ウィジェット操作の抽象化によるGUI部品の変更支援

4J−1

渡部 宏志　白銀 純子　深澤 良彰
早稲田大学　大学院　理工学研究科

1　はじめに

ソフトウェアは、機能(ユーティリティ)重視の傾向から使い勝手(ユーザビリティ)重視の傾向になってきている。これに伴い、ヒューマンインターフェース、認知心理学、人間工学等の分野に長けた人々がGUI(Graphical User Interface)デザイナとして求められるようになってきた。しかし、一般的に優れたGUIデザイナと優れたプログラマの素養は異なるため、GUIデザイナは、紙などでGUIプログラマに自分の意図するGUIを伝えることを余儀なくされている。これでは、GUIデザイナの意図を正確に伝えるために多大な労力が必要となるし、時間のロスにもつながる。

一方、従来はGUIの構築には複雑なプログラミングが必要であったが、RAD(Rapid Application Development)ツールや、自動生成の研究が行われ、煩雑なプログラミングの簡略化が図られてきた。しかし、最初に作成されるGUIはエンドユーザの意見が十分反映されているとは言い難いため、ユーティリティを満たしていても、ユーザビリティの面では不十分な可能性がある。そこで、ユーザビリティを重視してGUIを構築するには、エンドユーザの手によってGUIを評価・改良することが効果的である。そこでは、GUI部品(ウィジェット)も変更が必要となることがある。しかし、VisualAgeやJBuilderなどの開発環境は、依然としてプログラミング言語に依存した知識を必要とするため、GUIデザイナが利用することは困難である。アプリケーションのソースコードからGUIを生成し、そのGUIをカスタマイズする機能を提供するツールも研究されている[1]が、独自に作成したウィジェットなどを追加することは難しいため、ユーザビリティ向上のための変更を十分にサポートしているとは言い難い。

そこで本研究では、GUI変更に伴うプログラムの自動修正を行うための手法を提案し、簡単にGUIの変更を行うシステムを構築することで、洗練されたGUIプロトタイプの作成を行う。尚、本システムが対象とするユーザは、プログラムの知識をほとんど持たないGUIデザイナ、およびプログラマである。本研究では、アプリケーション本体の実装をする人をデザイナと区別してプログラマと呼ぶ。

Method for replacing GUI components by classifying widgets according to roles.

Koji Watanabe, Junko Shirogane and Yoshiaki Fukazawa.
Graduate School of Science & Engineering, Waseda University.

2　GUIプロトタイプ生成指針

2.1　GUIプロトタイプ

本研究で生成するGUIプロトタイプは、各ウインドウの外観やエンドユーザによるタスク実行の操作方法を評価・決定するために用いることを想定している。GUIプロトタイプは、GUIの外観の設定及び、アプリケーション処理部のメソッド呼出しとGUI部との連結で構成される。尚、メソッドは簡単な値を返すだけで、処理内容が記述されていないものとする。

2.2　ユーザビリティの向上

ユーザビリティを向上させるためには、次の点を考慮してGUIを構築する必要がある。

- 関連情報や実行させる処理(タスク)のグループ化
- タスク実行のためのユーザによる操作方式の決定
- レイアウト、情報の強調、等

以上を考慮したGUIを構築するために、本研究ではタスクをウィジェットの役割を中心に分析し、役割をウィジェットと結びつけることでGUIを構築する。

2.3　ウィジェットの役割

ウィジェットの機能としては、入力(選択による入力)、選択、表示、アクション要求(ユーザの操作により、何らかの処理を行うよう要求を発する役割)、コンテナ(GUI部品を配置する敷物の役割)の5つに大きく分けることができる。これらの機能を果たすために、ウィジェットには共通の処理が存在する。例えば、「選択による入力」ならば、「選択されている項目を返す」という処理により、ユーザの選択した値が、入力値として取得されることになる。本研究では、ウィジェットをこれらの機能で分類することで、同じ機能を果たすことのできるウィジェット同士を変更した際に生じる、プログラムの修正部分(データの参照、取得、などの共通な処理)を自動生成する。また、本システムでは、同じタスクを実行するためのGUIを複数生成することも容易に行うことができる。さらに、新しい部品も役割ごとに分類し、ウィジェット変換表に追加することにより、本システムで扱うことができる。

3　本システムの構成

本システムの構成を図1に示す。以下において、ファイル1〜4を選択して削除する例を用いる。この例のGUI部分を図2に示す。図2(a)の例は、JComboBoxにより削除するファイル名を選択して、JButtonの「OK」を

押すと選択されたファイル名のファイルが削除されるGUIである。図2(b)は、与えられた項目名(ファイル1～4)をファイルイメージと共に表示し、それらをごみ箱イメージを表示した部分にドラッグ&ドロップした瞬間にアクション要求(選択されたファイルを削除する)を発し、その項目名(選択されたファイル名)を取得できる、DnD-Selectというウィジェットを用いたGUIである。

