２０１０年度修士論文

要求記述をベースとしたアスペクト生成言語の実現

早稲田大学基幹理工学部

情報理工学専修

村岡研究室　修士２年

5108B089-8

豊沢　泰尚

概要

本研究ではゴール指向とアスペクト指向を統合する方法を提案する。

ゴール指向要求工学では、機能要求に対するビジネス上の目的を明確にすることで、顧客やシステム利用者などの利害関係者との相互理解を容易化することができる。アスペクト指向プログラミングでは、「横断要素」と呼ばれるオブジェクトにまたがる処理を、これらのオブジェクトごとに記述するのではなく、「アスペクト」としてまとめて記述する。アスペクト指向要求工学はこのようなコードレベルの「アスペクト」に対応する要求レベルの「アスペクト」に基づいて要求を抽出・分析する。アスペクトは設計レベルの概念であり、ビジネスに関するゴー ルと相補的な関係にあると考えられている。

アスペクト要求とゴール要求分析の構成要素を対応付けることで、ゴール要求の記述によって表される要求をアスペクトとして実装することができる機能を提案する。

そしてその実現のために、ゴール要求とアスペクトの実装を統合する言語 GRD を開発した。

Abstract

I suggest a method to integrate an aspect point with a goal point in this study.

In the goal point demand engineering, I can make mutual understanding with the parties intersted such as a customer or the system user simpleness by making a purpose in the business for the function demand clear. By the aspect point programming, I describe processing to sit astride an object called "a crossing element" as "an aspect" not what I describe every these objects in a mass. The aspect point demand engineering extracts a demand based on "the aspect" of the demand level corresponding to "the aspect" of such a cord level and analyzes it. The aspect is a concept of the design level, and it is thought that I have a complementary relation with a goal about the business.

I suggest the function that can implement the demand that is expressed by the description of the goal demand because correspondence attaches a componentry of an aspect demand and the goal demand analysis as an aspect.

And, for the realization, I developed language GRD which integrated the implementation of the aspect with a goal demand.

目次

図表目次

第１章 序論

1.1 研究背景

アスペクト指向プログラミングでは、「横断要素」と呼ばれるオブジェクトにまたがる処理を、これらのオブジェクトごとに記述するのではなく、「アスペクト」としてまとめて記述する。アスペクト指向要求工学はこのようなコードレベルの「アスペクト」に対応する要求レベルの「アスペクト」に基づいて要求を抽出・分析する手法である。アスペクトは設計レベルの概念なので、ビジネスに関するゴー ルと相補的な関係にあると考えられている。

そこで本研究ではゴール指向とアスペクト指向を統合する方法としてアスペクト要求とゴール要求分析のの構成要素を対応付けることで、ゴール要求の記述によって表される要求をアスペクトとして実装することができる機能を提案する。

第２章 システムの概要

2.1 動作環境

本システムの動作環境を以下に示す。

ハードウェア環境

OS	WindowsXPSP2
CPU	Intel PentiumM 1.2GHz
RAM	1024MB

ソフトウェア環境

サーバサイド制御	PHP5.2
HTTP サーバ	Apache2

図 1　システムの動作環境

2.2 システムの概要

本システムはアスペクト要求とゴール要求分析の構成要素を対応付けることで、ゴール要求の記述によって表される要求をアスペクトとして実装することができる機能を提供する。

ゴール要求とアスペクトの実装を統合する言語 GRD を用いてソフトゴールに関して記述し、GRD コンパイラはそこからアスペクトコードを生成する。これを AOP モジュールが織り込み対象となるソースコードへ織り込み処理を行うこととなる。

NFR フレームワークの考え方をもとに、ゴール要求に関するグラフを比較的自然言語に近い形で記述することができるように言語設計を行った。

本研究では対象言語として PHP を使用する。そのため、アスペクトの織り込みを実現するための AOP モジュールには PHP 上で動作する AOWP を使用している。

今研究で開発する言語のコンパイラを実現するために、PHP 上で動作する PHPSimpleCC というコンパイラコンパイラを開発した。これは PHP ネイティブで動作し、より高い可用性を求めて可読性の高いコードを出力することを目標とした。

