ウィンドウアプリケーションの開発支援

9H−1

長橋 賢児, 田中 英彦
{nag,tanaka}@mtl.t.u-tokyo.ac.jp
東京大学 工学部

1 はじめに

ウィンドウ環境に対応したアプリケーションの開発は、従来のアプリケーション開発よりも困難な作業であり、計算機による開発支援への要求が強い。開発支援の一方法としてユーザインタフェースの構成部品を汎用ツールキットライブラリパッケージとしたものを提供する方法が一般的になっているが、それを用いても、イベント駆動型のプログラム構造の弊害などが課題として残されている。

そこで筆者らはツールキットを用いたアプリケーションの開発においてプログラムの動作をより抽象的に記述できる言語を設計し、この言語による記述からアプリケーションコードを(半)自動的に生成するシステムを試作した。その概要を述べる。

2 イベント駆動型の問題

ツールキットを用いたウィンドウアプリケーションではライブラリとして提供される部品(ユーザインタフェースオブジェクト)の間の制御関係をプログラミングすることによってユーザインタフェースを構成する。このときプログラムはツールキット側からイベントの発生に対してアプリケーションコードを呼び出す、いわゆるイベント駆動と呼ばれる形態を採る。

イベント駆動型のプログラム構成は直接操作型のユーザインタフェースの実装には適したものであるが、ユーザインタフェースオブジェクト間の制御関係を記述するプログラマにとっては負担である。イベント駆動型プログラムではコードがイベントハンドラの集合体に断片化される。これらのイベントハンドラの間には所定の順序関係・階層関係が存在し、それらを何らかの方法で記述しなければならないが、通常のプログラム言語にはそれを明示的に記述する能力がないため細分化された平板な集合として記述せざるを得ない。そして関係するイベントハンドラ間で状態変数を共有するなど、間接的な方法によって順序関係や階層関係を実現している。従来の直接操作型でないアプリケーションでは、入出力の順序関係・階層関係などはコードの位置関係として明確に現れていた。しかしイベント駆動という形態では表現が間接的になり明確性が失われ、コードの相互関連が認識しにくくなる。このようなプログラム記述を強いられることがウィンドウアプリケーションでのプログラム記述・理解を妨げる一因であると考えられる。

そこで筆者らはイベントハンドラ間の制御関係を抽象的に記述する言語DCLを設計し、この言語による記述からツールキットを使用したアプリケーションプログラムのソースコードを生成するシステムを試作した。

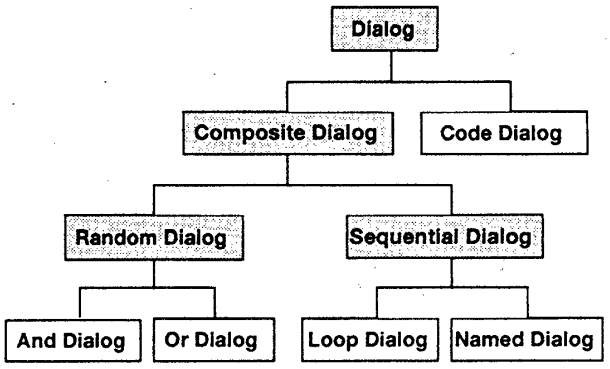

図1: ダイアログのクラス階層

このようなユーザインタフェースの制御部分を記述する言語としてはUIMS用言語として設計されたものが存在する[1, 2]。DCLはこれらの言語にヒントを得たが、特にイベント駆動型構造のもつ困難に注目した点が特徴である。

3 DCL

DCL(Dialog Control Language)はイベントハンドラ間に存在する順序関係、制御関係、階層関係などを記述する言語である。DCLの基本構成要素をダイアログと呼んでおり、アプリケーションのイベントハンドラの間の関係は基本的にダイアログの入れ子関係によって表現される。あるダイアログの中に含まれるダイアログをそのダイアログの子ダイアログと呼び、子ダイアログを含むダイアログを親ダイアログと呼ぶ。それぞれのダイアログは親ダイアログから呼び出され、ある一定の条件を満たすことによって終了する。これはユーザインタフェースにおいてある一連のユーザとアプリケーションプログラムの対話が開始され、終了することに対応する。

ダイアログにはそれが表す関係の性質によっていくつかの種類があり、図1のように階層的に定義されている(網かけされているダイアログは階層を定義するための概念的なもので、実際には使われない)。

以下各ダイアログの表現する関係を説明する。

- named **ダイアログ**
  呼び出されると対応付けられたイベントが発生するまで待ち、イベントを受けると子ダイアログを順次呼び出す。最後の子ダイアログが終了すると、ダイアログ自身が終了する。
- loop **ダイアログ**

