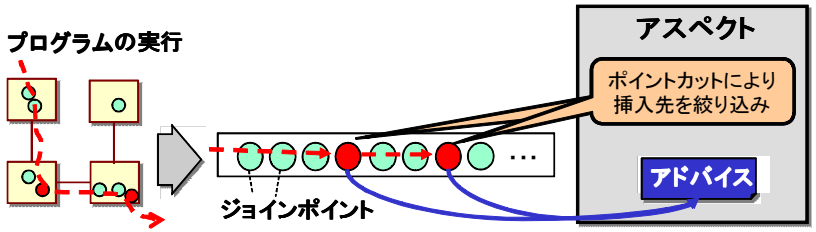

図 3 AOP の基本用語と関係

上述のようにソフトウェアが社会基盤としての役割を増すに従い，そのディペンダビリティの確保が社会的急務となりつつある．

例えば Web アプリケーション開発におけるリッチクライアント開発に限定してみても，2008 年 11 月〜2009 年 2 月において企業における技術者・研究者 30 名に開発上の課題をアンケート調査したところ[9]，図 4 に示すように上述の品質特性の確保が課題として認識されていることが分かった[⋆1]．

4. ディペンダビリティ確保に向けたアスペクト指向

我々は，そのアスペクト指向技術による効率的な確保に焦点を絞り，同技術の研究や実践の発表機会として主要な国際会議 International Conference on Aspect-Oriented Software Development における 2010 年およびこれまでの発表を中心として，動向を調査した．

調査にあたり，特定期間内における関連発表の網羅性を目指さないものの，複数の整理軸・次元に基づく分類により，代表的な取り組みの分布の状況（さらには今後の方向性）を提示することを試みた．具体的には，主に扱う品質特性，および，開発・運用のライフサイクルプロセスにおいて主に用いられる工程の 2 軸を持って，各技術や手法を図 6 に示すよ

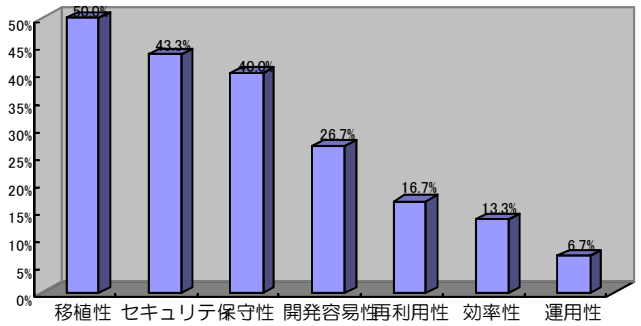

図 4 リッチクライアント開発における品質課題（有効回答数 30）

⋆1 アンケート調査時点においては信頼性・可用性を選択肢として挙げていなかった．

Vol.2010-SE-168 No.17
Vol.2010-EMB-17 No.17
2010/6/2

うに分析整理した．図 6 において各ラベル付き楕円は，本稿の以降における各技術・手法を示している．

整理により判明した事柄を以下にまとめる．

- ディペンダビリティ確保に応用可能な多数の代表的なアスペクト指向技術はプログラムを対象とし，従って用いる工程として構築（プログラミング）以降に位置づけられる．
- 開発の上流工程における取り組みは手薄であり，今後のさらなる拡充が期待される．特に，プログラミングレベルの技術と要求を結びつける手法や開発方法論については黎明期との印象であり，さらなる発展が望まれる．
- 扱う品質特性の広がりとして，もともとは保守性を扱う技術が多数であり，年月とともにセキュリティおよび信頼性・可用性へと広がりつつある．2008 年度において，リッチクライアント開発に絞って AOP の実務への導入目的をアンケート調査した結果として，AOP 導入者 11 名について図 5 に示すように保守性について一定の導入が見られるものの，セキュリティについては未導入であった．この調査結果は，上記の方向性の考察を幾らか裏付けるものと考えられる．

