ウィジェット変更に伴うソースコード変更箇所の探索支援

深谷 和弘† 白銀 純子‡ 深澤良彰†
早稲田大学大学院理工学研究科† 東京女子大学現代文化学部‡

1 はじめに

近年、コンピュータに関する知識を持たない多くの人がコンピュータやソフトウェアを利用するようになっている。そのため、ソフトウェアの使いやすさ（ユーザビリティ）を向上させることが重要になっている。特に GUI（Graphical User Interface）は、ユーザとソフトウェアとの直接の接点であるため、ユーザが感じるソフトウェアのユーザビリティに大きく影響する。つまり、GUI のユーザビリティを向上させることが、ソフトウェアのユーザビリティの向上に大きく貢献する。

GUI のユーザビリティを向上させるには、GUI を作成し、それを評価し、その評価結果を反映して GUI を変更し、また評価する、という過程を繰り返すことが有効である。そして、GUI を変更する際には、GUI のレイアウトだけでなく、GUI 部品（ウィジェット）の種類を変更することもある。ウィジェットの種類を変更すると、変更したウィジェットとソフトウェアの処理部分の連結部分も、変更する必要がある。また、Java などのオブジェクト指向言語では、変更箇所が複数のクラスに及ぶことがある。そのため、大規模なソフトウェアでは変更箇所を手動で探索・修正することは、開発者にとって大きな労力となる。

そこで本研究では開発者の負担を軽減するために、変更箇所を自動的に探索し、提示する手法を提案する。

2 準備

本研究では、ソフトウェアのソースコードの変更箇所を探索するために、JavaML[1] と Program Dependence Graph(PDG)[2] を用いる。

1. JavaML

本研究では、構文解析ツールとして JavaML を用いる。JavaML は、Java ソースコードに XML を適用したもので、従来のような文字ベースでのプログラム表記と違い、JavaML は、XML ベースの構文を使ってネストされた要素としてソフトウェア構造を直接的に表現するため、プログラムの構造化表記に適している。

2. Program Dependence Graph

Program Dependence Graph(PDG) は、プログラムに存在する文を節点、文間の依存関係（データ依存関係，制御依存関係）を辺で表現した有向グラフである。本研究では PDG を用いてウィジェット変更時の影響波及部分の探索を実現する。

3 変更箇所の探索方針

本研究では、ウィジェットの数や GUI の操作上の観点から変更可能なウィジェット群の形態を「変更パターン」として定義する。またそれと合わせて、ウィジェットをその機能ごとに分類しておく。開発者はまず、変更パターンを選択し、その後、変更対象のウィジェットと変更後のウィジェットを選択することで、本システムがソースコードの解析を行う。

3.1 GUI の機能分類

変更前のウィジェットが決定されたとき、ウィジェットが持つ機能に応じて変更可能なウィジェットが限定される。従って、本研究では予めウィジェットをその機能ごとに分類し、変更可能なウィジェットを開発者に提示する。本研究におけるウィジェットの機能の分類は、「表示」、「選択」、「入力」、「アクション」、「コンテナ」の 5 つである。

また、それぞれのウィジェットの機能分類ごとに、同じ振舞いをするメソッドも存在する。従って、メソッドに関しても、同じ振舞いごとに分類しておく。

例えば、JTextField と JLabel は、ともに「表示」の機能を持ち、表示するテキストを取得するという振舞いをする getText() メソッドを持つ。

3.2 変更パターン

ウィジェットを変更する際にウィジェットの数や GUI の操作上、可能な変換の方法を「変更パターン」として定義しておく。開発者が変更パターンを選択することにより、より正確にアプリケーション処理部の変更箇所を探索することができる。現在定義している変更パターンは以下の 6 つである。

- 1:1 パターン:
  1 つのウィジェットを 1 つのウィジェットに変換
- 1:n パターン:
  1 つのウィジェットを複数のウィジェットに変換

A Support Method for Searching Influenced Parts in Modifying GUI Widgets
Kazuhiro FUKAYA†, Junko SHIROGANE‡ and Yoshiaki FUKAZAWA†
† School of Science & Engineering, Waseda University
‡ College of Culture and Communication, Tokyo Woman's Christian University