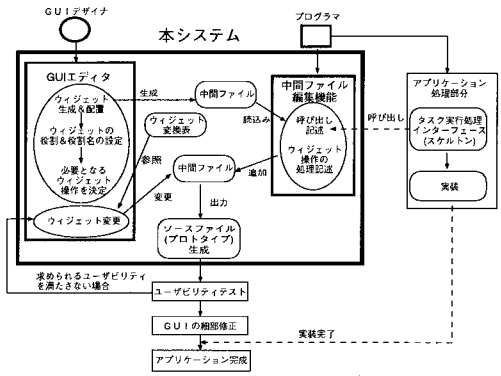

図 1: 本システムの構成図

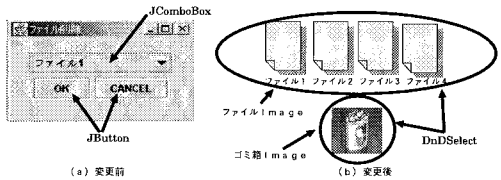

図 2: GUI変更例

3.1 プロトタイプ生成

デザイナは、アプリケーションのシナリオ等を参考にして、本システムのGUIエディタを用いて、ウィジェットを配置する時にそのウィジェットが果たす役割名を入力し、本研究で分類した5つの役割から、そのウィジェットが表す役割を選択する。例では、JComboBoxというウィジェットの役割名を「ファイル選択」、役割を「選択による入力」と仮定している。次に、アクション要求が起こった時に、処理が必要となるウィジェットに付けられた役割名を、そのアクション(タスク処理内容)と関連付ける。例では、「ファイル選択」が「削除アクション」に必要であると仮定して、関連付けている。デザイナがこのウィジェットの処理を記述する時は、関連付けされたウィジェットの役割である「選択による入力」の中から「選択されている項目を返す」という処理を選択する。例では、この処理の選択により「選択されたファイル名」を取得できることになる。

一方、プログラマは本システムの中間ファイル編集機能を使い、通常の処理メソッドの呼び出し記述の他に、デザイナが選択したウィジェットの処理記述を通常のメソッドと同じように使い、アクションに対する処理を完成させる(図3)。この内容が中間ファイルに追加される。

これによりデザイナがプログラミングすることなく、意図したウィジェットとアプリケーション処理との結合が可能となる。本システムでは、ウィジェットのレイアウト情報なども中間ファイル中に記述する。

これらの情報を元にしてプロトタイプのソースコードを自動生成する。

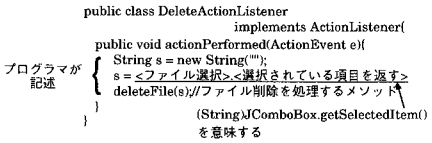

図 3: アクションに対するプログラマ記述例

3.2 プロトタイプ変更

デザイナはウィジェットに付けられた役割を参照しながら、同じ役割の新たなウィジェットをその役割名と付け替えることによりGUIを変更することができる。図2の変更例では、「ファイル選択」を「JComboBox」から「DnDSelect」に付け替えを行っている。システムは、予めウィジェットを役割ごとに分類し登録してあるウィジェット変換表(図4)を利用することで、中間ファイルに記述してあるウィジェットの処理を自動的に変換する。図4の例の「JComboBox」と「DnDSelect」は、予め同じ「選択による入力」の役割に分類され、その役割の「選択されている項目を返す」というウィジェットの処理に対して、「getSelectedItem」と「getDropedText」というメソッドがそれぞれ対応していると仮定している。返り値の型は、プログラマの指定により変換可能にする。DnDSelectウィジェットを使うとき、ファイルとごみ箱イメージは表示設定する必要があるが、基本的なウィジェットの処理(選択項目名を設定する等)は変換可能である。

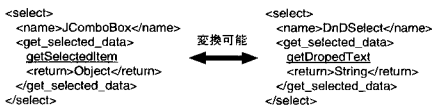

図 4: ウィジェット変換表(一部)

本システムで生成したプロトタイプのユーザビリティを評価し、その結果を反映させる形で、洗練されたプロトタイプを作成する。

4 終りに

本研究では、洗練されたプロトタイプを生成することを目的として、GUIの変更に伴うウィジェット変更時の依存性の自動再結合を実現する手法を提案した。今後の課題としては、以下のようなものが挙げられる。

- GUI変更の際にサポートできるウィジェット操作の範囲を広げること
- アプリケーション完成後にGUIを変更することのできるように本システムを拡張すること

参考文献

[1] 北村操代, 杉本明: 生成・カスタマイズ方式によるGUI構築手法の提案とクラスライブラリGhostHouseによる実現, 情報処理学会論文誌,Vol36,No.4,pp.944-957 1995.