バリアントソフトウェアの開発・保守支援に関する研究

渥美 紀寿[1,a)]

概要：本稿では，著者がこれまでに行ってきた，バリアントソフトウェアの開発支援，理解支援，保守支援に関する研究について紹介する．

1. はじめに

ソフトウェアの実行デバイスの多様化，同一機能を有する複数のライブラリの存在，利用者の機能に対する要求の違いなどに対応するため，ソフトウェアを多様な環境に適応できるように構成する必要がある．C 言語ではバリアントを構成する個々の要素をマクロを用いて表現し，前処理命令の条件命令 (以下では前処理条件命令と呼ぶ) を用いてソースコードを切り分けることでこれを実現する．

ソフトウェアの実行環境の多様化に伴ない，使用されるマクロが多くなり，そのマクロを利用した前処理条件命令の構成が複雑になる．その結果，ソースコードの可読性や保守性が低下している．本稿では，多数のバリアントを構成するソフトウェアの並行開発における支援手法，バリアント構成の理解支援，前処理条件と保守性の関係に関する調査に関する研究を紹介する．

2. バリアント並行開発の支援

多数のバリアントを構成するソフトウェアでは，コンパイルスイッチを用いて個々のバリアントを生成できるように，多くのマクロを用いて前処理命令を用いて制御する．そのため，制御に利用されるマクロ間の関係が複雑になり，前処理命令による分岐構造が複雑になる．その結果，特定のバリアントに対して修正や機能追加を行う場合，ソースコード中のどの部分が対応するか，そのコードに対する修正がが他のバリアントに影響しないかどうかを調査することが困難となる．また，並行開発では個々の開発者がそれぞれ改変を加えるため，同じコード断片を互いに変更した際，マージ時にコンフリクトが発生する．複数のバリアントを含むソフトウェアでは，プログラムのロジックに加え，前処理命令によるバリアントの切り替えを考慮しなければならないため，マージ作業が困難となる．

[1] 京都大学 学術情報メディアセンター
[a)] atsumi.noritoshi.5u@kyoto-u.ac.jp

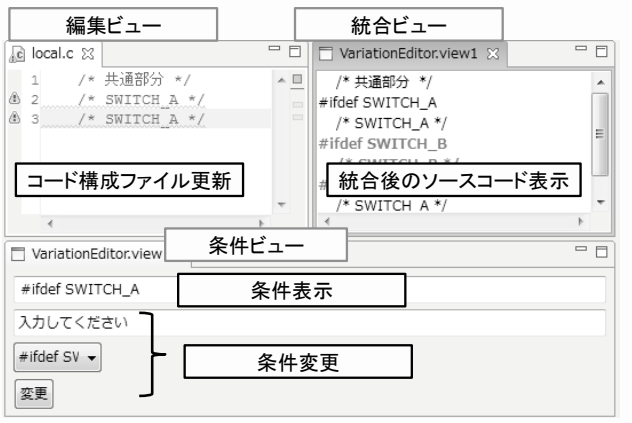

図 1 バリアントソフトウェア開発環境

我々はこれらの問題点を解決するために，バリエーションコードの抽出と統合手法を提案した [3]．また，本手法により抽出したバリエーションを利用してバリエーション開発を行うためのツールを統合開発環境 eclipse のプラグインとして開発した [1]．そのプラグインを使用した開発環境の画面を図 1 に示す．

本ツールではバリアントを考慮した版管理機構として，既存の版管理システムを拡張し，バリエーションを管理する枠組みを実現した．特定のバリアントコードの改変を支援するため，本ツールでは着目したいバリアントに関係のあるソースコードのみを抽出し，エディタ上に表示する．抽出の際，抽出前後の行の対応情報をファイル (コード構成ファイル) に記録する．バリアントコードの改変後，コード構成ファイルを利用して，改変内容をマージすることによってバリアントの統合を行う．また，コンフリクトした際，バリアントごとの変更内容を提示することによって，マージ作業を支援する．

3. バリアントを構成するフィーチャ間の関係

コンパイルスイッチを用いて複数のバリアントを構成するソフトウェアを構築する場合，ソースコード中にコンパ

イルスイッチに対応するマクロを用いた前処理命令があらゆる箇所に埋め込まれ，その構成が適切かどうかを確認することが困難である．これを解決するために，著者が所属する研究グループにおいて，ソースコードからフィーチャ間の関係を抽出し，フィーチャモデルを構築する手法を提案した [2]．本研究において，バリアントを構成するための前処理条件に利用されるマクロをフィーチャとして扱っている．

特定のバリアントを構成するフィーチャの組み合わせ方には様々な方法があり，単純にソースコード中の前処理命令による制御構造を抽出するだけではフィーチャ間の関係を適切に取得することができない．そこで，我々は，制御関係の出現頻度に基づいた依存度を定義し，依存度の高い関係のみを抽出することによって，フィーチャとは関係のないテストやデバッグ，ログ生成などのためのマクロによる細かい依存関係を除いたフィーチャモデルを構築する．

