様式 C -19

科学研究費補助金研究成果報告書

平成 21 年　6 月　2 日現在

研究種目：若手研究（A）
研究期間：2006～2008
課題番号：18680004
研究課題名（和文）　アスペクト指向言語の基礎モデルとその応用
研究課題名（英文）　Foundational models of aspect-oriented languages and their applications
研究代表者
氏　名　増原　英彦　（MASUHARA HIDEHIKO）
東京大学・大学院総合文化研究科・准教授
研究者番号：40280937

研究成果の概要：横断的関心事をモジュール化する技術として知られているアスペクト指向プログラミング言語に関して、言語モデルの深化および言語機構の拡張と実現を行った。実際の成果はいくつかのテーマに分かれるが、代表的なものとしては解析に基づくポイントカットを利用した動的なポイントカットの最適化手法の提案と、テストに基づくポイントカットの提案と実証が挙げられる。

交付額

（金額単位：円）

	直接経費	間接経費	合　計
2006 年度	1,700,000	510,000	2,210,000
2007 年度	2,200,000	660,000	2,860,000
2008 年度	1,800,000	540,000	2,340,000
年度			
年度			
総　計	5,700,000	1,710,000	7,410,000

研究分野：総合領域
科研費の分科・細目：情報学・ソフトウェア
キーワード：モジュール化、アスペクト指向、AspectJ、プログラミング言語モデル、表現力

１．研究開始当初の背景

情報システムが社会基盤の大きな部分を担うようになった今日では、それを構成するソフトウェアの規模は大きくなり続ける一方で、ビジネスルールの変化、企業やサービスの統合や、新規事業の創出などの様々な社会的要因から、短期間でのシステム開発や仕様変更を行わなければならない状況も増加の一途を辿っている。

そのようなソフトウェアシステムを効率的かつ確実に開発し保守してゆくため、ソフトウェアを小さな単位に分割して開発・保守する「モジュール化」が以前にも増して重要となってきている。モジュール化のための技術としては、オブジェクト指向プログラミングやオブジェクト指向設計が知られている。

オブジェクト指向は、大規模なソフトウェアを重要な役割を果たしており、今日のソフトウェア開発では欠かすことができないものとなっているが、一方でその限界もよく知られるようになってきている。特に、最適化やセキリティ対策のような関心事は、本質的にソフトウェア中の多数のモジュール(クラスや関数など)に散在しがちであり、オブジェクト指向のような従来の手法をもってもソフトウェア開発・保守の効率を低下させる要因になっていることが明確に認識されるようになってきている。

このような状況の中で、本質的にソフトウェア中の多数のモジュールに散在してしまう関心事(これを横断的関心事と呼ぶ)をモジュール化するための新たなプログラミング手法としてアスペクト指向プログラミング(AOP)が提唱され(Kiczales, et al. *ECOOP'97*)、研究・産業界の双方から大きな注目を集めるようになった。特に Java 言語と互換性のある AOP 言語 AspectJ の作成(Kiczales, et al. *ECOOP'01*)後は、企業における製品開発にも採用される(Colyer and Clement, *AOSD'04*)ようにもなっている。

このように実用性が注目される一方で、AOP 言語は充分な理論化をされずに言語が設計され、それを追うように理論化が進められてきている。本研究申請時点では、AOP 言語の理論・設計に関して次のような状況であった。

- AOP 言語の実行モデルに関しては、これまでに AspectJ の実行モデルを簡潔化したものに関する研究(Wand, Kiczales and Dutchyn, *FOAL'02* など)がほとんどであった。
- 関数型 AOP 言語に関してはいくつかの提案がある(Walker, Zdancewic and Ligatti, *ICFP'03* など)が、どれも非常に簡単化された AOP 機能のみを扱っている。また、研究代表者は実用的な AOP 機能を持った関数型言語を提案し(Masuhara, Tatsuzawa and Yonezawa, *ICFP'05*)、簡単化されたモデルでは無視されていた様々な性質を明らかにしている。
- 型システム・モジュールシステムに関しては、MiniMAO1(Clifton and Leavens, *FOAL'05*)などが知られている。また、最近になって Crosscutting Programming Interface (Griswold et al., *IEEE Software* 23(1),2006)のように、AOP における新しいモジュール化の方法が考案されている。本研究では、(2)を土台としてこれらの提案を具体的な言語処理系として検討を進め、理論面や言語設計面の深化をつなげる位置付けとなる。
- 言語処理系の設計に関しては、従来の研究はプログラム変換を前提としたコンパイラが中心であった(Hilsdale and Hugunin, *AOSD'04*, Avgustinov et al., *AOSD'05*)。そのため、AOP 言語の新しい実行モデルを考えた場合にそのプログラムをコンパイルする方法は自明ではない。

