

オブジェクト指向言語 Bramble による 視覚的開発環境の構築

上田 賀一, 石川 裕喜, 中野 喜之

茨城大学 工学部 情報工学科

〒316 茨城県日立市中成沢町 4-12-1

コンポーネントウェアに示されるようにソフトウェアシステムの開発においてオブジェクト指向によるプロトタイピングアプローチが有用であると考えられる。これの実現に向け、本研究ではメタ階層アーキテクチャに基づくプロトタイピング方式によるモデルベースソフトウェア開発基盤の構築を進めてきている。この開発基盤を実現するために、動的振舞いや種々の関係を柔軟に記述でき、一貫したオブジェクト指向開発を可能とするモデル記述言語 Bramble を開発してきた。

今回、メタ階層アーキテクチャに基づき、Bramble 上に視覚的開発環境の構築を試みた。そして、この構築を通して、メタ階層アーキテクチャに基づく開発基盤が特定ドメインの環境をスムーズに構築できること、およびモデル記述言語 Bramble がアプリケーション開発環境を構築できるだけの十分な記述力を持つことを示すことができた。

Visual Development Environment constructed on Object-oriented Modeling Language : Bramble

Yoshikazu Ueda, Hiroki Ishikawa, Yoshiyuki Nakano

Ibaraki University

4-12-1 Naka-Narusawa, Hitachi, Ibaraki, 316 Japan

In a software system development, prototyping approach with object-oriented paradigm is useful as componentware shows. For realizing this, the authors have been developing a model-based software development environment which is based on meta hierarchical architecture and takes prototyping approach. And they have been developing an object-oriented modeling language 'Bramble' in order to describe this environment flexibly.

This paper presents a visual development environment constructed on 'Bramble' using the meta hierarchical architecture. This construction shows two facts. One is the smooth construction of the environment for a specific domain by meta hierarchy. Another is sufficient descriptive ability of 'Bramble' that can describe the visual development environment.

1. はじめに

本研究室ではメタ階層アーキテクチャに基づくプロトタイピング方式によるモデルベースソフトウェア開発基盤の開発を進めてきている[2]-[4]。この開発基盤はメタの階層構造を持ち、各階層を各レベルのユーザが受け持つ形態を取る。

コンポーネントウェア[1]によるアプリケーション開発環境の構築には、以下の理由によりメタ階層アーキテクチャが有用であると考えられる。

- コンポーネントウェアでは、コンポーネントによる部品としてオブジェクトのパッケージ化により、コンポーネント作成者とアプリケーション開発者に分離される。これは、メタ階層アーキテクチャにおける各階層を各レベルのユーザが受け持つ考えに合致する。
- コンポーネントウェアによるアプリケーション開発には、プロトタイピングアプローチが適しており、メタ階層アーキテクチャを用いるとプロトタイピングが行いやすい。

メタ階層アーキテクチャに基づいた視覚の開発環境を構築するためには、メタ階層アーキテクチャの概念を損なわずに実装を行う必要がある。これを実現するための言語として、オブジェクト指向モデル記述言語 Bramble を開発している[5]。

本研究では、メタ階層アーキテクチャに基づいて、Bramble におけるコンポーネントベースのアプリケーション開発環境の構築を行うこととし、以下の目標を設定した。

- 視覚的プログラミングによるアプリケーション開発環境の構築
- メタ階層アーキテクチャに基づいたアプリケーション開発環境の構築

本研究では、バージョンアップに伴い仕様変更されたモデル記述言語 Bramble を概説し、これらの目標が、Bramble 上に如何に構築され、達成されたかを報告する。

2. Bramble の特徴と仕様

2.1 言語の特徴

本言語の特徴を以下に挙げる。

- 全ての要素がオブジェクト。
オブジェクトの集合は、その集合で一つのオブジェクトとして定義され、言語自身が振舞うために必要な要素もオブジェクトとして持つ。

- コピー操作を主体としたオブジェクト指向。
クラス-インスタンスの概念が無い。
- オブジェクトは属性だけを包含。
オブジェクトは変数とメソッドを区別せず、属性として包含する。
- GUI 構築、DB I/F のためのオブジェクト群。
GUI 部品のオブジェクト群により視覚的なモデル記述の支援が可能であり、データベースへのアクセス手段を持つことで生成したオブジェクトの管理が容易になる。