システムの処理に関するフォローチャートを以下に示す。

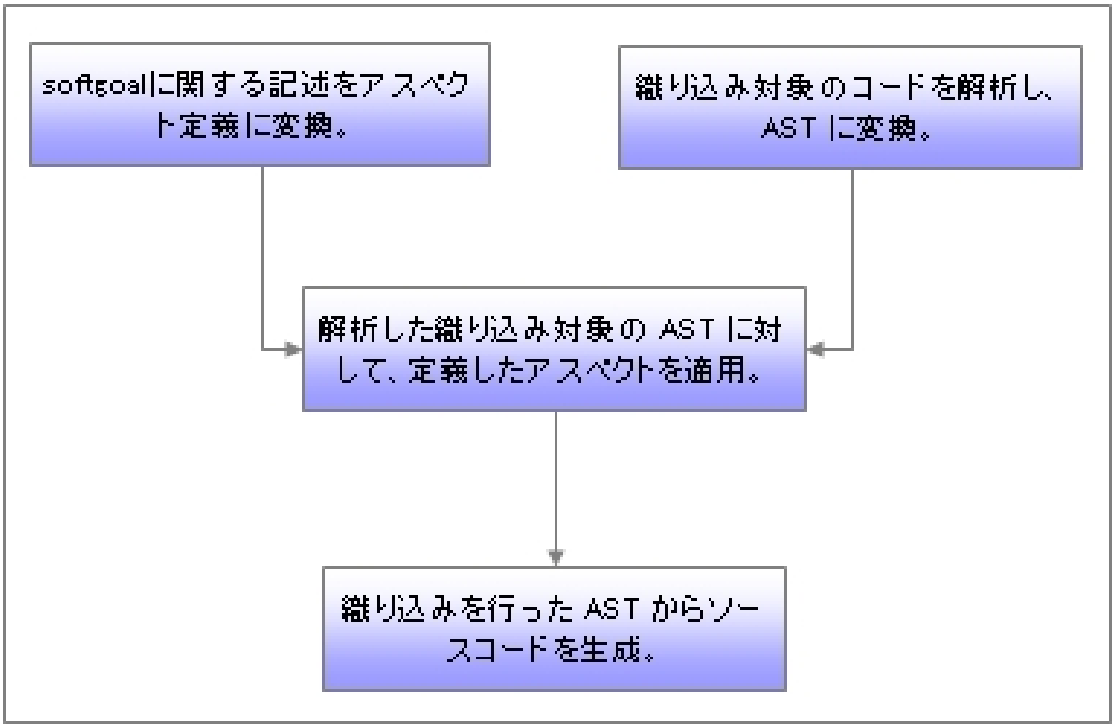

図 2　システムの処理フローチャート

第３章 既存研究

3.1 アスペクト指向プログラミング

3.1.1 アスペクト指向プログラミング（AOP）

アスペクト指向プログラミングとは、ソフトウェアの特定の振る舞いを「アスペクト」として分離し、モジュール化するプログラミング技法。オブジェクト指向プログラミングの問題点を補うために考え出された手法である。

オブジェクト指向プログラミングでは、属性(データ)と操作(メソッド)の集合であるオブジェクトをソフトウェアの分解単位として扱うが、オブジェクトとしてうまく分解ができないソフトウェアの「様相」や「側面」といったものが存在し、このような様相や側面は、複数のオブジェクト間にまたがる操作となる。これを「横断要素」と呼ぶ。

横断要素の代表例としては、プログラムの実行の様子を記録するロギング操作などが挙げられる。横断要素はプログラムコード中に散在するため、すべてを把握し管理することが難しく、また横断要素に対して変更を加える場合にはコード中のあらゆる場所から該当部分を探し出して書き換える必要が生じる。

アスペクト指向プログラミングでは、こうした要素を「アスペクト」としてモジュール化し分離することで、把握・管理・変更を容易にする。アスペクト指向プログラミングを導入することにより、既存のコードの手を加えなくてもプログラム中に散在する特定の機能を持った部分を書き換えることができる。アスペクト指向プログラミング環境は既存のプログラミング言語の拡張機能などの形で提供されているものが多く、Java を拡張してアスペクト指向プログラミングを可能とする「AspectJ」などが有名である。

3.1.2 アスペクト指向プログラミングの問題点

AOP によるセキュリティ対策の実装を通じて感じた問題点がいくつかある。まず、アスペクトのコードによりソフトウェアの主機能や他のアスペクトの動作を阻害する可能性があることが挙げられる。これは、アスペクトが織り込み先のコードをある程度改変できることによる。コーディングミスによるアスペクトの不具合を防ぐために、アスペクトのコーディングについては慎重さを要求されるが、多数のアスペクトを利用する場合にはミスの発生も十分考えられる。セキュリティ関心事を実装する場合には僅かなコーディングミスが重大な欠陥に結びつく可能性もあるため、ミスの早期発見・防止のための方策が必要

である。

さらに、アスペクトが複数存在する際にそれらが開発者の意図しない形で干渉してしまう場合が考えられる。一つのプログラム中に複数のアスペクトが存在する場合、織り込まれるポイントが重複するとアスペクト間で衝突が起こる可能性がある。AspectJ などの AOP の実装では、アスペクトの優先度を設定し実行順序を設定できるが、アスペクトが多数存在する場合などにはコーディングミスなどにより適切に記述されない場合も想定される。アスペクトの実行順序が適切に設定されていなければ、開発者の意図しない順序で実行されてしまう。

また、コーディングミスによりアスペクトが開発者の意図しないポイントに挿入されてしまう場合も考えられる。これにより、アスペクトによるセキュリティ関連のコードが妨害されてしまったり、プログラムが本来果たすべき役割が妨げられてしまったりする可能性がある。

セキュリティのコードは、プログラム中でも特にその正確な実行が求められる。そのためには、開発時のコーディングの正確さの保証とバグの早期発見が必要となる。これを実現するための手法の一つとして、契約による設計が挙げられる。これは、コードに対して契約を記述し、それをコードに守らせることによってコーディング時に正確性を増し、バグの早期発見に役立てるものである。

契約による設計では、アスペクト間の相互干渉についての問題は解決できない。契約による設計は個々のルーチンに関して契約を記述するものであり、アスペクト間の関係に関して記述するものではないからである。

そこで、アスペクト間の相互作用について適切に記述し、それを遵守させる機構が必要であると考えられる。アスペクトは従来のオブジェクト指向のルーチンと異なり、呼び出されるのではなくアスペクト側からポイントを指定して挿入されるため、個々のアスペクトを見ただけではアスペクトの相互関係について理解しにくい。そのため、アスペクト間の関係について記述していくことで開発者の予期しない形でのアスペクトの競合を防ぐとともに、記述によってプログラム中のアスペクトの関係について理解する助けにもなると考えられる。

アスペクトというものが横から入り込んでくるため、OOP のソースコードを見ただけでは実行時の動作がわからない。 そう言った意味でプログラムが複雑化する危険性をはらんでいる。 また、アスペクト同士の干渉が起こることもあり、アスペクトの適応順序によって挙動が変わることもある。

アスペクト指向でウィービングの対象をどう管理するかも疑問が多く、「全ての」の適用範囲はどこまでなのか、あるいは特定の範囲だけに適用したいこともあるだろう。 現行の DI コンテナでの AOP ではメソッド名のマッチングによって特定の名称のメソッドの前後に適用といったことができるがこれは自然とは言い難い。

gattaislime 氏のアスペクト指向の問題点では「要するに、現在の典型的なアスペクト指

向では、アスペクトを織り込むオブジェクトがホワイトボックスであることを要求するか、逆にアスペクト自体がオブ ジェクトの実装を規定するか、ということになってしまい、結果としてアスペクトが単なる暗黙的多重 TemplateMethod に成り下がっているのでは ないかと感じてしまうのだ。このような暗黙の TemplateMethod は非常に厄介で、ソースコードをそのまま追っても処理の流れがつかみにくく、最悪の場合ソースコード全体をメソッド名や属性名で grep して、関連するすべてのアスペクトを洗い出さなければならない。」と指摘する。