２．研究の目的

研究の全体構想は、アスペクト指向プログラミング(AOP)に対して新しい実行モデルと実現方式を与え、それに基づいた次世代の AOP 言語を作成することを目的としている。

特に、データ型に関するある種の誤りをコンパイル時に検出できない、言語が一貫性無く拡張されている等の問題について、本研究は、AOP に新たに理論的裏付けのある実行モデルを与え、次世代の AOP 言語の基礎を構築することを大きな目的とする。具体的には以下の 4 点を行う。

(1) 継続(continuation)をとり入れた AOP の新しい実行モデルの提案。
(2) その実行モデルに基づいた関数型 AOP 言語の設計および処理系の試作。
(3) 実用的 AOP 言語に対する型システムとモジュールシステムを設計の検討。
(4) 末尾呼出除去性を保存するコンパイル方式を提案および処理系の試作。

まず(1)に関しては、再利用性や表現力の高い AOP 言語の実行モデルを、従来の AOP 言語にとらわれずに設計する。その際、プログラミング言語の様々な機能を簡潔に説明できる継続を実行モデルの基本単位とすることで、再利用性や高い表現力を実現するアプローチをとる。提案したモデルは、インタプリタ処理系を作成することで実現可能性やプログラムの表現力を検討する。

(2)に関しては、上で提案した実行モデルを関数型言語に適用した AOP 言語を設計し、信頼性の高い関数型言語の理論と AOP 言語の実用的な機能の融合を図る。設計は、関数型言語 Objective Caml に、AspectJ の AOP 機能を導入して行う。処理系は、Objective Caml 言語のコンパイラを改造したソースプログラム変換系を試作し、プログラムの記述力などを検討する。

(3)に関しては、上で設計・試作した AOP 言語を土台として、型安全性を保証する型システムと、分割コンパイルを可能にするモジュールシステムを設計する。基本方針は、単純化された AOP 言語に対する型システム・モジュールシステムを上で設計した AOP 言語へ応用するものであるが、理論的な研究では明らかでなかった実用的な AOP 言語機能との関係に注意して検討を行う点が本研究の特色である。

(4)に関しては、(1)で提案した実行モデルのコンパイル方式を考え、(2)で設計した言語のコンパイラ開発につなげる。従来の AOP 言語は命令型言語をプログラム変換することを前提としていたのに対し、本研究ではより簡素化された実行モデルを対象とする点と、末尾呼出の除去(他の関数を呼び出して終了する関数を効率的に実現する技法)が基本である関数型語を前提としてより一般化した枠組になっている点が特徴である。

３．研究の方法

全体的な方針は、研究目的で述べた(1)〜(4)の項目を順に進めてゆくこととしたが、研究の進捗に対応して最終的には以下のような方法をとった。

研究目的の(1)については、AspectSandBox(Wand, Kiczales and Dutchyn, *FOAL'02*)で提案された AspectJ 流の意味論を出発点として実行モデルを作成した。具体的には、直接形式で記述されている AspectSandBox の実行モデルを継続渡し形式に改め、意味論における継続への適用時点を結合点として選び、アスペクトの適用の意味記述を与えた。さらに、作成した意味論の妥当性を検討するために、ML 言語でインタプリタ処理系を作り、AspectJ で書かれた既存のプログラムと等価なプログラムを作成して動作を確認し、表現力の違いを考察した。

研究目的の(2)については当初の予定を変更し、関数型言語 Haskell を基にした AOP 言語の設計を検討した。具体的には、Aspectual Caml の研究(*ICFP'05*)で行った方法論を踏襲し、AspectJ 流の AOP 機構を Haskell に導入した場合の言語設計の考察を行った。

研究目的の(3)に関しては当初の予定を変更し、次の 3 点について研究を進めた。

- (3a) AOP 言語のポイントカットに関する誤りを発見するための型システムの検討を行った。具体的には、AspectJ を簡単化した AOP 言語において適合する結合点が存在しないようなポイントカットを発見するような型システムを考案し、その型システムの正しさを示す。
- (3b) AspectJ 言語のアドバイス機構の型システムに関する強い制限を実際的なプログラムから発見し、それを緩和するような型システムを提案した。さらに、AspectJ 言語処理系を改造することで制限を弱めた言語処理系が実現可能であることを示した。
- (3c) 新しいモジュール化機構を検討する準備として、AOP による「モジュール化」の本質を再検討した。具体的には、モジュール化機構が達成すべき性質の再定義を Parnas(*CACM* 15(12),1972)や Baldwin-Clark(*Management Science* 52(7),2006)などを基に行い、横断的関心事の定義を検討した。