2.2 基本オブジェクトとオブジェクトタイプ

本言語の基本的な機能を持つオブジェクト群には、次のものがある。

- システムオブジェクト (system)
- 真偽値オブジェクト (true, false)
- ニルオブジェクト (null)
- GUI 構築支援のウィジェットオブジェクト群
- DB I/F であるデータベースオブジェクト

さらに以下のオブジェクトタイプがある。

- ユーザオブジェクト
- 文字列オブジェクト
- リストオブジェクト
- 配列オブジェクト
- ファイルオブジェクト

2.3 文法

プログラムコードの単位は文であり、以下の3種類がある。

- メッセージ送信

プログラムコードの基本的な単位は、メッセージ送信である。これは、送信先オブジェクト、メッセージ名、メッセージ実行に必要なパラメータを特定することによって行われる。

メッセージ送信により、対象となるオブジェクトのメッセージ名と同名のメソッドオブジェクトが起動される。このメソッドオブジェクトの評価値が、戻り値となる。

メッセージ送信は次のように記述される。

`object message-name tag-name: parameter ...`

パラメータはメッセージ送信後、起動されたメソッドのローカルオブジェクトとして tag-name という名前 parameter で示されたオブジェクトを参照するように設定される。

● オブジェクト

オブジェクトもプログラムコードの単位となり、以下の表記に従う。

変数名	ローカル変数に格納されたオブジェクト
& 属性名	属性のオブジェクト
"..."	文字列オブジェクト
[object object ...]	リストオブジェクト
[要素名:object ...]	(連想)配列オブジェクト
{ ... }	メソッドオブジェクト
true, false	真偽値オブジェクト
null	ヌルオブジェクト
local	ローカル変数が格納されている配列オブジェクト
self	メッセージを受け取ったオブジェクト
alias	selfが配列としてのメソッドを持ったもの

● 代入文

代入文は、内部で属性変更(set)に直され、メッセージ送信される。また、代入先の属性が、'&'付き表記の場合はその属性が変更され、'&'無しの場合はローカル変数として保持する配列オブジェクトであるローカルオブジェクトに追加/変更される。

```
&attribute-name <- object
```

```
variable-name <- object
```

戻り値はオブジェクトであり、括弧“(, ”)”で括弧することによってメッセージ送信や代入文の一部として使うことができる。

BNF 記法により本言語の文法を以下に示す。

```

<program> ::= <method>
<method> ::= <sentence>
              | <sentence> ; <method>
<sentence> ::= <message>
              | <substitution>
              | <object>
<substitution> ::= <variable> <-> <object>
                  | <variable> <-> <message>
<object> ::= STRING
           | DIGIT
           | <variable>
           | ( <sentence> )
           | [ <array> ]

```

```

           | [ <list> ]
           | { <method> }
<variable> ::= SYMBOL
           | & SYMBOL
<message> ::= <object> <symbol>
           | <object> <symbol> <object>
           | <object> <symbol> <array>
<array> ::= <symbol> : <object>
          | <symbol> : <object> <array>
<list> ::= <object>
          | <object> <list>
<symbol> ::= DIGIT
           | SYMBOL

```

2.4 オブジェクトの操作

ここでは、各オブジェクトタイプが持つ、メソッドの種類を示す。

- 全てのオブジェクトがもつメソッド
コピー操作, 同一性比較
- システムオブジェクト(system)
処理系の機能利用操作, オブジェクト生成
- 真偽値オブジェクト(true, false)
論理演算, 条件操作
- 文字列オブジェクト
算術・比較演算, 一般的な文字列操作, 文字列評価, メソッド変換, 多岐選択操作
- リストオブジェクト
一般的なリスト操作, 全要素反復操作
- 配列オブジェクト
一般的な配列操作
- メソッドオブジェクト
メソッド評価, 反復操作
- ファイルオブジェクト
一般的なファイル操作
- ユーザオブジェクト
属性の追加(変更), 属性の取出し, 属性の評価