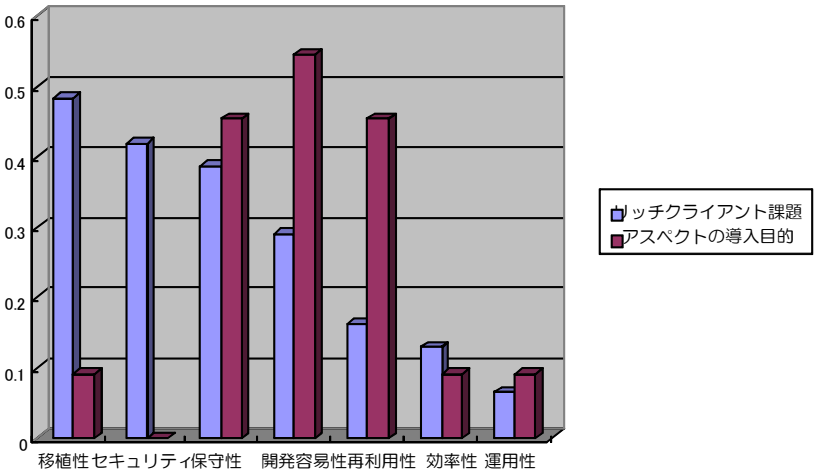

図 5 リッチクライアント開発における AOP 導入目的（有効回答数 11，ただし「リッチクライアント課題」については有効回答数 30）

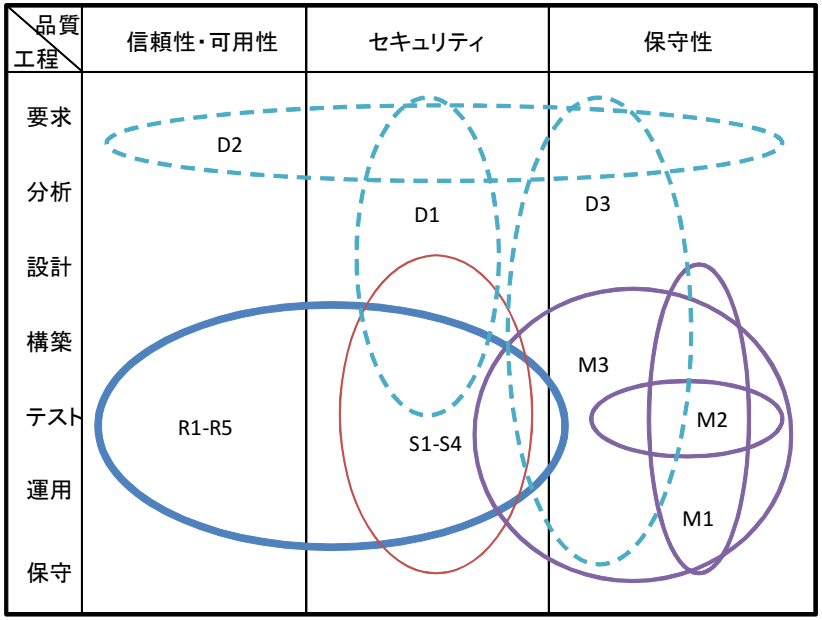

図 6 アスペクト指向技術の整理結果

4.1 実行基盤による可用性や信頼性向上

プログラミングシステムやミドルウェア，フレームワーク等の実行基盤を用いて信頼性・可用性を向上させる取り組みを以下に紹介する．

4.1.1 R_1: 拡張性の高いコンポーネント間接続

千葉らはアスペクト指向ならびに依存性注入（Dependency Injection）の応用として，プログラムを修正せずにコンポーネントの組み合わせを変更可能とするプログラミング環境 GluonJ を提案している[10]．同環境を用いることで，例えば過負荷時のディペンダビリティを考慮せずに開発された Web アプリケーションに対して，その修正なく，ディペンダビリティを高める機能を提供するコンポーネント（例えばスケジューラ）を組み入れられる．

4.1.2 R_2: スケジューリング

光来らは，スケジューリングを実現するコードを JavaEE サーバアプリケーションへと自動挿入する QoSWeaver を提案し，もともとディペンダビリティを考慮していないアプリ

