

5Z-4 オブジェクトフロー型概念によるソフトウェア開発技法への一提案

VBによるセマンティックギャップ短縮のためのビジュアル化と自動化

伊藤貴雄

帝京平成大学大学院情報学研究科

平嶋史武

(株) ソフテック

黒坂功

帝京大学大学院理工学研究科

永井正武

帝京大学理工学部

1. はじめに

本論文は、オブジェクトフロー型ソフトウェア開発技法の一つ^[1]として、マウス操作によりビジュアル的なインタフェース技法について実現するものである。本提案では、VB クラスモジュールを自動生成し、作成時のイメージしやすい環境、ソフトウェアの再利用、セマンティックギャップの短縮を目的とする。オブジェクトフロー型とは、計算モデルとして、データ駆動方式^[2]の概念をし、計算や演算をオブジェクトレベルに工夫・考案した計算処理方式である。

2. オブジェクトフローによるプログラミング

オブジェクトフローによるプログラミングはコーディング作業とオブジェクトフロー図の作成作業の2つの連携作業によって行われる。コーディングによりオブジェクトフローの最小単位の作成する。

フロー図作成作業は、図1のようなユーザインタフェースをもち、マウス操作により、プログラムの流れをオブジェクトフロー型に記述していく。

3. オブジェクトの型

この開発環境では、オブジェクトの型として、2つの種類がある。1つは、入出力データ型、もう1つは計算処理オブジェクト型である。例えば、図1には、入出力データ型として、整数型 (IOinteger)、浮動小数点型 (IOdouble) の2つを用意し、処理オブジェクト型として、整数型の加算 (CalcAdd)、浮動小数点型の加算 (CalcDoubleAdd)、を用意している。これらは、コーディング作業により自由に作成することができる。整数型の加算 (CalcAdd) は、入力が2つの出力が1つでそれぞれ、整数型 (IOinteger) で入出力が行われるように作ってある。これらをつなぎ合わせることによってソフトウェア開発を行っていく。

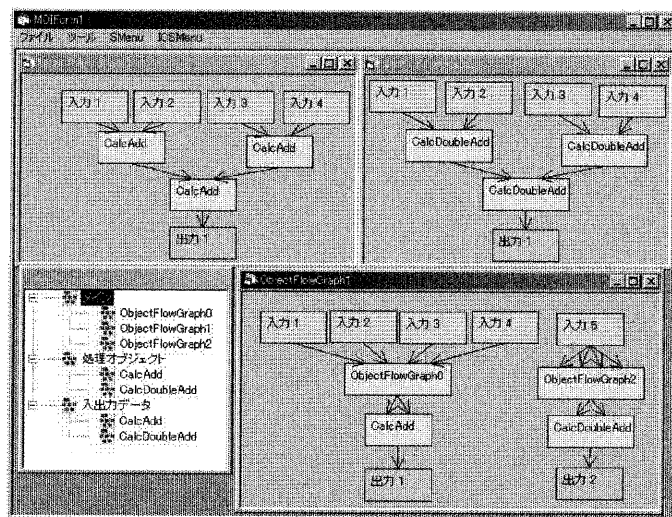


図1. オブジェクトフロー図作成画面

(1) 入出力データ型

リスト1に入出力データ型のコーディング例を示す。

このクラスは、a,b,cの3つの整数型をもつものである。

リスト1. IOdata.cls

```
Public a As Integer
Public b As Integer
Public c As Integer
```

(2) 処理オブジェクト型

リスト2に処理オブジェクト型のコーディング例を示す。これは、整数型の加算をする処理オブジェクトの例である。

リスト2. CalcAdd.cls

```
Public Function MainCalc(a As IOinteger, b As
IOinteger, c As IOinteger) As Integer
c.a = a.a + b.a
a.a = 0
b.a = 0
MainCalc = 1
End Function
```

A development method of programs using layered object flow
Takao Ito: Graduate School of Information, Teikyo Heisei
University
Fumitake Hirasima: SOFTECH Corp.
Isao Kurosaka: Graduate School of Science and Engineering,
Teikyo University
Masatake Nagai: School of Science and Engineering, Teikyo
University

4. クラスモジュールの自動作成

オブジェクトフロー型で作成されたグラフ図から、自動的にクラスモジュールを作成することができる。自動作成されたクラスモジュールのプログラムリスト例の一部をリスト3に示す。

