5M－2

GUIアプリケーションプログラムからGUIオブジェクトを自動抽出する方法

小野 泰志
(株) 東芝

1. はじめに

コンピュータのユーザインタフェースは、GUI(グラフィカルユーザインタフェース)が主流である。それにともない、GUIアプリケーションを効率よく開発するためのツールも数多くリリースされている。特に、GUIビルダと呼ばれている、画面レイアウトを視覚的に定義し、ソースコードを出力するツールは、GUIアプリケーションの開発効率を大幅に向上させる。

しかし、人手で作成されたC言語アプリケーションにおいては、ウィンドウシステム間の移植や保守にGUIビルダーを使用できない場合が多い。ほとんどのGUIビルダーは、C言語を直接扱うことができないためである。

GUIビルダは、画面レイアウトを行なうものであり、CASEツールのようなアプリケーションの処理部分も含めた開発全体を支援するものではない。C言語の構文は、GUIビルダの対象である画面レイアウトを表現するには複雑すぎる。多くのGUIビルダは、汎用プログラミング言語ではなく、画面レイアウトを記述するための専用言語（ユーザインタフェース言語）を扱えるようになっている。

今回、C言語アプリケーションをGUIビルダで利用できるようにするために、C言語ソースコードからGUIオブジェクトを抽出し、その抽出結果をユーザインタフェース言語に変換し出力するツールを開発した。

<変換ツールの目的>

(1)他ウィンドウシステムへの移植作業の効率化

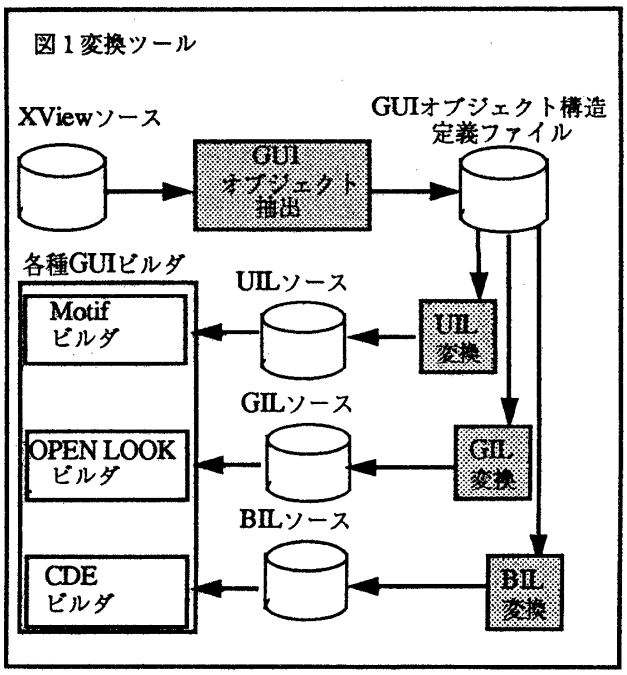
(2)既存ソースコードのGUIビルダによる再利用

本稿では、本変換ツールで用いたGUIアプリケーションソースコードからGUIオブジェクトを抽出する方法について述べる。

2. 変換ツール概要

変換ツールは、XView*アプリケーションを対象としている。XViewは、X WindowクライアントをC言語で開発するためのツールキット（ライブラリ、ヘッダーファイル）である。変換ツールの最終出力は、ユーザインタフェース言語である。（図1参照）

図1変換ツール



変換対象は、ユーザインタフェース言語で表現できる範囲とする。つまり、GUIオブジェクトの形状、位置、階層などを変換対象とし、動作の部分（例えばボタンを押されたときの処理）は対象外とする。

3. ソースコードからGUIオブジェクトの抽出

人手で作成されたGUIアプリケーションのソースコードは、GUI部分とそれ以外の処理が混在しているケースが多い。ユーザインタフェース言語が扱える情報は、GUIオブジェクトの形状、位置、階層などGUI構成情報だけである。そのため、C言語のソースコードから、GUIオブジェクトの情報のみを抽出する必要がある。GUIオブジェクト情報を抽出するには、ソースコードに記述されているXViewの関数とシンボル情報に注目すればよい。

XViewは、オブジェクト指向型のプログラミング環境を提供している。アプリケーションは、XViewが提供している部品（オブジェクト指向でいうクラス）を使用して、明示的にGUIオブジェクト（オブジェクト指向でいうインスタンス）の生成を行なう。アプリケーションのソースコード内のオブジェクト生成関数とそこで指定されている部品のシンボルを解析することで、GUIオブジェクトの抽出が可能となる。

Automatic Extraction of GUI Object from GUI Application Program
Yasushi Ono,
TOSHIBA corporation
＊XViewは、米国SunMicrosystems,Incの商標です。