各バリアントはフィーチャの組み合わせによって表現され，その組み合わせは論理式で表現される．ソースコード中では前処理命令の条件命令 (`#ifdef`，`#ifndef`，`#if`，`#elif`，`#else`，`#endif`)(以下，前処理条件命令と呼ぶ) を用い，その条件式が論理式で記述される．

提案手法において，抽出する依存関係は (1) 前処理条件命令が入れ子になっている場合，その条件で利用されているマクロ間，(2) 前処理条件命令によってマクロが定義される場合，その条件で使用されているマクロと定義されるマクロ間の 2 つである．このマクロ A と B の依存関係 A → B における依存度を下記の通り定義する．

$$Dep(\mathtt{A} \rightarrow \mathtt{B}) = \frac{\mathtt{A}\text{ が }\mathtt{B}\text{ と同時に出現する頻度}}{\mathtt{B}\text{ の出現頻度}}$$

この依存度が高い関係のみを抽出し，マクロ間の依存関係をフィーチャ間の関係として捉えることによって，フィーチャモデルを構築する．OSS の openldap-2.4.31 を対象に提案手法を適用したところ，コンパイル時にユーザが設定可能な外部モジュールの中で利用するライブラリを選択可能な 4 種類のモジュールのうち 3 種類は適切にフィーチャ間の関係を分割でき，適切に分割できなかったのは 1 種類のみであった．得られたフィーチャモデルの例を図 2 に示す．

4. バリアント構成と改変頻度との関係

前処理条件命令によって切り分けられたソースコードの保守性は，ソフトウェアを構成するフィーチャが増えるに従って困難になる．フィーチャの組み合わせにより構成されるバリアントを実現するための前処理条件命令の構成方法は様々であるが，その構成方法は保守性に影響すると考えられる．そこで，保守性が低いバリアントでは，そのバリアントを実現するコード (バリアントコード) の改変頻度が高いと仮定し，改変頻度とバリアントを構成する前処理条件命令の構造との関係について調査した [4]．

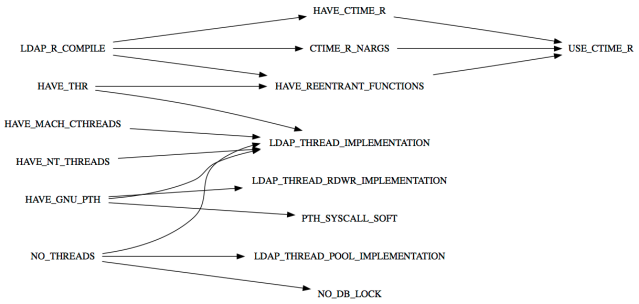

図 2 スレッドに関するフィーチャ間の依存関係

本調査では，(1) バリアントコードの改変頻度とそのバリアントコードを制御する前処理条件命令との関係，(2) バリアントコードの改変頻度とバリアントコードの行数との関係，(3) バリアントコードの改変頻度とバリアントコードを制御する前処理条件命令の分岐命令数との関係について OSS を対象に調査した．

その結果，特定の前処理条件命令によって制御されるバリアントコードは高頻度で改変されていることがわかった．そのコードまたは前処理条件命令の構成には保守性を低下させる要因があると考えられる．しかし，改変頻度とバリアントコードの行数，分岐命令数に関する調査結果では，改変頻度の高低との間で差異は見られず，保守性との関連を得ることができなかった．

5. おわりに

本稿では著者が行ってきたバリアントソフトウェアに対する開発・保守支援に関する研究について紹介した．我々はこれらの研究を基に前処理条件命令の構成情報およびそれによって制御されるバリアントコードの情報から，保守性の評価基準を定義し，保守性の定量的な評価を可能にすることを目指している．また，どのような構造が保守性が高いかを調査し，保守性が向上する構造へのリファクタリングに関する研究を進める．

参考文献

[1] 渥美紀寿，小林隆志，山本晋一郎，阿草清滋：バリエーション並行開発のための版管理機構，日本ソフトウェア科学会第 28 回大会論文集 (2011).
[2] 渥美紀寿，小林隆志，阿草清滋：プリプロセス命令の制御構造を利用したフィーチャ間の依存性解析，電子情報通信学会技術研究報告，Vol. 112, No. 373, pp. 67–72 (2013).
[3] 横山祐司，日高隆博，山本晋一郎，小林隆志，手嶋茂晴，阿草清滋：バリエーション並行開発のための版管理ツールと統合開発環境，情報処理学会研究報告ソフトウェア工学，Vol. 2010-SE-167, No. 6, p. 18 (2010).
[4] 今西洋二，渥美紀寿，森崎修司，山本修一郎，阿草清滋：バリアントコードの改変履歴に基づく前処理条件の構造に関する特徴調査，ソフトウェア工学の基礎 XXIII，pp. 115–120 (2016).