Vol.2010-SE-168 No.17
Vol.2010-EMB-17 No.17
2010/6/2

ケーションをディペンダブルとすることを実現している[11]．

4.1.3 R_3: 分散環境下で拡張性をもたらす動的配備

分散環境における動的なアスペクトのデプロイにあたり，従来のメソッド実行等に加えどのサーバで実行されたか等の分散環境特有のジョインポイントを取り扱う必要がある．また，柔軟性や再利用のためには動的なデプロイが有利である．しかし，従来分散環境を取り扱うアスペクト指向技術は何れも静的にデプロイされるため，複雑なポイントカット定義（静的なデプロイはアスペクトがグローバルに伝播することで条件を厳しくする必要が出てくるため）等が必要になり再利用性が損なわれる．そこで Tanter らは動的にデプロイされる分散アスペクトを表現できるスコーピング（ジョインポイントの集合）を提案している[12]．

4.1.4 R_4: 実行時診断，エラー回避

Hohenstein らはアスペクト指向技術のデータベース分野への適用を通して，サードパーティアプリのバグの修正やオープンソースではないツールのバグの位置を特定する診断に取り組んでいる[13]．また，Java では JDBC の利用によりデータベース実装を隠蔽することができるが，一方で各データベースが独自にサポートするチューニングやフェイルオーバの扱いが統一的にできない課題がある．その場合に，データベース実装に依存する部分をアスペクトとして分離し，再利用可能な形で管理する方法を示している[13]．

4.1.5 R_5: 実行時修正

伊三野らは，アスペクトの動的なウィーブにより，高い信頼性や可用性が求められるアプリケーションについて，不具合の発生時においてその停止なく，応急処置を施すアスペクトを適用する仕組みを提案している[14]．

4.2 実行基盤や開発環境によるセキュリティ

ミドルウェア等の実行基盤やツール等の開発環境を用いて主にセキュリティを向上させる取り組みを以下に紹介する．

4.2.1 S_1: データフローに基づくセキュリティ関心事の表現

データフローポイントカットはデータの発生起源に基づくポイントカットであり，セキュリティ上の関心事の分離，特にプロセスインジェクション，ログフォージング，パスインジェクション等の検出に有効であることがすでに示されている．Alhadidi らは λ calculus に基づくフレームワークにより，データフローをスタティックに記述できる方法を開発した[21]．また，それが従来の動的な記述と矛盾のないことを示している．

4.2.2 S_2:セキュリティ強制

デスクトップクライアントアプリはローカルとリモートの双方へのアクセスが可能であるため，セキュリティ上のリスクも存在する．AOP はこのようなセキュリティ関連の関心事の分離に利用されるが，従来はサーバサイドのセキュリティへの取り組みが中心である．そこで Cannon らは，デスクトップクライアント上で最小権限の原則（Principle of Least Privilege）を強制する Authority Aspects を提案している[22]．具体的には Java 開発において，セキュリティマネージャを利用してアプリケーションに権限を付与するコンポーネント Authority Aspects を提案している．

4.2.3 S_3:セキュリティポリシー記述とチェック

ファイル操作やトランザクション処理といったセキュアなコードが必要とされる部分において，具体的にコードが満たすべき数々のルールをセキュリティポリシーと捉えると，その多くは横断的関心事となるためアスペクトとして既述されることが望ましい[23]．そこで Jones らは，XML でセキュリティポリシーを記述する独自の言語 SPoX（Security Policy XML）を開発している．SPoX は AspectJ の拡張として設計されており，AspectJ によるジョインポイントの検出を利用して，対象となる Java コードに SPoX で記述されたセキュリティポリシーがどの程度満たされているかを調べられる．

4.2.4 S_4:セキュリティポリシー記述と強制

AspectKE*は，タプル空間に基づく最初のアスペクト指向プログラミングシステムであり，klava と呼ばれるタプル空間システムのアスペクト指向拡張である[24]．タプル空間システムでは，プログラムはプロセスとタプル（データ）を持つノードで構成され，ノード間でプロセスによるタプルのやりとりが行われる．