3.2 Web アプリケーションにおける AOP

Web アプリケーションには、アクセス制御、ページ遷移制御、パフォーマンス・チューニング等の、Web アプリケーション固有の横断的関心事があり、それらはプログラムの保守性を低下させる要因となる為、モジュール化する事が望ましい。横断的関心事をモジュール化する為の技術として、AOP (Aspect-oriented Programming)がある。

代表的な AOP 言語である AspectJ では、横断的関心事をアスペクト(aspect) と呼ぶモジュールとして定義する。アスペクトは、横断的関心事の挿入位置であるジョインポイント(join point) を選択するポイントカット(pointcut) の定義と、ポイントカットで実行されるアドバイス(advice) から構成される。

しかし、既存の AOP 言語は、ポイントカットで Web 固有のイベントや利用コンテキストを直接取り扱う事ができない為、Web アプリケーションの横断的関心事をモジュールする上で、必ずしも適しているとは言えない。そのため、Web アプリケーションに特化した AOP 機能を導入し、これらの問題の解決に用いる。

3.3 アスペクト指向要求工学

アスペクト指向開発は、オブジェクト指向開発における次のような問題を解決するために提案された。

（１）分散問題

オブジェクトに機能が隠蔽されるため、同様の記述が複数のオブジェクトに分散する

（２）友連れ問題

オブジェクトの中で複数の機能が同時に混在して記述される

このため、複数のオブジェクトに対して横断的に出現する共通的な関心事を抽出して独立に記述しておき、必要に応じて参照するという考え方としてアスペクト指向が提案された。このように共通的な関心事を分離することを SoC（Separation of Concerns）という。

SoC では、できるだけ機能を単純化することが求められる。これによって、機能定義の重複を削減することができる。このような考え方は、ソフトウェア工学の基本原理の一つでもある。つまり機能の定義は１箇所に限定しておき必要な箇所で参照するのである。

そしてなぜアスペクト指向要求工学が必要なのか、この理由は以下の３点に整理できる。

（１）共通関心事を識別する必要がある。

そうしないと、複数個所に同じ要求が分散して重複記述されるために変更時の影響範囲が特定できないだけでなく、理解容易性も低下する。

（２）要求のモジュール化が必要。

共通関心事を独立に記述することで要求のモジュール化が可能になり、理解容易性を向上できる。

（３）影響範囲の明確化と限定が必要。

アスペクトを独立に記述し参照元との関係を明示的に記述することで、アスペクトの影響範囲を限定できる。

クロスカット要求とは、要求 B の構成要素が要求 A を考慮しないと満足できないとき、要求 A が要求 B をクロスカットするという。この場合、要求 A がクロスカット要求になる

クロスカット要求の形態は、オブジェクトの分散問題と友連れ問題と同様に、分散要求と友連れ要求の２つがある。

分散要求とは、同じ要求が異なる複数の要求に散在する。このとき、複数の要求で必要とされる要求がクロスカット要求になる。

友連れ要求とは、ある要求が複数の性質や機能要求を同時に必要とする。このとき同時に必要とされるクロスカット要求が友連れ要求になる。

そしてこの AORE には以下のような特徴がある。

（１）抽象化

共通関心事の詳細を隠蔽して抽象化することで、要求を複数個所に分散することなく一意に記述できる。

（２）モジュール化

共通関心事を分離独立させておくことで、影響範囲を限定できる。

（３）一貫性

共通関心事と他の要求との相互関係を明示的に記述できるようになるのでその一貫性を系統的に保証できる。

AORE は従来の要求工学を補完する技術であって、ゴール指向要求工学やシナリオ指向要

求工学などと組み合わせることができる。ゴール指向手法でも、関心事を扱うことができるが、すべてを同列に扱うため組合せが系統的に支援されていないこと、手段目的分析がアクタごとに局 所的にしかできないことなどの問題がある。そこで、ゴール指向で抽出したアクタとしてのステークホルダのニーズを AORE でモジュール化して組み合わせることができる。

3.4 ゴール指向要求分析

3.4.1 ゴール分析

ゴール分析は開発対象システムの要求が達成すべき目標を分析し定義するための手法である。要求の目標を明確にして顧客と合意することで、設計の妥当 性が機能仕様に対して確認できるように、機能要求の妥当性もゴールに対して確認できるようになる。

もしシステム開発の背景となる問題が明確であれば、トップダウンでシステム開発のゴールを明示的に定義できる。しかし、システムが解決すべき問題が 明確でなければ、問題を解決する前に、問題自体の構造を明らかにする必要がある。問題構造が明確になったら、解決すべき問題を定義し、システムが具備すべ き特性としての非機能要求と機能要求との関係を明らかにする。

したがってゴールの定義には①問題分析②問題定義③非機能要求と機能要求の関係分析という３つの段階があることが分かる。そう考えると、ゴールと非機能要求の観点から機能要求を段階的に抽出する手法を構成できることに気づくだろう。

システムに対する一般的な非機能要求としては、使いやすさ、柔軟性、移行の容易性などが考えられる。ゴール分析ではこれらの定性的で曖昧な要求を「ソフトゴール」と呼ぶ。

そしてソフトゴールには、以下のような４つの関係がある。

（１）AND 関係

ゴール A、B がともに成立するときゴール G が成立するか、ゴール A、B のどちらか一方でも成立しなければゴール G も成立しないとき、ゴール A と B は、ゴール G に関して AND 関係にある。

（２）OR 関係

ゴール A、B のどちらか一方が成立するときゴール G が成立するか、ゴール A、B のすべてが成立しなければゴール G も成立しないとき、ゴール A と B は、ゴール G に関して OR 関係にある。

（３）＋関係

ゴール A が成立するとき､ゴール B が成立するとき､ゴール A は B に対して＋関係にある。

（４）－関係

ゴールＡが成立するとき、ゴールＢが成立しないとき、ゴールＡはＢに対して－関係にある。

もしソフトゴールに対して、十分なだけの肯定的な根拠があれば、ソフトゴールを満足できるという。もしソフトゴールに対して、十分なだけの否定的 な根拠があれば、ソフトゴールを満足できないという。このように、ソフトゴールは、肯定か否定かのどちらか一方にいつも決められるわけではないことを注意しておく。このようなソフトゴールに対しては、関係者に意思決定を仰ぐことになる。重要なことは、ソフトゴールの背景を明確化できるように分析しておくことが必要である。