また、研究目的で挙げたテーマ以外の研究として、次の研究を行った。

(5) 解析に基づくポイントカットを利用した、動的なポイントカットを最適化する手法の提案。

(6) ポイントカットの精度と頑健性を同時に向上させる「テストに基づくポイントカット」の提案と、その実証。

４．研究成果

研究方法の節で述べた各テーマについての成果を述べる。

(1) 新しい AOP 実行モデルとしては、新しいポイントカット・アドバイス機構として結合点が時間軸上の点であるような Point-In-Time モデルを提案し、その意味論を継続(continuation)概念を用いて記述した。また、このポイントカット・アドバイス機構では、これまでの AOP 言語では複数のアドバイスによって定義されなければならなかったような横断的関心事を、1 つのアドバイスによって定義できることを示した。この成果は査読付国際会議論文[7]として採録・発表された。

(2) 関数型 AOP 言語に関しては、Haskell 言語を基にしたポイントカット・アドバイス機構の検討を行い、特に副作用の扱いの面から機構の設計を進めた。Haskell のような純関数型言語では、副作用を扱うためにはモナドという特別な記法を用いる必要があるため、そのような副作用をアスペクトに隠蔽するための方法を中心に検討を行った。今後、言語処理系の試作と、実際の Haskell プログラムに対する実験を通して言語設計の精緻化を進める予定である。

(3a) AOP 言語のポイントカットに関する誤りの発見手法として、型システムと制約解消系を用いた手法を提案した。結合点は引数やメソッド名などの複数の属性を持つデータであることに注目し、結合点をレコードとして表わし、ポイントカットからレコードに対する制約を生成する。そして、制約が矛盾している場合には適合する結合点が存在しないものとして指摘するような型システムを作成し、型システムの正当性の証明を行った。この成果は、口頭発表、査読付国際ワークショップ論文[5]、論文誌論文[1]として発表されている。

(3b) アドバイス機構の型システムの緩和に関しては、まず現在の AspectJ 言語では、異なる型の返値を与えるような around アドバイスが適用できない問題を、具体的なアスペクト定義から発見した。さらにその原因が、AspectJ 言語が、アドバイス適用の可否を適用対象となる結合点の型に基づいて判断している点にあることを明らかにした。この問題に対し、結合点の型ではなく、結合点の返値が使用される場合の型によって適用の可否を判断する手法を提案し、その規則の明確化と処理系の作成を行った。この成果は、問題点の指摘が国際会議におけるポスター発表[16]として、解決策のアイデアが査読付国際ワークショップ論文[3]として、また規則の明確化と処理系については査読付ワークショ

ップ論文[2]として発表されており、今後これらをまとめた論文を投稿する予定である。

(3c) AOP による「モジュール化」の再定義として、モジュール化機構が達成すべき性質の再定義を Parnas(*CACM* 15(12),1972) や Baldwin-Clark(*Management Science* 52(7),2006) などを基に行い、横断的関心事の定義を検討した。特に Parnas によるモジュール化の定義に従って、モジュール化を「困難か変化しがちな設計を隠すもの」とした場合に、横断的関心事は「境界が困難か変化しがちな関心事」と定義し、この定義に基づいた AOP の性質を他の定義と比較した。現時点では予備的な段階であるが、この成果は国際ワークショップの基調講演[14]において公表している。

(5) 動的なポイントカットの最適化手法として、利用者が解析に基づくポイントカットを定義できる AOP 言語のコンパイラ SCoPE の提案と、その応用として動的ポイントカットの最適化を行った。SCoPE では、利用者が定義した条件ポイントカットを解析し、全ての自由変数がコンパイルに決定可能な場合には評価を行ってしまうというものである。さらに、SCoPE を用いて保守的な制御流解析や情報流解析に基づくポイントカットを定義できることも示した。これは動的に決定される解析結果を、コンパイル時に絞り込んでおくという考え方に基づいている。この成果は、国際会議論文[6]、論文誌論文[11]、国際ワークショップ論文 [10]、国際会議ポスター発表[13,15]として公表されている。

(6) 新しい種類のポイントカットとして「テストに基づくポイントカット」を提案し、処理系を作成してその有効性を実証した。テストに基づくポイントカットは、単体テストプログラムを用いて間接的に対象プログラムの結合点を指定する。テストプログラムの実行履歴を用いることと、間接的な指定によってポイントカットの精度と頑健性を同時に向上させることができる。この成果は、論文誌論文[3]および査読付国際会議論文[4]として発表されている。

また、上記以外の成果として、「コンピュータソフトウェア」誌に投稿した AOP に関するチュートリアル記事[12]が採録され、日本ソフトウェア科学会解説論文賞を受賞した。

５．主な発表論文等

〔雑誌論文〕（計 12 件）

[1] 青谷知幸, 増原英彦. アドバイスの安全な実行のためのアスペクト指向プログラミング言語の型システム, コンピュータソフトウェア, 査読有, 26(2), 2009, pp.170-182.