An approach to supporting the development of windowed application
NAGAHASHI, Kenji and TANAKA, Hidehiko
University of Tokyo

```
dialog MatrixInput {
   local {
     int column_value;
     int row_value;
     Widget arraybase;
   }
   and {
     dialog ColumnEnter {
       { column_value =
         atoi(XmTextFieldGetString(widget)); }
     }
     dialog RowEnter {
       { ... }
     }
   }
   {
     arraybase = WcFullNameToWidget(w, "*base");
     ...
   }
   dialog InputMatrixElements {
     dialog MatrixInputQuit { }
   }
}
```

図 2: DCL による記述の例

呼び出されると子ダイアログを順次呼びだし、最後の子ダイアログが終了すると、一連の呼び出しを再度繰り返す。明示的なダイアログ終了命令によって終了する。

- **and ダイアログ**
  子ダイアログすべてを同時に呼び出し、全ての子ダイアログが終了すると終了する。
- **or ダイアログ**
  子ダイアログすべてを同時に呼び出し、その中のある一つが終了すると終了する。
- **code ダイアログ**
  特殊なダイアログで、子ダイアログでなくC言語による記述を内容として持つ。呼び出されるとその記述を実行して直ちに終了する。

and/or/loop ダイアログは制御用である。ユーザインタフェースの動作はnamed ダイアログを制御用のダイアログと組み合わせながら階層的に配置していくことで表現される。これは一般のプログラミング言語において動作が制御構文と手続き呼び出しの階層によって表現されることと表面的に同じになることを目標としたものである。図2にDCL言語で記述したアプリケーション(部分)の例を示す。この例は行列の大きさを指定後、その行列の各要素を入力する、という動作を記述している(codeダイアログの内容は一部省略されている)。

4 コード生成システム

コード生成システムはDCLによる記述から、イベントハンドラの集合である実際のアプリケーションプログラムコードへの変換を行なう。これは現在出力としてX windowのAthena widgetセットまたはOSF/Motifを用いるコードを生成するが、言語自体はユーザインタフェース部分とは独立であり他のツールキット、他の形態のユーザインタフェースのためのコードを生成するように変更することは容易である。

コード生成システムはCommon Lispを用いて実装されており、これにXツールキットのwidget階層構造の記述と、上記のダイアログ記述、さらにツールキットからのコールバックイベントとダイアログの対応関係およびwidgetとダイアログの対応関係の記述(それぞれ専用の言語が用意される)を与えるとwidget構成を生成するコードと、イベントハンドラのコードをC言語による記述として生成する。これを最終的にアプリケーションとして完成するにはさらに人手によるコードの追加が必要である。

widgetとダイアログの対応関係から、ウィンドウの表示、消去などに必要なコードがある程度自動生成されるので、widgetの階層構造の記述と、widgetとダイアログの対応を変化させればダイアログの記述を変更することなくユーザインタフェースの構成を変化させることが可能である。

5 おわりに

イベント駆動型プログラミングのもつ問題点に着目し、イベントハンドラ間の関係を記述する言語を設計することにより、ウィンドウアプリケーションの開発を支援することを試みた。

今後の課題としては以下のようなものが考えられる。

1. codeダイアログ中には使用するツールキットに依存したコードが記述されるため、ユーザインタフェースとアプリケーションコードの分離は不十分である。UIMSが目指すような各コンポーネントの分離を促すような設計を検討する必要がある。
2. codeダイアログの内容についてはDCLは関知しないため、記述の誤りの検出やダイアログ局所変数への参照制限、ツールキットとのインタフェース記述の支援は十分ではない。イベントハンドラの実装部分を言語にとりこむことにより、より高い支援能力を付加することを検討する。
3. 制御用ダイアログとして他に導入すべきものについて、またムダのない制御用ダイアログの組合せについてはなおも検討中である。

今後様々なアプリケーションの記述を試すことによってシステムの評価・改良を行なっていく予定である。またアプリケーションの他の部分のための言語についても考察していきたい。

本研究に際してはNTTソフトウェア研究所の小野諭氏他の方々より数々の有用な示唆をいただきました。感謝致します。

参考文献

[1] M. A. Flecchia and R. D. Bergeron. Specifying complex dialog in algae. In *Proceedings of CHI and Graphics Interface '87*, pages 229–234. ACM, 1987.

[2] R. D. Hill. Supporting concurrency, communication, and synchronization in human-computer interaction—the sassafras user interface management systems. *ACM Trans. Graph*, 5(3):179–210, 1986.