ゴールを記述する際の詳細化の度合い（粒度）やゴールの抽象度を決めておくことが重要である。たとえば、サブゴールをどこまで詳細化していくかについては良く考える必要がある。サブゴール間の粒度のバランスを調整していくことがポイントだろう。また、ゴールグラフやトポイ図などの表現法がいくつかあるので、課題に合わせて適切な表現法を選択する必要がある。ゴールが非機能要求だけでなく機能要求の修正に伴って変更されることもあるかもしれない。またゴールがトポスとしての問題の背景を記述するものであ れば、ビジネス環境やシステム環境の変化に伴って、ゴールが変更されるのは当然だ。したがってゴールの変更管理に留意する必要がある。またゴールが表現す る内容の抽象度をゴールの変更に際して適切に維持しておくことも重要である。いずれにしても、ここで紹介したように、ゴールと機能要求とが明確に関係付けられていて変更管理ができていれば、環境変化にも容易に対処できる要求仕様を作成できるに違いない。

ゴール分析では、ゴール間の因果関係や AND／OR 分解を記述するわけだが、その関係はあくまでも論理的に記述しただけであり、実際に成立するかど うかについては、システムを運用した上でないと実証できないことに注意する必要がある。もしゴールが、オブジェクト、オブジェクトの属性、属性の状態に よって表現できるとすると、ゴールの状態をシステム運用時に監視できることになる。したがって、ゴール間の依存関係の妥当性をシステムの運用情報に基づい て評価できるわけだ。このようにゴール分析をしっかりやっておくことで、システム導入効果を評価できるだけでなく、新たなシステムや業務課題を定量的な データによって検証できるようになる。

3.4.2 NFR フレームワーク

ソフトゴールには NFR、操作要求、主張（claim）の３種がある。ソフトゴール依存関係、グラフでゴールの依存関係を分析する。NFR を NFR 型として予めパターン化して再利用することで網羅的な分析ができるという特徴がある。

実際のシステム開発では、ゴール分析の初期段階でこのような NFR 型を参考にして具体的なゴール分析を進めていくことができる。
NFR と操作要求の対応付けを次に一例に示す。

オフィスのワークスタイルを変革するための新しいシステムの開発を顧客から依頼されたとしよう。まず、オフィス業務の分析では、社員、組織、居室、 会議室、会議室予約や会議運営などの社内手続き、会議資料などの共有情報などの資源と、これらの資源に対する操作の両方をモデル化しないといけない。コミュニケーション作業の効率化には、情報作業の効率化とそのための手続き作業の効率化が必要だとしよう。情報作業の効率化では、情報共有、情報作 成、情報収集が必要である。情報共有では情報保護と情報活用が必要だ。使いやすさの点では情報活用は＋関係にあるが、情報保護は－関係である。一方、セキュリティ の観点では、情報保護は＋関係だ。情報収集には、ヒトの活用とインターネットの活用が必要だ。手続きの効率化では、スケジューリングとモニタリングが重要である。このようにして、段階的に、ゴールを詳細化していくと、ゴール分析の木構造ができる。
次に、ゴール分析木に操作要求を追加していく。たとえば、情報保護というソフトゴールを達成するためには、IC カードと社員認証という機能要求（操 作要求）が考えられる。このようにして、図４の下部に操作要求を追加することにより、非機能要求に対する操作要求を識別してゴール分析木に追加していくこ とができる。

3.6 類似の関連研究

本研究は契約による設計を支援する表明記述のアスペクト指向的モジュール化方式(契約による設計を支援する表明記述のアスペクト指向的モジュール化方式山田聖・渡部卓雄)、アスペクトを用いた表明の記述(アスペクトを用いた表明の記述石尾隆・神谷年洋・楠本真二・井上克郎）などで紹介される表明（Assert）とアスペクトを橋渡しする、という手法に影響を受けている。セキュリティモジュールを組み込むために IDL からアスペクトコードを自動生成する仕組みを考察する中で、表明で記述しそこからアスペクトコードを生成するというアプローチを参考にした。

第４章 実装

4.1 システムの構成

本システムは GRD コンパイラと AOP モジュールから成る。それぞれ GRD コードと織り込み対象となるソースコード群を入力としてとり、アスペクトコードと AST を中間生成物として出力する。