4. GUIオブジェクト構造定義ファイル

抽出した結果を出力するファイルとして、固有のAPIに依存せずにGUIオブジェクトを表現できる中間ファイルを定義した。抽出したオブジェクト情報を、直接、ユーザインタフェース言語に展開することも可能ではあるが、固有のユーザインタフェース言語の制限を受けてしまう。例えば、UILはMotifのオブジェクトとその属性しか表現できない。

なるべく多くのGUIビルダで利用できるようにするため、特定のユーザインタフェース言語を対象とせず、中間ファイルから各種のユーザインタフェース言語を出力できるようコンバートプログラムを提供するようにした。（現在は、UILコンバータのみ実装完了）

5. GUIオブジェクトの属性情報と抽出率

ソースコードからGUIオブジェクトを抽出するにあたり、オブジェクトの属性情報を採取ができないケースが数多く見つかった。XViewのGUIオブジェクト生成関数は、いろいろな属性を引数で指定できるが、アプリケーションのソースコードを調べてみると、引数に変数を使用しているケースが多い。ソースコード解析だけでは、このようなアプリケーションのGUIオブジェクトの属性内容は抽出できない。

例えば、あるアプリケーションのボタン作成部分が、ボタンのラベル（ボタン内に表示されている文字列）に変数を使用していたとする。ソースコード解析によって得られるGUIオブジェクトの抽出結果では、このボタンのラベルは空白となってしまった。

ソースレベルで変数を追って、その変数の内容（この場合は文字列）を採取することは困難である。その変数が、どこでどのように定義されて、GUIオブジェクトの生成関数に引き渡されるかわからないためである。

このように、実行してみないとわからない情報を動的情報と呼び、これに対して、ソースコードのみから得られる情報を静的情報と呼ぶことにする。

GUIオブジェクトの抽出率を以下に定義し、いくつかのアプリケーションで計測した。

$$\text{抽出率} = \frac{\text{抽出できたGUIオブジェクト数}}{\text{全GUIオブジェクト数}}$$

抽出できたGUIオブジェクト数は、視覚的な属性値を採取できたものをカウントする、不完全に抽出したものはカウントしない

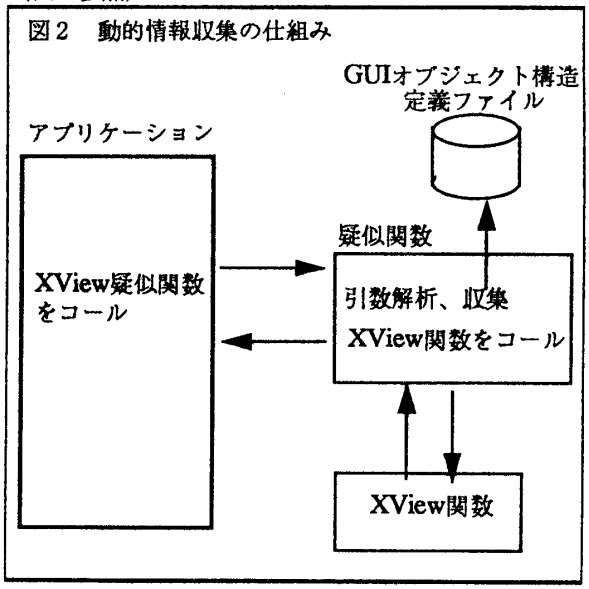
その結果、静的情報だけでは、抽出率は30~40%と低い値であった。特に、ウィンドウの位置や大きさを採取できないケースが多く、ユーザインタフェース言語に変換した後の、GUIビルダでの作業効率も良くない。

6. 動的情報の収集により抽出率を向上

静的情報だけでは抽出率が低いので、動的情報の収集を行なうよう工夫した。動的情報を収集するには、アプリケーションを実行させ、そのときに指定された属性値を取り出せばよい。

本ツールは、アプリケーションのXViewのオブジェクト生成関数と属性設定関数をコールしている行を、本ツールで提供している疑似関数を呼ぶように置き換える。置き換えたソースをコンパイル、リンクして実行可能形式ファイル（ロードモジュール）を作成し、それを実行することで動的情報を収集できるようにした。（図2参照）

図2　動的情報収集の仕組み



アプリケーションが持つすべてのGUIオブジェクトが表示されるように操作することで、GUIオブジェクト生成時の属性情報が収集できる。このようにして、動的情報収集を行なった結果、30~40%の抽出率であったアプリケーションが、70~80%と大幅に向上させることができた。

7. おわりに

現在、変換ツールはXViewからMotifへの移植作業支援に使用されており、移植作業の工数削減が見込まれている。今後は、GUIオブジェクトの抽出率の向上、ユーザインタフェース言語に変換されたGUIオブジェクトとアプリケーション固有の処理部分のリンクをどのように実現するかが課題である。

[参考文献]

[1]片岡　雅憲　著「ソフトウェア・モデリング」日科技連

[2]片岡　雅憲　著「ソフトウェア再利用技術」日科技連

[3]松本　正雄　編「ソフトウェアのモデル化と再利用」共立出版　bit 1995,3月号別冊