AspectKE*では，アスペクトはグローバルな領域においてすべてのプロセスを監視するアクティビティとして定義される．ポイントカットでは，ノードやアクション (タプルの read など) の指定を行い，プログラムがその命令に到達すると，アスペクトはそのアクションを捕捉する．アクションを捕捉した後には，対象となるプロセスが持つ残りのアクションを解析する．解析には特定の関数が定義されており，例えば，beused 関数は指定したタプルが残りのプロセスにおいて使われるかの真偽値を返す．アドバイスでは，関数を組み合わせて条件を設定し，対象となるプロセスの実行を続けるか停止するかを指定できる．

AspectKE*は，信頼性のないノードに対してセキュリティポリシーを追加することを想定しており，例えば，あるタプルにパスワードを保存した後に，そのタプルが特定のノード以外から参照されることが解析された場合，プロセスの実行を停止するといった利用を著者らは想定している．このようにプロセスの将来の振る舞いを扱えることによって，柔軟なセキュリティポリシーの追加が可能であると提案者らは主張している．

Vol.2010-SE-168 No.17
Vol.2010-EMB-17 No.17
2010/6/2

4.3 開発環境による保守性向上

ツールを含む開発環境（や実行基盤）により，変更容易性や拡張性，テスト容易性といった主に保守性の向上を目指す取り組みを以下に紹介する．

4.3.1 M_1: 変更容易性・拡張性をもたらすコード自動生成系

Spring Roo は，Spring によるシステム開発を行う際に必要となる基幹的な Java のコードや XML による設定ファイルを自動生成する開発補助ツールである[18),19)]．具体的には Roo シェルと呼ばれる CUI 環境で hint コマンドを使いながら対話的にコマンドを入力していくことで，Entity クラスの生成に伴い，DB テーブルの自動生成や DB との連携に必要な Java コードや設定ファイルを自動で得ることができる．Spring Roo は生産性の向上と仕様変更や機能追加にも耐えうる継続的な開発を容易にすることを目的としている．

Spring Roo は，従来の Java による Web アプリケーションのソース自動生成ツールとは異なり，ソースを自動生成した後も，モデルクラスの変更に対応してビューやコントローラクラスを自動で再作成する．この仕組みは Spring Roo が内部で利用している AspectJ[2)] のインタータイプ宣言の機能によって実現されており，クラウド時代の Web システム開発におけるアスペクト指向技術の貢献の 1 つの形と捉えることができる．

4.3.2 M_2: テスト容易性向上のためのページ生成

JavaEE による Web アプリケーション開発において JSP（Java Server Pages）から生成される HTML/Web ページでは，意味情報が消失しテストが困難になる問題がある．そこで小高らは，JSP Web アプリケーションに対して，意味情報を Web ページに残すように AspectJ を用いて自動拡張することで解決を図っている[20)]．

4.3.3 M_3: 汎用プログラミング環境の応用

上述以外にも，Web アプリケーションプログラムやスクリプト言語等に特化した様々な汎用アスペクト指向プログラミング環境が提案されており，それらをサーバ側およびクライアント側におけるディペンダビリティ機能の効率的な追加と変更，拡張に応用できる可能性がある．例えば外村らは，ページ遷移やページリクエストといった Web アプリケーション固有のポイントカットをまとめた枠組み AOWP を提案している[15)]．

鷲崎，久保，大橋，村上らは，JavaScript 用 AOP 環境として AOJS を提案し[16)]，その応用として Web アプリケーションの高い変更容易性を実現可能なことを示している[17)]．

4.4 上流からの開発手法

アスペクト指向開発（AOSD: Aspect Oriented Software Development）とは，要求定義から分析，設計，実装，テストに至るまで，アスペクトによってソフトウェアシステムを開発する全体的な手法のことであり，上述のような従来の開発における横断的関心事の問題を解決する．AOSD は単なる AOP ではなく，モジュール化をうまく行えるようにするあらゆる範囲の技法を網羅する．また AOSD は，既存の技法（例えばオブジェクト指向，コンポーネントベース開発，デザインパターン，J2EE，.NET）とは競合せず，これらの技法の上に位置付けられる，もしくは補完するものである．