最終的にそれらを合成して、織り込み済みのソースコードを出力する。

以下に、これらの一連の処理に関するフローチャートを示す。

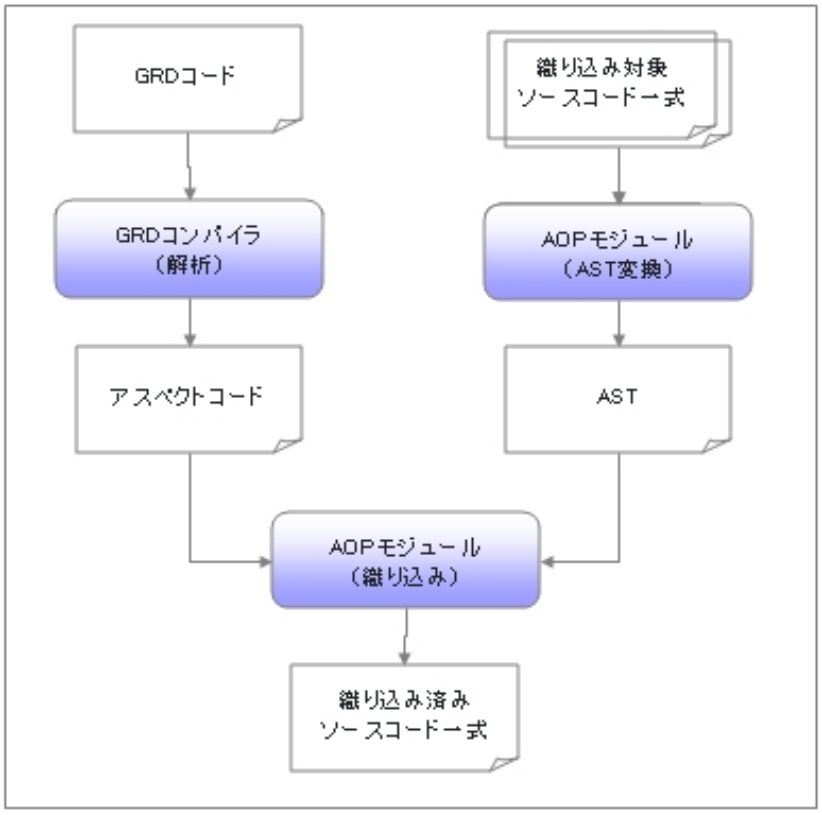

図 3 システムの処理フローチャート

4.2 基盤となる技術

4.2.1 PHPSimpleCC の概要

今回、GRD 言語を開発するにあたって独自のコンパイラコンパイラ“PHPSimpleCC“を開発した。

コンパイラコンパイラとは字句解析器＋構文自由文法解析器を生成するプログラムのことである。別称パーサジェネレータとも呼ばれ、コンパイラを開発するための支援的な役割を果たす。

既存のコンパイラコンパイラに Yacc/lex、Bison/Flex 等があるが、それらは出力コードの可読性が低く出力言語も C 言語であったため PHP で可読性の高いパーサコードを生成する CC を作成する必要があった。

PHPSimpleCC の主な特徴としては、以下の４点である。

（１）文脈を考慮した字句解析が可能

（２）1 ファイルで字句解析、構文解析、解析処理すべてを記述可能

（３）BNF+PHP で記述し、PHP で記述されたパーサを生成する

（４）出力されるパーサコードの可読性が高い

```
// 字句解析＋解析処理
num ::=
      "/¥d+/" :TOKEN <?
 return intval($image);
?>;

// 構文解析＋解析処理
<add> ::=
      num "+" num :CLOSE <?
 list($ln,,$rn) =$context->get_values();
 return $ln + $rn;
?>;
```

図 4 PHPSimpleCC の記述例

4.2.2 AOWP の概要

AOWP は、PHP を対象とした AOP（Aspect-Oriented Programming）フレームワークである。多くの Web アプリケーションには、ユーザ認証、アクセス制御、負荷分散等の、システム全体に影響を与えるような処理が含まれますが、 AOWP では、これらの処理をアスペクトと呼ぶモジュールとして記述する事ができる。アスペクト指向プログラミングは、横断的関心事を分離する為の技術であり、 オブジェクト指向プログラミングを補完する技術である。 横断的関心事とは、システムの広い範囲に記述される処理であり、 例えば、ロギング処理は、ロギング自体はクラスとして モジュール化できますが、どうしてもそのクラスの呼び出しを、 ロギングが必要な全ての箇所に記述する必要がある。 AOP では、このような処理をアスペクトとしてモジュール化する事ができる。

代表的な AOP 言語として AspectJ がある。AspectJ では、ポイントカットとアドバイスを用いて アスペクトの定義を行いるが、AOWP でも、 このポイントカット/アドバイス機構に基づく AOP を採用している。つまりポイントカットでは、"どこに処理を適用するか" を定義し、アドバイスでは、"どのような処理を行うか" を定義する。 例えば、先ほどのロギングの例では、ロギング処理を追加する 特定のメソッド呼び出し等をポイントカットとして定義し、 そのメソッド呼び出しの前にロギング処理を追加する、 といった処理をアドバイスとして定義する。

ポイントカットは、ジョインポイントと呼ばれる、 あらかじめ定義されたアスペクトの適用可能なポイントの集合を、 横断的関心事の挿入位置として選択する。 AOWP では、メソッド呼び出し、関数呼び出し等の、 プログラム実行中のイベントをジョインポイントとして定義している。（AspectJ では、プログラム実行中のイベントに加え、 クラスの構造等もジョインポイントとして定義している。 現在、AOWP では、それらのクラスの構造に関するジョインポイントはサポートされていない。）

ポイントカットは、幾つかのポイントカット記述子を用いて定義する。 AOWP が提供する記述子は、大きく分けて、 特定のイベントと対応した記述子、 特定の条件を満たすときのジョインポイントを選択する記述子、 の 2 種類がある。

アドバイスでは、ポイントカットで選択したジョインポイントに、 どのように横断的関心事を適用するかを定義する。 アドバイスの定義の際には、適用されるジョインポイントに関する情報を 参照する事ができる。 例えば、メソッド呼び出しの場合、呼び出されるオブジェクトや、 メソッド名、メソッドに渡される引き数等を用いる事ができる。

アドバイスは、横断的関心事を実行するタイミングが異なる before、 after、around の 3 つのタイプがある。 before アドバイスは、ポイントカットで選択したジョインポイントの発生前に、 横断的関心事を実行するアドバイスであり、 このアドバイスは、ジョインポイントの発生前に実行させる為、 メソッド呼び出しや、関数呼び出しの場合、事前に引

き数の値をチェックし、それらの内容を修正したりする事ができる。after アドバイスは、ジョインポイントの発生後に実行されるアドバイスであり、 after アドバイスでは、メソッド呼び出し等の場合、返り値を参照する事ができ、 また、返り値に何らかの処理を適用し、新たな返り値を設定する事が可能である。 around アドバイスは、ジョインポイントが表す元のイベントを置き換える アドバイスであり、 around アドバイスでは、任意のタイミングでジョインポイントが表す元の処理を 実行する事ができ、例えば、ポイントカットで選択した処理を複数回繰り返すような、 アドバイスを作成する事ができる。

AOP では、作成したアスペクトは、 織り込みと呼ばれる処理を行う事で、対象のシステムに適用される。 現在、AOWP では、対象の Web アプリケーションのルートフォルダ等を設定ファイルに記述し、 AOWPCompiler.php のスクリプトをコマンドライン上で実行する事で、 織り込みを行う。 AOWP の織り込み処理では、対象の Web アプリケーションを解析し、 ポイントカットが選択するジョインポイントと対応するソースコード上の位置に、 アドバイスを実行する事を追加する、コード変換を行っている。

4.3 システムの機能

4.3.1 システムの有する機能

前述の通り、本システムはアスペクト要求とゴール要求分析の構成要素を対応付けることで、ゴール要求の記述によって表される要求をアスペクトとして実装することができる機能を提供する。