2.5 オブジェクトの記述

本言語によるオブジェクトの作成方法を示す。

- オブジェクトの生成
一般的なオブジェクトの生成は、システムオブジェクトに template "user" を送信することにより得られたオブジェクトに属性を追加する。具体的には、

```

tv <- system template "user";
tv set name:"channel" value:"NHK";
tv set name:"power" value:"OFF";

```

```
tv set name:"volume" value:"0";
```

とすると, channel, power, volume という属性を持った tv というオブジェクトができる。属性はそれぞれ, NHK, OFF, 0 という文字列オブジェクトを参照している。

- メソッドの作成と追加

メソッドの追加は, 他の属性と同じように属性追加(set)によって行われる。

```
tv set name:"on" value:{ power<- "ON"; };
tv set name:"off" value:{ power<- "OFF"; };
tv set name:"copy" value:{
  newobj <- system template "user";
  newobj set name:"channel" value:&channel;
  newobj set name:"power" value:&power;
  newobj set name:"volume" value:&volume;
  newobj set name:"on" value:&on;
  newobj set name:"off" value:&off;
  newobj set name:"copy" value:&copy;
  newobj;
};
```

こうすることで, オブジェクト tv に, on, off, copy というメソッドが追加される。

このオブジェクトは, 以下のようなプログラムで利用することができる。

```
tv on;
tv2 <- tv copy;
tv2 off;
```

2.6 GUI 構築支援および DB I/F の機能

本言語では, グラフィックスインタフェースの構築を支援するための機能を持ったオブジェクト群を提供する。現在, このオブジェクト群は, Tk を用いて実現している。また, データベースを対象とした操作のために, データベースとのインタフェースの機能を持つオブジェクトを提供する。本研究ではオブジェクト指向データベースである MATISSE を使用している。

3. 視覚的開発環境の設計

3.1 VisualBramble の設計方針

本開発環境の構築における目標を達成するために, 本研究では, 視覚的開発環境を構築する上で, 以下の二つに関して設計を行った。

- 本開発環境により開発されるアプリケーション
本開発環境を利用するユーザの負担を軽減するために, コンポーネントを基準とするアプリケーションの開発を進める。これにより, ユーザは必要なコンポーネントを用意し, コンポーネント間のメッセージの流れを指定するだけで, アプリケーションの開発が可能となる。

- 本開発環境を構成するエディタ

コンポーネントの構築を中心に考え, それをサポートするエディタが必要となる。そこで, アプリケーションを構成するコンポーネントを視覚的に表示し, より開発効率を上げるために, コンポーネントを視覚的に構成できるエディタを開発する。

これらを踏まえて, 本開発環境の設計を行った。なお, 本開発環境により開発されるアプリケーションを, VisualBramble アプリケーション(以下, VB アプリケーション)と呼び, 本開発環境を構成するエディタを総称して, VisualBramble 開発環境(以下, VB 開発環境)と呼ぶ。

3.2 VisualBramble の基本設計

本研究の目的の一つに, メタ階層アーキテクチャを用いたモデル例の作成がある。そこで, 本開発環境のメタモデルを ERF モデルを用いて記述した。それを図 1 に示す。

3.3 VisualBramble アプリケーションの設計

3.3.1 概要

本開発環境により開発されるアプリケーションは, コンポーネントの組合せで構成される。ユーザは, アプリケーションに必要なコンポーネントを用い, それらのコンポーネントのプロパティを設定し, 動作をメソッドに記述する。