AOSD の種類としては，サブジェクト指向を利用したものや，オブジェクト指向の親和性を重視したもの，ユースケースによるアスペクト指向開発手法[7)] などがある．関連して，”Early Aspects” と称して要求工学やアーキテクチャ設計といった開発の早い段階におけるアスペクトの識別や適用が取り組まれつつある．

以降において，AOSD のディペンダビリティ確保に向けた取り組みを紹介する．

4.4.1 D_1: セキュア・アプリケーション開発手法

従来のソフトウェア開発手法はセキュリティの観点が不足しがちであった．また，セキュリティ独自の手法は各開発工程（要求分析，設計，実装，テスト）に閉じており，工程を通じた開発プロセスには成っていない．そこで[25)] では，ソフトウェア開発に関わる要求の中からセキュリティ関心事を分離して開発を行うアスペクト指向開発プロセス，および，各開発工程においてその実践を支援する要素技術からなるフレームワークが提案されている．ケーススタディでは，Web のブックストアのアプリケーション開発を題材に，ユーザに対するアクセス制御と SQL インジェクション攻撃対策をセキュリティ関心事に取り上げて，提案フレームワークの妥当性を確認している．

述べられている各開発工程における要素技術の特徴的な点は以下の通りである．

(1) 要求分析工程: セキュリティ要求の記法ミスユースケースを拡張した記法，Web アプリケーションにおける脅威や対策をパターン化した Web セキュリティパターンを用いる点
(2) 設計工程: 依存性注入に基づくセキュリティ関心事横断点の設計手法と制約の導入，セキュリティ設計パターンを用いて再利用を促進する点
(3) 実装・テスト工程: ソースコードの自動生成，ライブラリ化による再利用を推進する点

4.4.2 D_2: ゴール指向によるディペンダビリティ要求分析

機能要求と非機能要求の関連性を特定するために関係情報などが詳細化されたゴールモデル（要求分析モデル）上で，構造等に基づいて将来のアーキテクチャや実装におけるアスペクトを特定する手法が提案されている[26)]．提案されるゴール指向手法はディペンダビリ

Vol.2010-SE-168 No.17
Vol.2010-EMB-17 No.17
2010/6/2

ティ要求分析に特化したものではないが，例えばセキュリティといったディペンダビリティ要求が特定された場合に，実装においてコードクローン（例えば SSL 通信コードの散らばり）として横断的に実現されていることを実験により確認している．

4.4.3 D_3: ユースケースによるディペンダビリティ要求分離

Jacobson らは，非機能要求を機能要求へと転化させた形で初期段階よりモジュール化し，実装やテストに至るまでモジュール性を維持するユースケースによるアスペクト指向開発手法[3] を提案している．同手法は，「アスペクトは要求レベルでユースケースとして表現できる」ことを基本方針とする．同手法は，既存のオブジェクト指向開発方法論と親和性が高く，ユースケースに駆動される形で，要求を段階的に実装へ落とし込むことが可能である．ここで，アスペクト指向の考え方とユースケース駆動開発は概念が似ているため，後者からのシームレスな移行が可能である．具体的には，ユースケースをアスペクトとして捉えて，ユースケースを実現するクラスの集合の一部がアスペクトとなる．

つまりユースケースによるアスペクト指向開発手法は，機能・非機能要求の両方をユースケースとして捉え，各実現をクラスとアスペクトで分離して実装，最後に纏め上げる開発手法であり，アスペクトの扱いに特化した独自の UML 拡張記法を含む（図 7）．同手法の適用により，関心事の早期分離が達成され，拡張性・保守性の向上を期待できる．

Jacobson らは同手法の応用として，セキュリティ要求を機能要求へと転化させた結果としてのユースケース「認可処理をする」を例として，アプリケーション独立に再利用可能な基盤ユースケースとして表現し，分析，設計，実装へとアスペクト指向モデリングからアスペクト指向プログラミングに至るまでモジュール性を維持して最終的に合成する様子を示している．

5. おわりに

本稿では，文献や発表を中心として代表的な技術を取り上げる形で，ソフトウェアのディペンダビリティ確保に有効なアスペクト指向技術の研究ならびに実践動向の調査結果を報告した．