ゴール要求とアスペクトの実装を統合する言語 GRD を用いてソフトゴールに関して記述し、GRD コンパイラはそこからアスペクトコードを生成する。これを AOP モジュールが織り込み対象となるソースコードへ織り込み処理を行うこととなる。言語仕様に関しては、NFR フレームワークの考え方をもとに、ゴール要求に関するグラフを比較的自然言語に近い形で記述することができるように言語設計を行った。

4.3.2 GRD 言語の仕様

ゴール要求をパラグラフとし、その集合により全体は記述される。それぞれ非機能性要求と機能性要求、要求間の関係性を記述する。

機能性要求については、更に実装を記述することで、アスペクト織り込み時にプログラム

の動作に作用を与えるようになる。

また、実装を別途用意する抽象ソフトゴールを記述できる機能を設け、NFR フレームワークの特徴である NFR 型をパターン化して再利用する機能を実現した。

全構文木を記述すると膨大な量となってしまうため、代表的な部分のみを前述の処理記述部分を除き紹介する。

```
// ドキュメントは複数のソフトゴールにより成立
<doc>::=
        ( <goal> ) *
<goal>::=
        <first_goal> ( <soft_goal> ) *
<first_goal>::=
        <aspect> "," "the" "goal"
        ( <extends> ) ? <goal_def> "."
<soft_goal>::=
        ( "abstract" ) ? "soft" "goal" "<ident>" <goal_def> "."

// ソフトゴールの構成要素についての記述
<goal_def>::=
        ( <when_st> ) ?
        ( <rule_st> ) ?
        ( <proc_st> ) ?
        ( <implement_st> ) ?

// 抽象ソフトゴールの実装を行う機能
<extends>::=
        "extends" <aspect>
```

図 5　ドキュメントのパラグラフ定義に関する BNF 記述

```
// ソフトゴールの要素となるruleをAOPの要素aspectへ対応付ける記述
<rule_st>::=
        "should"  "be"  <rule>  (  "and"  <rule>  ) *
<rule>::=
        (  <rule_valid>  |  <rule_implement>  )
<rule_valid>::=
        "valid"  <func>
<rule_implement>::=
        <phpblock>
```

図 6　rule/aspect 要素に関する BNF 記述

```
// ソフトゴールの要素となるwhenをAOPの要素pointcutへ対応付ける記述
<when_st>::=
       <when>  (  "or"  <when>  ) *
<when>::=
        (  "before"  |  "after"  |  "around"  |  "when"  ) <pointcut>
<pointcut>::=
        <pointcut_called>
<pointcut_called>::=
        "call"  <func>
<pointcut_extern>::=
        <func>
<pointcut_implement>::=
        <phpblock>
```

図 7　when/pointcut 要素に関する BNF 記述

```
// ソフトゴールの要素となるprocをAOPの要素adviceへ対応付ける記述
<proc_st>::=
        "unless"  <proc>  (  "and"  <proc>  ) *
<proc>::=
        (  <proc_cancel_operation>  |
        <proc_call>  |
        <proc_implement>  )
<proc_cancel_operation>::=
        "cancel"  "operation"
<proc_call>::=
        <func>
<proc_implement>::=
        <phpblock>
```

図 8　proc/advice 要素に関する BNF 記述

第５章 利用シナリオ

5.1 想定する利用シナリオ

利用ケースとしては WEB アプリケーションにおいて､横断的関心事に類するような要求(アスペクト要求）が生じるケースを想定する。
WEB サービスにおける会員登録の手続きをゴールとし、そこに非機能要求としてセキュリティに関するソフトゴールを置く状況を例として、本手法の利用のシナリオを提案する。そして、そのシナリオのユースケースの中で本システムの機能の評価を行いたいと考える。

5.2 要求の定義

まずはセキュリティに貢献するソフトゴールとして、プライバシーと機密性を置く。更に、プライバシーに貢献する機能要求としてパスワードの強度検査を想定する。サンプルのシナリオでは説明のため網羅的なソフトゴールの提示に関しては省略するものとする。
まずは、NFR フレームワークを用いてセキュリティというソフトゴールに貢献するいくつかのソフトゴールの組み合わせを抽出する。そしてそれを次の図に示すように自然語で列挙する。

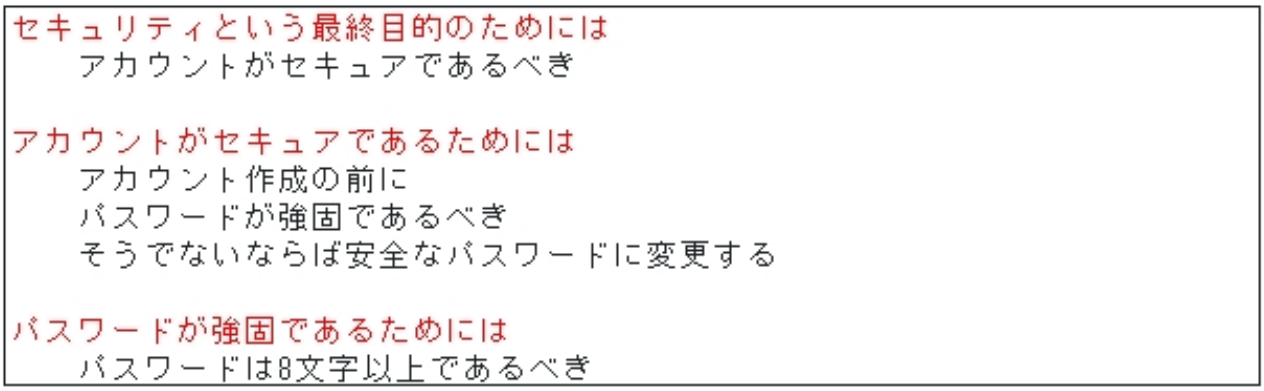

図 9　要求の定義に関する自然語での記述

5.3 GRD 言語による要求記述

先述の自然語で表されたソフトゴールについて、それぞれ GRD 言語での記述を行う。

この時、ソフトゴールに含まれる実装を抽象ソフトゴールと言う形で分割して記述することが可能である。この機能を用いることで、システム横断的に利用可能な NFR グラフとシステム固有の実装を切り分けソフトゴール定義の可用性を向上させる。