さらにインタフェースを用いて, メッセージを送りたいコンポーネント間を接続することにより, アプリケーションを開発できる。

アプリケーション開発の様子を図 2 に示す。

3.3.2 コンポーネント

VisualBramble におけるコンポーネントは, 図 3 に示すように次の要素により構成される。

- プロパティ
- メソッド
- インタフェース
- サブコンポーネント

コンポーネントは、その性質により以下の3種類に分類される。

- 空コンポーネント

ユーザの作成するコンポーネントの基本となるコンポーネントである。これは、全てのコンポーネントが有すべきプロパティのみをプロパティリストにより保持する。

- テンプレートコンポーネント

ある機能を実現するために必要と思われる最低限のプロパティとメソッドを持つコンポーネントである。

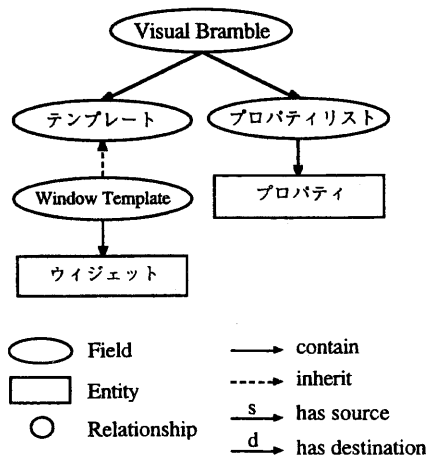
- 既成コンポーネント

既にプロパティとメソッドが完全に用意されているコンポーネントである。これはアプリケーションに組込むだけで使用できるが、プロパティやメソッドは追加できない。

コンポーネントは基本的に本開発環境が提供するが、ユーザ側が用意することもでき、これを開発環境に組込むことも可能である。この場合、テンプレートコンポーネントか、既成コンポーネントとして登録される。表1に、これら三種類のコンポーネントの特徴を示す。

表1. 各種コンポーネントの特徴

コンポーネントの種類	空	テンプレート	既成
メソッドの追加, 変更	可	可	不可
プロパティの追加, 変更	可	可	不可
インタフェースの追加	可	可	可
追加, 変更無しで使用	不可	可(一部)	可



3.3.3 プロパティ

プロパティは、コンポーネントの状態を保持するものである。プロパティは、プロパティの内容を示すプロパティ名と、プロパティの状態を保存するプロパティ値から成り、プロパティ名とプロパティ値を一組として、プロパティリスト内に保持される。コンポーネントは、このプロパティリストを保持することにより、プロパティを管理している。プロパティの内容は、他のコンポーネントから直接取得することはできず、取得したいプロパティの内容を持つコンポーネントのプロパティ取得メソッドを介さなければならない。

3.3.4 メソッド

コンポーネントが受け取るメッセージに対する動作が記述される。メソッドは、インタフェースを持っており、他のコンポーネントにメッセージを送る場合は、インタフェースを介することになる。メソッドの持つインタフェースにメッセージを送ることにより、インタフェースの接続先にメッセージを送ることができる。なお、メソッドの記述はオブジェクト指向モデル記述言語Brambleを用いる。

3.3.5 インタフェース

インタフェースは、コンポーネント間で受け渡されるメッセージの出入り口である。コンポーネント間で受け渡すことのできるメッセージを規定するために用いる。インタフェースは、必要に応