[2] 当山学, 増原英彦, 異なる型の値を返すアドバイスを許すアスペクト指向言語の織込機構, 第 11 回プログラミングおよびプログラミング言語ワークショップ(PPL2009)予稿集, 査読有, 2009, 185-199.

[3] Hidehiko Masuhara, On Type Restriction of Around Advice and Aspect Interference, *Proceedings of the 3rd International Workshop on Aspects, Dependencies and Interactions (ADI'08)*, 査読有, 2008. (online proceedings)

[4] Kouhei Sakurai and Hidehiko Masuhara, Test-Based Pointcuts for Robust and Fine-Grained Join Point Specification, *Proceedings of the 7th International Conference on Aspect-Oriented Software Development (AOSD'08)*, 査読有, 2008, pp.96-107.

[5] Tomoyuki Aotani and Hidehiko Masuhara, Towards A Type System for Rejecting Never-Matching Pointcut Compositions, *Foundations of Aspect-Oriented Languages (FOAL2007)*, 査読有, 2007, pp.23-26.

[6] Tomoyuki Aotani and Hidehiko Masuhara, SCoPE: an AspectJ Compiler for Supporting User-Defined Analysis-Based Pointcuts, *Proceedings of the 6th International Conference on Aspect-Oriented Software Development (AOSD'07)*, 査読有, 2007, pp.161-172.

[7] Hidehiko Masuhara, Yusuke Endoh and Akinori Yonezawa, A Fine-Grained Join Point Model for More Reusable Aspects, *Proceedings of the Fourth ASIAN Symposium on Programming Languages and Systems (APLAS 2006)*, 査読有, *Lecture Notes in Computer Science* 4279, 2006, pp.131-147.

[8] 櫻井 孝平, 増原英彦, アスペクト指向プログラミングにおけるテストに基づいたポイントカットの提案, コンピュータソフトウェア, 査読有, 24(3), 2007, pp.141-152.

[9] 大根田 裕一, 増原英彦, 米澤 明憲, 値間依存性に基づくポイントカット記述のためのバイトコード変換, コンピュータソフトウェア, 査読有, 24(2), 2007, pp.27-40.

[10] Hidehiko Masuhara and Tomoyuki Aotani, Issues on Observing Aspect Effects from Expressive Pointcuts, *Proceedings of Workshop on Aspects, Dependencies and Interactions (ADI'06)*, 査読有, 2006, pp.53-61.

[11] 青谷知幸, 増原英彦, ユーザー定義され

たプログラム解析を利用するアスペクト指向プログラムのコンパイル手法, コンピュータソフトウェア, 査読有, 23(2), 2006, pp.157-167.

[12] 増原英彦, チュートリアル: アスペクト指向プログラミング, コンピュータソフトウェア, 査読無, 23(2), 2006, pp.4-28.

〔学会発表〕（計 6 件）

[1] Tomoyuki Aotani and Hidehiko Masuhara, Optimizing Dynamic Pointcuts by using SCoPE, Poster presentation at AOSD.09, March 2009, University of Virginia.

[2] Hidehiko Masuhara, Towards Right Abstraction Mechanisms for Crosscutting Concerns, A keynote talk at 5th ECOOP Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE'08), 07 July 2008, Paphos, Cyprus.

[3] Tomoyuki Aotani and Hidehiko Masuhara, SCoPE: an AspectJ compiler for developing intuitive and robust aspects by using program analysis, Poster presentation at AOSD.08, April 2008, Brussels, Belgium.

[4] Hidehiko Masuhara, Relaxing Type Restrictions of Around Advice in Aspect-Oriented Programming, Poster presentation at APLAS 2007, November 2007, National University of Singapore.

[5] Tomoyuki Aotani and Hidehiko Masuhara, アドバイスの安全な実行のためのアスペクト指向プログラミング言語の型システム, 日本ソフトウェア科学会全国大会, 14 September 2007, 奈良先端科学技術大学院大学.

[6] Jan Hannemann and Hidehiko Masuhara, Aspect Mining using Structural Program Properties, The Second DIKU-IST Joint Workshop on Foundations of Software, 22 April 2006, 神奈川.

〔その他〕

６．研究組織

(1)研究代表者

増原　英彦　（MASUHARA　HIDEHIKO）
東京大学・大学院総合文化研究科・准教授
研究者番号：40280937

(2)研究分担者

(3)連携研究者