調査により，プログラミングレベルにおける取り組みの充実や，保守性からセキュリティ，信頼性へと至る技術適用の流れの一方で，開発の上流やプログラミングへと至る全体的な手法・方法論について取り組みが活発ではなく，さらなる拡充が期待されることを明らかとした．調査結果を受けて我々は，開発上流からのディペンダビリティ要求の識別とパターン化によるアスペクト指向開発の適用について方法をまとめつつある[27]．

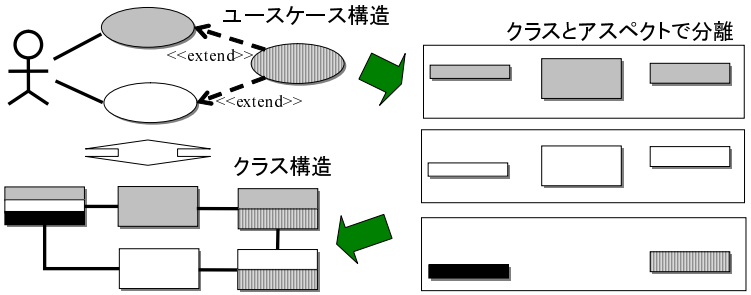

図 7 ユースケースに基づくアスペクト指向開発

今回の調査はアプリケーションレベルのアスペクト指向ソフトウェア技術に焦点を絞ったが，ディペンダビリティの確保にはアスペクト指向に限らず他のモジュール化・関心事の分離を促進する種々のソフトウェア技術が応用可能と想定される（例えばコンテキスト指向[28] など）．さらに，特にセキュリティや信頼性といった実行時品質については，その確保にあたりアプリケーションソフトウェアに限らず，その基盤や環境を与える OS やハードウェア，ネットワーク，システム全体における取り組みが重要である．例えばネットワークシステム（特にクラウドコンピューティング）におけるセキュリティ向上のための研究動向については堀らの調査[29] が参考となる．

謝辞

本調査研究は，財団法人情報科学国際交流財団 SSR 産学戦略的研究フォーラム 2009 年度の助成を受けて，研究課題「効率的なディペンダブル Web 開発に向けたアスペクト指向ソフトウェア技術の調査研究」の枠組みにおいて実施されました．

参 考 文 献

1) T. O'Reilly, Web 2.0：次世代ソフトウェアのデザインパターンとビジネスモデル, CNET Japan, http://japan.cnet.com/column/web20/story/0,2000055933,20090039,00.htm
2) G. Kiczales, et al., "An overview of AspectJ," ECOOP, 2001.
3) I. Jacobson and P.W. Ng, "Aspect-Oriented Software Development With Use Cases," Addison-Wesley, 2005 (鷲崎弘宜ほか訳: "ユースケースによるアスペクト指向ソフトウェア開発")
4) Yijun Yu, Julio Cesar Sampaio do Prado Leite, John Mylopoulos, "From Goals to