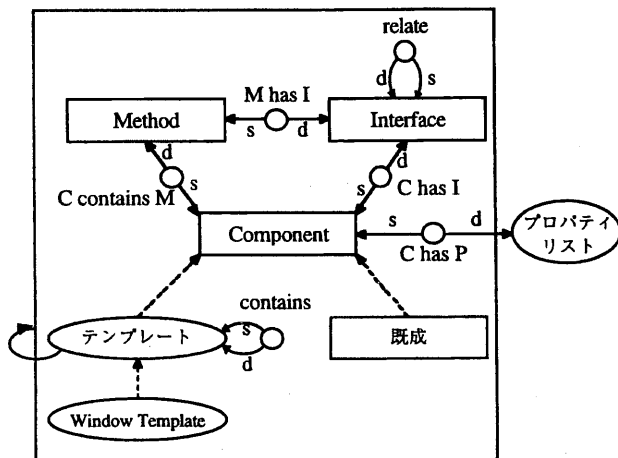


図1. 視覚的开发環境 Visual Bramble のメタモデル

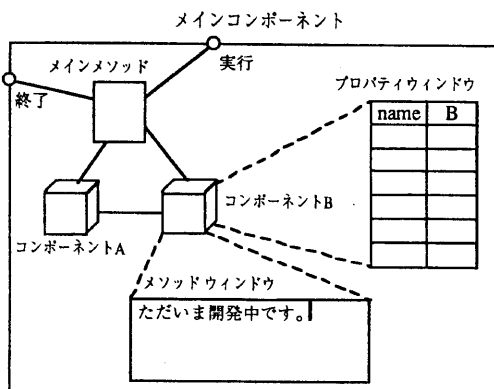


図2. アプリケーション開発のイメージ

じてユーザによってコンポーネントやメソッドに追加される。インタフェースは、メッセージ送受信のプラグであるから、追加しただけではメッセージの送受信を行うことはできない。インタフェース追加後は、メッセージの送受信を行いたいコンポーネントやサブコンポーネント、メソッド同士を接続する必要がある。インタフェースは、常に一對一に接続されていなければならない、一對多という接続は許されない。

3.4 VisualBramble 開発環境の設計

本開発環境は図4に示すように5種類のエディタから構成される。これらは、ビジュアルアプリケーション開発環境を構築するために必要であり、VBアプリケーションを開発するために用意されるものである。VB開発環境は、ユーザによって開発されるアプリケーションを、プロジェクトという単位で管理する。このプロジェクトは、ユーザにより開発されるアプリケーションを構成するコンポーネントを VisualBramble が管理するために、VB 開発環境が用意するものである。

VB開発環境上のエディタについて簡単に述べる。

- メインメニュー
各エディタを管理し、VB開発環境の設定を行うためのメニューの集合である。
- コンポーネントエディタ
コンポーネントを生成、削除、インタフェースの追加、接続、メッセージの送受信の決定などの機能を持つエディタ。各コンポーネントがそれぞれ持っているので、VB開発環境上に複数立ち上げることができる。

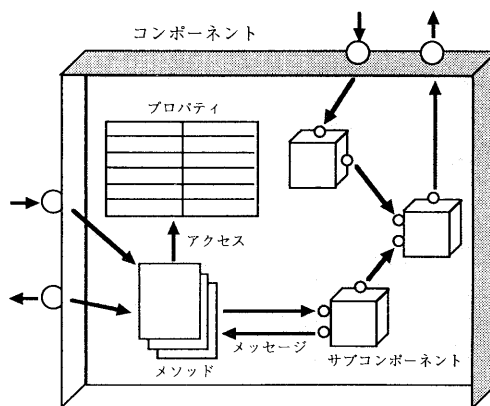


図3. コンポーネントの構成

- プロパティエディタ
各コンポーネントのプロパティを表示し、プロパティ値を設定するために用いるエディタ。VB 開発環境に常に一つしか存在しない。
- メソッドエディタ
各コンポーネントが持つメソッドを記述するためのエディタ。各メソッドがそれぞれ持っているの、VB 開発環境上に複数立ち上げることができる。
- GUIエディタ
GUI関連のコンポーネントが持つエディタで、GUIを構築するために用いる。

4. 視覚的開発環境の実装

本開発環境の実装は、オブジェクト指向モデル記述言語 Bramble により行った。

ここでは、本開発環境を実装する上で用いた、実装方法を簡単に説明する。

4.1 メタ階層アーキテクチャに基づいた実装

本開発環境を構築する上で、メタ階層アーキテクチャに基づいた設計を実装レベルで実現しなければならない。本開発環境の実装においては、メタオブジェクトを用意し、それらに生成メッセージを送ることにより、オブジェクトが生成されるようにした。