- n:1 パターン:
  複数のウィジェットを 1 つのウィジェットに変換
- n:n パターン:
  複数のウィジェットを複数のウィジェットに変換
- アクション合併パターン:
  あるウィジェットに「アクション」を要求するウィジェットを付加
- アクション分離パターン:
  「アクション」の機能が付加したウィジェットを元のウィジェットと「アクション」のウィジェットに分離

4 本システムの構成

図 1 に本システムの構成を示す。また以下で、本研究の各ステップを順に説明していく。

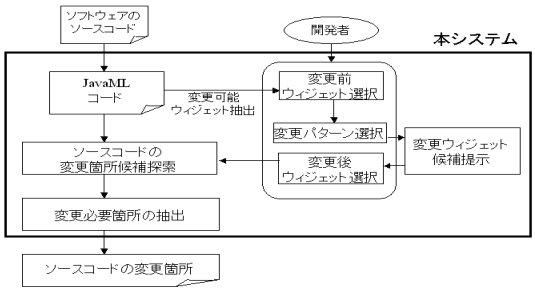

図 1: システム構成図

1. ソースコードからの変更可能ウィジェット抽出

対象のソフトウェアの Java コードを JavaML コードに変換する。次に、JavaML コードより、ソフトウェアの中で利用されているウィジェットを抽出する。

2. 変更前ウィジェット・変更パターン選択及び変更ウィジェット候補提示

本システムが、抽出したウィジェットを型と名前を提示する。次に、開発者は、ユーザの要求を考慮し、変更前ウィジェットと変更パターンを選択する。その後、本システムは、開発者による変更パターンと変更前ウィジェットの選択に基づき、変更後ウィジェットの候補を提示する。開発者は変更後ウィジェットの候補群から変更ウィジェットを選択する。

3. PDG の構築及びソースコードの変更箇所候補探索・提示

2. で与えられた変更前・変更後ウィジェットと変更パターンを元に、変更が必要な箇所を以下の 3 つのステップで探索する。

(a) ウィジェット宣言クラス内の探索

まず、変更前ウィジェットが宣言されているクラス内の PDG を構築する。構築方法を以下に示す。

一般に、文 s から文 t に対して変数 v に関するデータ依存関係があるとは、以下の 3 条件を満たすときを言う。

- 文 s は変数 v を定義する
- 文 t は変数 v を参照する
- 文 s から文 t への 1 つ以上の実行経路の中に、変数 v の再定義が起こらない経路が少なくとも 1 つ存在する

この変数 v として、変更前ウィジェット名を選択し、抽出した定義文と参照文の関係を解析し、PDG を構築する。変更前ウィジェットの宣言に対応した PDG 中の節点から、データ依存辺を経て推移的に到達可能な節点の集合が、このクラス内での変更箇所候補となる。

(b) 他のクラス内の探索

変更箇所は 1 つのクラスだけに収まるとは限らないため、他のクラスにおいても変更箇所を探索する必要がある。ここでは、(a) で探索したクラスを使用しているクラス群において PDG を構築し、探索を行う。

(c) 変更必要箇所の抽出

変更が必要な箇所は、主に変数宣言や代入、メソッドの引数・戻り値、そしてウィジェットが持つメソッド名などである。メソッドに関しては、どのウィジェットからどのウィジェットに変更するかによって、変更が必要な場合と不要な場合がある。例えば、JList を JComboBox に変更する場合、選択されている値を受け取るための getSelectedValue() は getSelectedItem() に変更する必要がある。しかし、ウィジェットの大きさを取得する getSize() は変更不要である。

従って、3.1 でのメソッドの分類を元に、(a) 及び (b) で抽出された変更箇所の候補から、変更が必要な箇所のみを抽出する。

5 おわりに

本研究では、Java 言語における GUI ウィジェット変更の際のアプリケーション処理部分への影響を考慮し、その変更箇所を自動で探索する方法を提案した。今後の課題として以下のようなものが挙げられる。

- 変更の対象となるウィジェットの種類の増加
- 変更箇所の探索・提示だけではなく、具体的な修正方法の提示
- 提示された変更箇所が妥当なものかどうかの検証

参考文献

[1] Greg J. Badros, "JavaML: A Markup Language for Java Sourse Code", 9th International World Wide Web Conference, 2000.

[2] Horwitz S., Reps T., "The Use of Program Dependence Graphs in Software Engineering", Proceedings of the 14th International Conference on Software Engineering, 1992.