Aspects: Discovering Aspects from Requirements Goal Models," RE 2004: 38-47.
5) Elisa Baniassad and Siobhan Clarke, "Theme: An Approach for Aspect-Oriented Analysis and Design," 26th International Conference on Software Engineering (ICSE'04), 2004.
6) IEC, IEV 191-02-03, Dependability and quality of service / Item related performance, 1990.
7) ISO/IEC 9126-1:2001 Software engineering – Product quality – Part 1: Quality model （JIS X 0129-1: ソフトウェア製品の品質 – 品質モデル）
8) Linda M. Laird and M. Carol Brennan 著, 野中誠, 鷲崎弘宜 訳, "演習で学ぶソフトウエアメトリクスの基礎 ソフトウエアの測定と見積もりの正しい作法", 日経 BP 社, 2009.
9) 鷲崎弘宜ほか, SSR 産学戦略的研究フォーラム 2008 年度, Web2.0 におけるリッチクライアント開発のためのアスペクト指向技術の調査研究, http://www.washi.cs.waseda.ac.jp/SSR2008.html
10) Shigeru Chiba and Rei Ishikawa, "Aspect-Oriented Programming beyond Dependency Injection," ECOOP 2005 – Object-Oriented Programming, LNCS 3586, 2005.
11) Kenichi Kourai, "Aspect-oriented application-level scheduling for J2EE servers," 6th international conference on Aspect-oriented software development, 2007.
12) Eric Tanter, et al., "Expressive scoping of distributed aspects," 8th ACM international conference on Aspect-oriented software development, 2009.
13) Uwe D.C. Hohenstein, et al., "Using aspect-orientation in industrial projects: appreciated or damned?," 8th ACM international conference on Aspect-oriented software development, 2009.
14) 伊三野直志, 山本哲男, "動的アスペクト指向プログラミングを利用する応急処置システム", ソフトウェアエンジニアリングシンポジウム 2008, 2008.
15) 外村慶二，成瀬龍人，塩塚大，白石卓也，鵜林尚靖，中島震, Web アプリケーション開発向け AOP 機構の実装, 情報処理学会第 163 回ソフトウェア工学研究発表会, SIG-SE-163-9, 2009.
16) Hironori Washizaki, Atsuto Kubo, Tomohiko Mizumachi, Kazuki Eguchi, Yoshiaki Fukazawa, Nobukazu Yoshioka, Hideyuki Kanuka, Toshihiro Kodaka, Nobuhide Sugimoto, Yoichi Nagai, Rieko Yamamoto, "AOJS: Aspect-Oriented JavaScript Programming Framework for Web Development", Proc. 8th AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS 2009), ACM Press, 2009.
17) Shinichi Murakami, "Modularization of Extended Requirement by Use of Aspect in Rich Internet Application that Uses Database," 9th Annual Aspect-Oriented Software Development Conference (AOSD.10), Students Poster, 2010.3.
18) Rod Johnson, "AOP in the wild: Killer applications and why AOP is perfect for the cloud," 9th Annual Aspect-Oriented Software Development Conference (AOSD.10), 2010.
19) Spring Roo, http://www.springsource.org/roo
20) 小高敏裕, 上原忠弘, 片山朝子, 山本里枝子, "アスペクト指向を利用した Web アプリケーションテストの自動化", 情報処理学会 155 回ソフトウェア工学研究会, 2007.
21) Dima Alhadidi, et al., "The Dataflow Pointcut - A Formal and Practical Framework," 8th ACM international conference on Aspect-oriented software development, 2009.
22) Brett Cannon, et al., "Enforcing security for desktop clients using authority aspects," 8th ACM international conference on Aspect-oriented software development, 2009.
23) Micah Jones and Kevin W. Hamlen, "Disambiguating Aspect-Oriented Security Policies", 9th Annual Aspect-Oriented Software Development Conference (AOSD.10), 2010.
24) Fan Yang, Hidehiko Masuhara, Tomoyuki Aotani, Flemming Nielson, and Hanne Riis Nielson, "AspectKE*: Security Aspects with Program Analysis for Distributed Systems," 9th Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS 2010), pp.27-31, 2010.
25) 大久保隆夫：アスペクト指向によるセキュアなアプリケーション開発環境の提案，情報処理学会研究報告，Vol2009-SE-164 No.9，(2009).
26) Y. Yu, J.C.S.P. Leite and J. Mylopoulos: From goals to aspects: discovering aspects from requirements goal models, Proc. 12th IEEE International Requirements Engineering Conference (RE'04), 2004
27) 鷲崎弘宜ほか, SSR 産学戦略的研究フォーラム 2009 年度: 効率的なディペンダブル Web 開発に向けたアスペクト指向ソフトウェア技術の調査研究, http://www.washi.cs.waseda.ac.jp/SSR2009.html
28) Robert Hirschfeld, et al., "Context-oriented Programming," Journal of Object Technology, vol.7, no.3, 2008.
29) 堀良彰ほか, "クラウドコンピューティングにおけるセキュリティ研究動向", 情報処理学会研究会報告, 第 47 回コンピュータセキュリティ研究発表会, CSEC47, 2009.