本開発環境構築において用意したメタオブジェクトは、次のものである。

- コンポーネントメタオブジェクト
- メソッドメタオブジェクト
- インタフェースメタオブジェクト

- 各種テンプレートコンポーネントのメタオブジェクト
- 各種既存コンポーネントのメタオブジェクト
- VB メニューメタオブジェクト
- コンポーネントエディタメタオブジェクト
- メソッドエディタメタオブジェクト
- プロパティエディタメタオブジェクト
- GUIエディタメタオブジェクト

VB アプリケーション開発において、コンポーネントを生成する場合、次の手順で、コンポーネントの生成を行っている。

1. 生成したいコンポーネントのメタに対して、生成メッセージを送る
2. メッセージを受け取ったメタオブジェクトは、新たなコンポーネントを生成する。
3. メタオブジェクトは、生成したコンポーネントをメッセージ送り先に戻り値として渡す。

このようにして実装レベルにおいて、メタ階層アーキテクチャに基づいた VisualBramble の設計を実現している。

4.2 コンポーネントの内部構造

本開発環境の設計において、コンポーネントが VB 開発環境のエディタを持つよう設計した。そ

こで、本開発環境を構築する上で、コンポーネントに、VB 開発環境が必要とするオブジェクトを属性として持たせる形で実装した。これに伴い、コンポーネントは、VB 開発環境上のアプリケーションが必要とするオブジェクトを属性として持つことになる。エディタが必要とするオブジェクトは、主に VB 開発環境上のコンポーネント、メソッド、インタフェースを管理するためのオブジェクトである。また、コンポーネントの属性を管理する目的もある。

4.3 VisualBramble 開発環境の参照

本開発環境を構成するエディタは、メタオブジェクトに生成メッセージを送ることにより生成される。これらのエディタは、コンポーネントやメソッドの属性となるため、コンポーネントやメソッド生成の際に、任意のメタオブジェクトへの参照を必要とする。

そこで、本開発環境の実装では、VisualBramble オブジェクトを生成し、各種メタオブジェクト、VB メニューを参照し、プロジェクトを VisualBramble オブジェクトの属性として管理した。そして、VisualBramble オブジェクトを、生成される全てのコンポーネントに参照させる。