次に、先述の自然語で表されたソフトゴールについて、GRD 言語での記述を行ったソースコードを提示する。

```
// セキュリティという最終目的のためには
//    アカウントがセキュアであるべき
Security,the goal
    should be valid accountSecurity.


//アカウントがセキュアであるためには
//    アカウント作成の前に
//    パスワードが強固であるべき
//    そうでないならば安全なパスワードに変更する
soft goal accountSecurity
    before call addUser
    should be valid passwordStrength
    unless <? $this->jp->setArgument("ASafetyPassword", 1);  ?>.

//パスワードが強固であるための条件は後述
abstract soft goal passwordStrength.
```

図 10　要求の定義に関する GRD 言語での記述

```
//セキュリティという最終目的について詳しく言及
ASecurity,the goal extends Security.

//パスワードが強固であるためには
//    パスワードの文字数が8文字以上であるべき
soft goal passwordStrength
    should be <?
        $check =$check && strlen($this->jp->getArgument(1)) >= 8;
?>.
```

図 11　抽象ソフトゴールの実装に関する GRD 言語での記述

5.4 GRD 言語によるアスペクトの生成

続いて、先述の GRD 言語によるソフトゴールの記述を本システムの GRD コンパイラにてコンパイルする。これにより記述したソフトゴールにおける機能性要求、非機能性要求、実装を盛り込んだアスペクトコードとして出力される。

以下に、実際に生成されるアスペクトコードを抜粋して提示する。このアスペクトコードは AOWP によって織り込み可能な形態で記述されている。

```
class SecurityAspect extends AOWP_PerJoinPointAspect {

    public function __construct () {
        $advice = new AOWP_BeforeAdvice();
        $pointcut = new AOWP_FunctionCallPointcut('addUser');
        $advice->setAdviceBody('advicebody');
        $advice->setPointcut($pointcut);
        $this->addAdvice($advice);
    }

    public function advicebody (AOWP_JoinPoint $jp) {
        if ( ! $this->passwordStrength()) {
            $jp->setArgument("ASafetyPassword", 1);
        }
    }
```

図 12　GRD 言語によって生成されたアスペクトコード

5.5 アスペクトの織り込み

続いて、先程生成したアスペクトコードを AOWP を用いて既存システムのソースコード群への織り込みを実施する。以下に示す通り、アスペクトコードが織り込まれた状態で PHP ネイティブのソースコードが作成される。

```
<?php

addUser("tommy","pass");

function addUser($userName, $password) {
 global $users,$passwords;
 $users[] =$userName;
 $passwords[$userName] =$password;
}

?>

ユーザを追加しました。<br>
```

図 13　織り込み対象のソースコード

```
<?php if(!function_exists('function12581506340')) {
function function12581506340($parameter12581506341, $parameter12581506342)
{
 require_once(dirname(__FILE__) . DIRECTORY_SEPARATOR .
'aowp¥sources¥libs¥autoloadplus¥bin¥include.php');
$aspect12581506343 =
AOWP_AspectInstanceManager::getInstance('ASecurityAspect');
$joinPoint12581506344 =
unserialize('O:26:"AOWP_FunctionCallJoinPoint":11:{s:41:"
AOWP_FunctionCallJoinPoint _functionName";s:7:"addUser";s:48:"
AOWP_FunctionCallJoinPoint _staticArgumentCount";i:2;s:43:"
AOWP_JoinPointWithArguments _argumentArray";a:0:{}s:49:"
AOWP_JoinPointWithArguments _staticArgumentCount";N;s:21:" AOWP_JoinPoint
_line";i:3;s:25:" AOWP_JoinPoint _fileName";s:9:"index.php";s:29:"
AOWP_JoinPoint _fileFullPath";s:133:"C:¥root
```

図 14　織り込み後のソースコード

第６章 考察

6.1 妥当性の検証

先述の利用シナリオにおいてはさほど複雑性の高くない要求を扱った。

言語仕様としては、機能的には WEB アプリケーションに特化する形で AOWP が提供する機能を網羅するような拡張を行った。そのため多くの AOP 機能をサポートしている。

しかしながら本手法においては一般的な AOP における問題であるアスペクト間の相互関係による問題は解決されているとは言えない。そのため、単一のジョインポイントに対して多くのポイントカットが集中するとアスペクト間の相互作用により、期待した効果が得られなかったり、予期しない動作を引き起こすことが考えられる。例えば入力チェックに関するアドバイスで、条件エラー時の処理に処理の停止があった場合、その後に入力データの書き換え処理があった場合には、その処理は実行されない。この実行順序が逆転した場合とは異なった動作をすることとなる。しかしながら、この織り込み順序を制御する方法については一般的に AOP モジュールでは制御する事が難しい。

今回開発したシステムでは、ソフトゴール記述（ゴール要求）をアスペクト要求へと変換する機能に重点をおいた。そのため、AOP の織り込み処理に関しては既存研究の成果である AOWP の機能をそのまま利用できる構造となっている。このことにより、AOP 機能に関する問題点については本システムの内部で対処することが難しくなってしまっている。

さらに、AOWP の特徴としてウェブアプリケーションに特化したポイントカットを提供しているという点（Cookie の書き換えや Request の取り出しなどが取得可能な点など）ではメリットを生かすことが出来ているが、反対に既存のコードを直接書き換える、という特徴はマイナスに働いてしまっている。既存コードを書き換える際に著しく可読性を落としてしまっているという点が実用上では非常に大きな問題となることは容易に想像できる。

6.2 性能の検証

GRD 言語の解析処理に関しては実装を PHP インタプリタ上で実行している都合上、PHP インタプリタの性能に依存している。アルゴリズムとしては、字句解析処理を PHP 組み込みの PCRE 正規表現で行っているため比較的高速な処理が期待できる。一方構文解析については構文の木構造を作成し、さらに再帰的に参照を辿りながらアスペクトコードの組み立ても行うため、字句解析に比較すると処理の絶対量の大きい割合を占めることとなる。オン

