VACL(Visual Application Control Language(VACL) の研究*

4W－3

重定 如彦†　越塚 登‡　坂村 健§

東京大学大学院理学系研究科　東京大学大学院人文社会系研究科　東京大学総合研究博物館

1　はじめに

近年GUIの普及に伴いコンピュータの操作環境がユーザフレンドリーなものになってきている。しかし、GUIではテキスト言語を使ったユーザインタフェースと比べて、定型処理やアプリケーションの拡張を行うのが難しい。そこでGUI上で定型処理を効率的に行うことができ、GUIアプリケーションを容易に拡張することを目的としたスクリプト言語 Visual Application Control Language(VACL) の構築を行ったので報告する。

2　VACL が取り組む GUI の問題点

2.1　定型処理

GUIは一つの処理を行う場合にはメニューやパーツ等(以下対話オブジェクト)に対して直接操作を行うという直観的でわかりやすい操作方法をユーザに提供する事ができるが、定型処理のような処理を行うのはかえって困難である。例えばあるディレクトリ内のファイルをすべて印刷するといったような定型処理は数行のコードをテキストのスクリプト言語を記述し実行することができるが、GUIでは全てのファイルに対してメニューやパネル等の印刷の為の操作を行わなければならない。

2.2　アプリケーションの拡張に関する問題

アプリケーションプログラマーがエンドユーザの全ての要求をGUIアプリケーションに埋め込む事は不可能である。そこでスクリプト言語を使ってエンドユーザがアプリケーションをカスタマイズするというアプローチがある。そのような言語としてはEmacs Lisp、AppleScript、Tcl/Tk などが挙げられるがこれら既存のスクリプト言語には以下のような問題点が挙げられる。

まず既存のスクリプト言語はエンドユーザがプログラミングを行うには複雑すぎるものが多い。又GUIとテキスト言語環境ではシステムモデルが異なっている為、同じオブジェクトでも記述や指定の方法が異なっており、新たに両者の対応づけを覚えなければならない(図1)。

	GUI環境	テキスト言語環境
オブジェクトの表現	二次元	一次元　circle .
オブジェクトの指定方法	PDなどで直接指定	ID、名前などで指定
セマンティクスの起動	直接操作	コマンドで記述

図1. GUIとテキスト言語環境のシステムモデルの違い

次に、GUIアプリケーションの制御を行うスクリプト言語は通常アプリケーションとスクリプト言語間の特別なプロトコルを用いて制御を行う(図2)。この方法はアプリケーション側に余分なコードが必要であり、対応していないアプリケーションを制御することはできない。

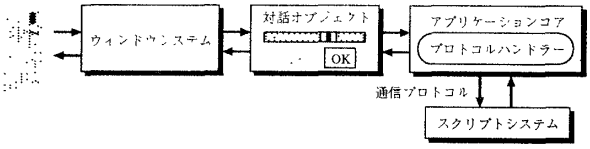

図2. 既存のスクリプト言語のアーキテクチャ

3　VACL のアプローチ

VACLでは以下のようなアプローチをとることで、上記の問題を解決する。

3.1　ビジュアルプログラミング

人間の思考は強いビジュアル指向にあり、絵のほうが文章よりも効率よく情報を得る事ができることが知られている[1]。そこでプログラムの記述性と可読性を高める為にVACLをビジュアルなプログラミング言語として

*The research of Visual Application Control Language

†Yukihiko Shigesada, Graduate School of Science, The University of Tokyo

‡Koshizuka Noboru, Division of Humanities and Sociology, The University of Tokyo

§Ken Sakamura, The University Museum, The University of Tokyo

設計した。VACLではGUIのオブジェクトの表現、指定、操作等の記述は、テキスト言語ではなく、GUIのオブジェクトそのものを使って記述する事ができる。例えばGUIアプリケーションのボタンを指定するにはそのボタンを直接そのアプリケーションからドラッグしてVACLのプログラムウィンドウへ取って来るという方法で指定できる。このようにVACLではGUIのシステムモデルをそのまま使いプログラミングを行う事ができる為、GUIとテキスト言語環境のシステムモデルの違いを考慮する事無くプログラミングを行う事が可能となる。

3.2 付箋メタファ

VACLではスクリプトプログラムの起動対象を特定するメタファとしてスクリプトが起動された時に作り出されるウィンドウを付箋と呼び、その付箋をスクリプトが対象とするオブジェクトに張り付ける事でスクリプトを起動するという付箋メタファをとっている。あるディレクトリ内のファイルをすべて印刷するといった定型処理はそのディレクトリのウィンドウにファイルを印刷する付箋を張り付けることで実行することができる(図3)。

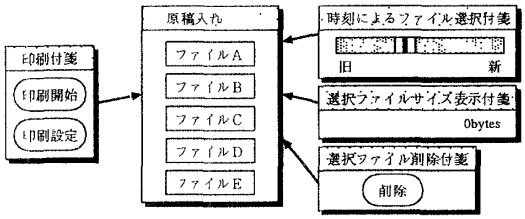

図3. 付箋メタファによるVACLスクリプトの実行例

又、一つのオブジェクトに同時に複数の付箋を張り付けて実行させる事も可能であるため、あるディレクトリの中にあるファイルを古い順に一定量削除するといったような複雑な定型処理も「指定した時間よりも古い時間に作成されたファイルを選択する」、「選択されているファイルのサイズを表示する」、「選択されているファイルを削除する」という3つのVACLスクリプトを組み合わせることで実現する事ができる(図3)。

3.3 アプリケーション独立なスクリプトプロトコル

本研究ではアプリケーションから独立にアプリケーションの制御を実現するために、VACLを東京大学坂村研究室で研究されてきた共有対話オブジェクトアーキテクチャ[2]上に構築した。このアーキテクチャではGUIアプリケーションは対話オブジェクトデータの格納庫(ウィンドウ実身(WRO))とそれを対等に共有するユーザインターフェースタスク(ユーザエージェント(UA))とアプリケーションタスクによって構成される(図4)。UAはユーザの入力に応じてWROの対話オブジェクトデータに変更を加えたり、WROの対話オブジェクトデータの変化を捉えて画面上に描画処理を行う。アプリケーションタスクはユーザの操作をWROから受け取りアプリケーションのセマンティックスを実行する。

VACLはWROを共有するタスクの一つとして実装され、対話オブジェクトに対して直接操作を加える事でアプリケーションを制御する(図4)。このようにVACLはアプリケーション本体にではなく直接対話オブジェクトに対して操作を加える事でアプリケーションの制御を行うのでアプリケーション本体に余分なコードを必要とせずにアプリケーションの制御が可能となる。

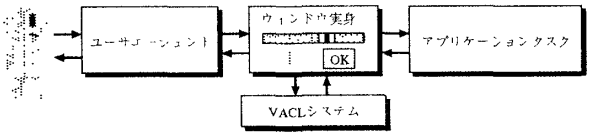

図4 システム構成図

4 おわりに

本研究ではGUI環境上で定型処理やアプリケーションの拡張を行うスクリプト言語の設計、実装を行った。又、付箋メタファの採用により複数のVACLスクリプトを同時に組み合わせて実行するというスクリプトの協調動作の機構を実現した。

参考文献

[1] G. Raeder. A survey of current graphical programming techniques. IEEE Computer, 18(8):11-25, Aug. 1985.

[2] N.Koshizuka and K.Sakamura. Window real-objects: a distributed shared memory for the distributed implementation of gui applications. In Proceedings of the ACM Symposium on User Interface Software and Technology, Nov.1993.