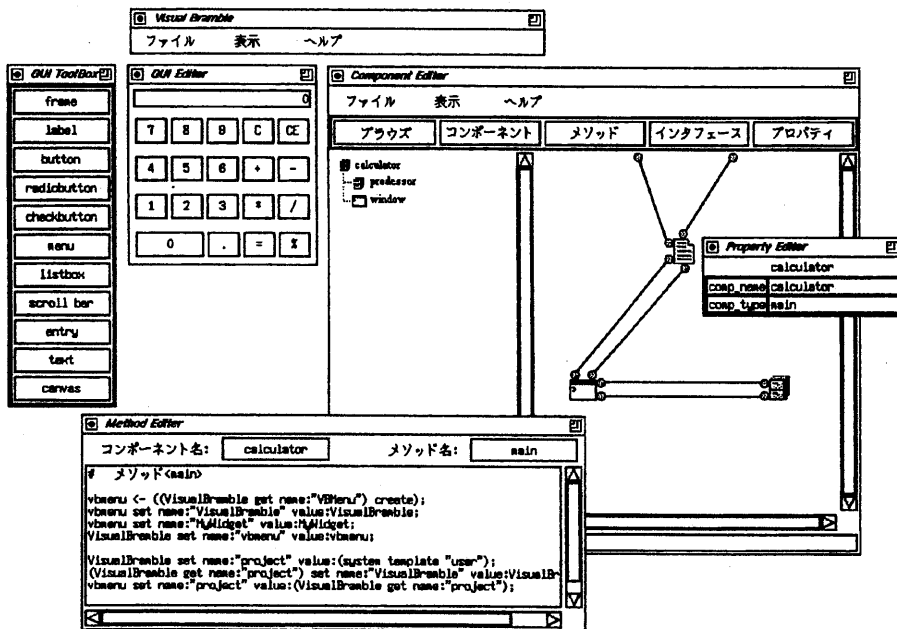


図 4. VisualBramble の画面

これにより、本開発環境の管理を実現し、任意のメタオブジェクトとVB開発環境をコンポーネントやメソッドから見えるようにしている。

5. アプリケーション開発例

開発例として、電卓アプリケーションを作成した。電卓アプリケーションは、本開発環境が提供するGUI関連のコンポーネントであるラベルコンポーネントとボタンコンポーネント、テンプレートコンポーネントであるウィンドウコンポーネント、空コンポーネントから作成した電卓アプリケーションの処理を行うコンポーネントから構成されている。

以下に、本開発環境を用いた開発手順を簡単に説明する。

1. VisualBrambleを起動し、VBメニューから「新規作成」を選択する。
2. プロジェクト名を入力し、メインコンポーネントを選択する
3. コンポーネントエディタにwindow_templateコンポーネントを追加する。
4. windowコンポーネントから、GUIエディタを立ち上げる
5. GUIエディタを用いて、GUI部分を作成する。
6. コンポーネントエディタに処理用コンポーネントを追加する。
7. processorコンポーネントのメソッドを記述する
8. 各コンポーネントのプロパティを設定する。
9. インタフェースを追加、接続し、メッセージの流れを設定する

このように、簡単に電卓アプリケーションを作成することができた。これにより、本開発環境の

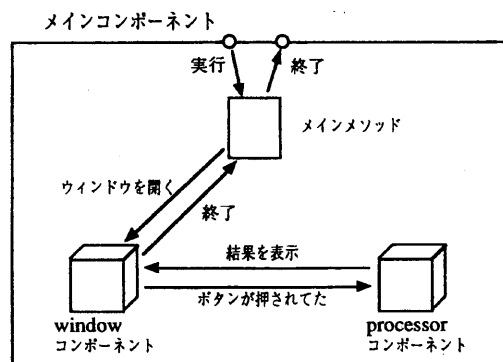


図5. 電卓アプリケーションのメッセージの流れ

アプリケーション開発能力がGUIアプリケーションドメインにおいて十分であり、本開発環境でのアプリケーション実行が可能であることを示すことができた。しかし、本開発環境を用いて作成したアプリケーションは、現在のところ本開発環境上でのみ実行可能である。今後の課題として、本開発環境上で作成したアプリケーションを本開発環境外で実行できるようにする必要がある。

6. まとめ

本研究では、メタ階層アーキテクチャに基づき視覚的プログラミングによるアプリケーション開発環境を構築することができた。

本研究により、Brambleがアプリケーション開発環境を構築できるだけの十分な記述力を持つこと、およびメタ階層アーキテクチャを用いることによるスムーズなアプリケーション開発を例証することができた。しかし、開発環境としてはコンポーネント作成における原則の欠如、コンポーネント使用時のユーザに対するサポートの不足などの問題があり、これらの改良による本開発環境の充実が必要である。

参考文献

- [1] 青山幹雄：“コンポーネントウェア：“部品組み立て型ソフトウェア開発技術”，情報処理，Vol.37, No.1, pp.71-79 (1996)
- [2] 安間その美，高橋大輔，上田賀一：“システムプロトタイピング支援のための基盤機構の開発”，情報処理学会ワークショップ論文集，Vol.95, No.1, pp.49-56 (1995)
- [3] 上田賀一，安間その美，高橋大輔：“メタ階層に基づくモデルベースソフトウェア開発基盤の提案”，ソフトウェア工学の基礎II, pp.11-20, 近代科学社 (1996)
- [4] 高橋大輔，上田賀一：“メタ階層に基づくモデル構築とその支援環境”，ソフトウェア工学の基礎III, pp.146-149, 近代科学社 (1996)
- [5] 上田賀一，中野喜之，金村星吉，高橋大輔：“オブジェクト指向モデル記述言語Brambleの開発”，情報研報(SE), Vol.96, No.32, pp.65-72 (1996)