メモリですべての処理を実行する前提で設計しているため、ソフトゴールの記述量に従ってメモリの使用が一次関数的に増加してしまう点も問題としては上げられる。

アスペクトの織り込みにかかる処理量は、全ソースコードの解析と内部形式への変換なども含めて膨大となる。これはアスペクトコードの記述量、織り込み対象のシステムの規模に比例して大きくなる。第５章でとりあげた利用シナリオの中で示したコードに関する処理だけでも、3000ms 程度の処理時間を要する。ただ、この織り込みに関する部分については完全に AOWP のモジュールに依存しており改善が難しいと考えている。

6.3 考察と今後の課題

機能面では WEB アプリケーションに特化する形で advice、pointcut を定義できるよう言語仕様レベルで対応した。この点については、最終的には AOWP が提供する機能を可能な限り網羅できるように拡張を行ったため多くの AOP 機能をサポートしている。これまで WEB アプリケーションを行ってきた経験上、必要とされる機能は充足していると考える。

しかしながら先述の通り、アスペクトの織り込みがソースコード中にコードを直接混入させる形での実現となっている。これにより実用性が著しく損なわれていると考える。

これは AOWP の性質であるが、保守性の観点からインタプリタが起動するタイミングで織り込みが行われる実装の方が望ましいと考える。インタプリタ型の言語を対象としているため、コンパイル後のオブジェクトコードに手を加えるなどの手法が取れないことも一因である。AspectJ のような性能を実現することを視野に入れた場合、インタプリタの機能として織り込みを実現する必要があると考える。これが可能になればより可用性は増すことになるだろう。

第７章 結論

ゴール指向とアスペクト指向を統合する方法として、GRD 言語を開発した。PHP により開発された WEB アプリケーションを対象として、ゴール要求により表された機能性要求、非機能性要求、実装を組み合わせてアスペクトコードを生成し、織り込みを行うことができるような機能を実現した。これにより設計レベルの概念であるアスペクトを、ビジネスに関するゴールの考え方と統合することを可能にした。

今回、実際にシステムの実現を行い利用のシナリオを想定した上でシステムの検証を行ったが、機能面では不足ないものとなっていると考えられる。しかしながら実用性という観点からシステムを考察すると、織り込みの処理性能、結果的に出力されるコードを鑑みて満足行くものであるとは言い難い。

最終的な AOP 機能に関しては、AOWP に依存する形となりその機能を踏襲することで機能的には不足のないものとなったが、性能的な限界も同時に踏襲してしまう結果となった。この点に関しては、PHP インタプリタのアクセラレータのような形でプログラムのコンパイル時に織り込みが実行されるように実現できれば、更なる可用性が望めたのではないかと考える。

参考文献

[1] An Overview of AspectJ, In Proceedings of the15th European Conference on Object OrientedProgramming Gregor Kiczales, Erik Hilsdale, J.H. M. K. J. P. and Griswold.,W. G. ECOOP 2001,pp. 327.35

[2] アスペクトを用いた表明の記述石尾隆・神谷年洋・楠本真二・井上克郎情報処理学会研究報告(2004-SE-144), Vol. 2004, No.30, pp.75-82, 2004.3.18

[3] 契約による設計を支援する表明記述のアスペクト指向的モジュール化方式山田聖・渡部卓雄情報処理学会論文誌，2005.04.15

[4] 契約による設計を支援するアスペクト指向的振舞インターフェース記述言語 Moxa 山田聖・渡部卓雄情
報処理学会論文誌，2005

[5] AOWP: Web アプリケーション開発向け AOP 機構外村慶二・鵜林尚靖・中島震情報処理学会ＳＥ
シンポジウムソフトウェアエンジニアリング最前線,2008

[6] Aspect-Oriented Programming beyond DependencyInjection, In Proceedings of the 19th European
Conference on Object Oriented ProgrammingChiba, S. and Ishikawa, R. ECOOP 2005, pp. 121.143

[7] An Easy-toUse Toolkit for Efficient Java BytecodeTranslators, In Proceedings of 2nd InternationalConference on Generative Programmingand Component Engineering Chiba, S. and Nishizawa, M. GPCE '03, pp. 364.376 (2003).

[8]W. T. Tsai, Z. Jin, P. Wang and B. Wu, Requirement Engineering in Service-Oriented System Engineering, Proceedings of the IEEE International Conference on e-Business Engineering, pp.661-668, 2007

[9] Lawrence Chung, Brian Nixon,Eric Yu, John Mylopoulos, Non-Functional Requirements In Software Engineering, Kluwer Academic Publishers, 2000.

[10]Baniassad,E.,etal,DiscoveringEarlyAspects,IEEE oftware,Jan./Feb.,pp.61-70,2006.

[11]Safoora Shakil Khan, Muhammad Jaffar-ur- Rehman, A Survey on Early Separation of Concerns, Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05), Pages: 776 - 782 , 2005

[12] Yijun Yu, Julio Cesar Sampaio do Prado Leite, John Mylopoulos, From Goals to Aspects:Discovering Aspects from Require ments Goal Models, Proceedings of the 12th IEEE International Requirements Enginee ring Conference, 2004

参考サイト

[1] Awais Rashid, Aspect-Oriented Requirements Engineering,www.resg.org.uk/EAslides/Awais RashidSlides.pdf

[2] Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, http://www.early-aspects.net/events/oopsla04ws

[3]アスペクト指向の概念
http://blogs.wankuma.com/nagise/archive/2008/03/25/129472.aspx

[4]Lars Rosenhainer, Identifying Crosscutting Concerns in Requirements Specifications, trese.cs.utwente.nl/workshops/oopsla-early-aspects-2004/Papers/Rosenhainer.pdf

[5]第 50 回 : ゴール指向とアスペクト指向要求工学(要求工学 : Requirements Engineering)
http://www.bcm.co.jp/site/youkyu/youkyu50.html

[6]Nan Niuy, Steve Easterbrooky, and Yijun Yuz, A Taxonomy of Asymmetric Requirements Aspects, www.cs.toronto.edu/~sme/papers/2007/Niu-EA07.pdf

謝辞

村岡洋一教授には、研究内容の決定から進め方まで、自分の環境や得意分野を尊重頂き、自由な研究を進められたことを深く感謝致します。また構想にあたり協力頂いた、有限会社シェアリングシードのシステム開発部の皆様に感謝致します。