論文 / 著書情報
Article / Book Information

題目(和文)	ソフトウェアプロダクトラインにおけるフィーチャの相互作用の実装に関する研究
Title(English)	A Study on Implementations of Feature Interaction in Software Product Lines
著者(和文)	武山文信
Author(English)	Fuminobu Takeyama
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第9359号, 授与年月日:2013年12月31日, 学位の種別:課程博士, 審査員:渡辺　治,千葉　滋,増原　英彦,首藤　一幸,脇田　建
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第9359号, Conferred date:2013/12/31, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Category(English)	Doctoral Thesis
種別(和文)	論文要旨
Type(English)	Summary

（博士課程）
Doctoral Program

論 文 要 旨

THESIS SUMMARY

専攻： Department of	数理・計算科学専攻　専攻	申請学位（専攻分野）： Academic Degree Requested	博士 Doctor of　（理学）
学籍番号： Student ID Number		指導教員（主）： Academic Advisor(main)	渡辺　治
学生氏名： Student's Name	武山　文信	指導教員（副）： Academic Advisor(sub)	千葉　滋

要旨（和文 2000 字程度）
Thesis Summary（approx.2000 Japanese Characters ）

ソフトウェアプロダクトラインは共通のソフトウェアアーティファクトから作り出される、類似し、かつ多様なソフトウェアからなる集合である。このようなプロダクトラインを開発する方法に、ソフトウェアをフィーチャとよばれるものに分解し、プロダクトラインをフィーチャの集合と考える方法がある。フィーチャとは機能や動作環境といったソフトウェアが持つエンドユーザ視点の特徴である。開発者はこのフィーチャの集合からあるプロダクトに用いるフィーチャを選ぶことで、生成されるプロダクトをカスタマイズすることができる。フィーチャ指向プログラミングはこのようなフィーチャの実装のためのプログラミングパラダイムであり、あるフィーチャに関するコードを、独立したフィーチャモジュールに分離して実装できるようにする。この分離には破壊的クラス拡張と呼ばれる仕組みが用いられる。破壊的クラス拡張は既存のクラスの定義を、そのクラスの外側から拡張できる仕組みである。これにより、開発者はソースコードを変更せず、フィーチャモジュールを組み合わるだけでソフトウェアプロダクトを生成することができる。アスペクト指向プログラミングもまた、アスペクトと呼ばれる破壊的クラス拡張を提供するため、フィーチャモジュールの実装に用いることができる。

しかしながら、フィーチャが相互作用するとき、単純なフィーチャモジュールの組合せでは問題のある動作を引き起こすことがある。フィーチャの相互作用は、ある組合せのフィーチャの間に必要不可欠な動作であり、プログラマはこのような組合せが正しく動作するように注意深く相互作用を実装しなければならない。相互作用を実装するモジュラな方法として、相互作用をデリバティブに記述する方法が提案されているが、デリバティブの数が保守できないほど大きくなる可能性がある。

アスペクト指向プログラミングにおいても、複数のアドバイスが同じメソッドを拡張するときに発生する衝突という相互作用の問題が存在する。フィーチャの組合せを同じプロダクトに選択した結果、それぞれのフィーチャを実装するアスペクトのアドバイスが衝突するならば、プログラマはどのように衝突したアドバイスが合成され、実行されるかを明示的に指定しなければならない。従来の言語では、衝突するアスペクトに優先順序を与えアドバイスを直線化する方法が使われてきたが、アスペクトの組合せによっては適切な順序が存在しないことがある。

これらの問題を解決するために、本論文ではフィーチャの相互作用を実装するための言語機構を提供する 2 つの言語拡張、FeatureGluonJ と Airia を提案する。FeatureGluonJ は新しいフィーチャ指向言語で、Java と GluonJ の言語拡張である。FeatureGluonJ は継承をサポートする新しいフィーチャのためのモジュールシステムを導入する。スーパーフィーチャモジュールはそのサブフィーチャモジュールのインタフェースを定義する。FeatureGluonJ はこのインタフェースを用いてサブフィーチャ間の複数の組合せに再利用可能である汎用的なデリバティブを実装する言語機構を提供し、デリバティブの数を減らすことを可能にする。

Airia は AspectJ の言語拡張でアドバイスの合成のための新しいアドバイス resolver を提供する。Airia では、アスペクトの合成を衝突しているアスペクトとは独立した resolver に記述する。この resolver に記述されたコードは指定したアスペクトが衝突するときに実行される。Airia は AspectJ の proceed() 呼び出しを拡張し、衝突しているアスペクトの一部だけを優先度を与えて呼び出すことができる。Resolver では複数の proceed() 呼び出しの戻り値から、衝突しているアドバイスの組合せの最終的な戻り値を決めることができる。また、合成のための新しい言語機構を導入すると、この言語機構同士の衝突が新たな問題となることがある。Airia では resolver はアドバイスであるため、resolver 同士の衝突もまた、resolver で合成できる。

備考 ： 論文要旨は、和文 2000 字と英文 300 語を 1 部ずつ提出するか、もしくは英文 800 語を 2 部提出してください。
Note : Thesis Summary should be submitted in either a copy of 2000 Japanese Characters and 300 Words (English) or 2 copies of 800 Words (English).

（博士課程）
Doctoral Program

論 文 要 旨

THESIS SUMMARY

専攻： Department of	数理・計算科学専攻　専攻	申請学位（専攻分野）： Academic Degree Requested	博士 Doctor of　（理学）
学籍番号： Student ID Number		指導教員（主）： Academic Advisor(main)	渡辺　治
学生氏名： Student's Name	武山　文信	指導教員（副）： Academic Advisor(sub)	千葉　滋

要旨（英文 300 語程度）
Thesis Summary (approx.300 English Words)

A software product line is a family of similar software products built from common software artifacts. One approach to develop a product line is to decompose software into features and customize the resulting product by selecting a subset of those features. Feature-oriented programming is a paradigm that offers mechanisms for modularizing features. It allows implementing code related to a feature separately into a *feature module*. The key technique for the separation is destructive class extensions, which can extend the definition of existing classes from the outside of those classes. Developers generate products by combining feature modules. Aspect-oriented programming also supports destructive class extensions, and hence it is usable for implementing feature modules.

However, naive combination of feature modules may cause problematic behavior if the features interact with each other. Feature interaction is essential behavior between a specific combination of features. Programmers must implement interaction so that such combinations of feature modules work correctly. A modular approach is to implement interaction separately in *derivatives*, but the number of the derivatives is too large to maintain.

Interaction in aspect-oriented programming is conflict of aspects, which happens when multiple advices extend the same method. Programmers need to specify how conflicting advices are executed. An incomplete approach for advice composition used in the existing languages is to specify precedence order to conflicting aspect to linearize them. This is because some combinations of aspects do not have acceptable order.

To address these problems, this dissertation proposes two language extensions with mechanisms for implementing feature interaction: FeatureGluonJ and Airia. FeatureGluonJ introduces a new module system for features supporting inheritance. Inheritance of feature module enables to implement a generic derivative reusable for multiple combinations of features, which reduces the total number of derivatives.

Airia, a language extension to AspectJ, provides a novel kind of advice called a *resolver* for advice composition. Resolvers implement aspect composition separately from conflicting aspect. They can execute part of the conflicting aspects and merge the result of the execution by using proceed() call extended in Airia.

備考 ： 論文要旨は、和文 2000 字と英文 300 語を 1 部ずつ提出するか、もしくは英文 800 語を 2 部提出してください。
Note : Thesis Summary should be submitted in either a copy of 2000 Japanese Characters and 300 Words (English) or 2 copies of 800 Words (English).