リスト 3. 自動作成されたプログラムリスト

```
Public Sub Clock0
    Dim errorNum As Integer
    Dim box As Integer

    errorNum = CalcObj1(0).MainCalc(data1(CalcCheckA, 0),
        data1(CalcCheckA, 1), data1(CalcCheckB, 4))
    errorNum = CalcObj1(1).MainCalc(data1(CalcCheckA, 2),
        data1(CalcCheckA, 3), data1(CalcCheckB, 5))
    errorNum = CalcObj1(2).MainCalc(data1(CalcCheckA, 4),
        data1(CalcCheckA, 5), data1(CalcCheckB, 6))
    errorNum = CalcObj1(3).MainCalc(data1(CalcCheckA, 6),
        data1(CalcCheckA, 6), data1(CalcCheckB, 7))
    errorNum = CalcObj2(0).MainCalc(data2(CalcCheckA, 0),
        data2(CalcCheckA, 0), data2(CalcCheckB, 1))
    errorNum = CalcObj2(1).MainCalc(data2(CalcCheckA, 0),
        data2(CalcCheckA, 0), data2(CalcCheckB, 2))
    errorNum = CalcObj2(2).MainCalc(data2(CalcCheckA, 1),
        data2(CalcCheckA, 2), data2(CalcCheckB, 3))
    errorNum = CalcObj2(3).MainCalc(data2(CalcCheckA, 3),
        data2(CalcCheckA, 3), data2(CalcCheckB, 4))

    box = CalcCheckA
    CalcCheckA = CalcCheckB
    CalcCheckB = box
End Sub
```

このプログラムリストの基本構造を図 2 に示す。これは、左側に入力されているデータを演算ユニットにより計算させ、右側に書き込むことを示している。1 クロックごとに演算を行い、左右のメモリを逆転させる。それを繰り返すことにより処理を進めていく。

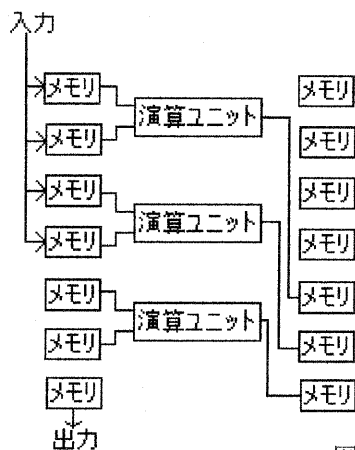


図 2. 基本構造

5. システム評価

従来のコーディングのみのプログラミングは、システム全体のデータの流れの直感的な把握には適していない。本提案システムでは、視覚的にみやすくすることで、ソフトウェア開発やデバッグをやりやすくしている。出力としてはネットリスト及び VB ソースリストで記述しているためセマンティックギャップ (図 3) の短縮化が可能となった。プログラム開発技法として、新しい部品 (オブジェクトフローモジュール) とを交換するだ

けで行うことが可能となった。

データの流れはフロー図作成によるマウス操作で行い、コーディング作業量は最小限にしている。データ型はクラスとして作成されるため、GUI をそのままデータ型として利用することが可能となり、VB のヒューマンマシンインタフェースを重視した可視化プログラミングとプログラムコードの自動作成化を実現している。

適用例として、SOUND プロセッサ用データフローエディタ『創流』^[3] によりデータフローグラフからハードウェアの再構成が可能となっているが、本提案システムでは直接、記述言語出力のサポートはされていないが、SOUND 用の演算ユニットを組み込むことにより SOUND プロセッサのシミュレートが可能であり、さらに仮想ユニットを増設することがすぐにできるため、SOUND プロセッサの開発に容易に寄与できる。

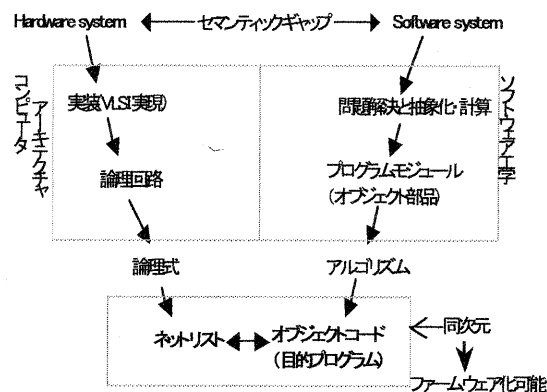


図 3. セマンティックギャップ

6. おわりに

本論文では、オブジェクトフロー型による流れ図作成から Visual Basic のクラスモジュールの自動作成について述べている。最小単位の部品をライブラリとして用意することで、ほとんどのプログラミング作業をマウス操作により行うことが可能となるだろう。

7. 参考文献

- [1] 平嶋史武, 島田肇, 伊藤貴雄, 永井正武, オブジェクトフロー型ソフトウェア開発方法への一提案, 情報処理学会 第 62 回全国大会, 8Q-7
- [2] 川口進, 白須宏俊, 奥田哲, 階層化データフロー図によるプログラムの設計と実動, 電子情報通信学会論文誌 D-I Vol. J82-D-I No. 10, 1999. 10
- [3] 佐々木浩志, 槻岡秀朗, 庄司修芳, 小林広明, 中村維男, 再構成可能な同期式データフロー計算機に関する一検討, 信学技報 VLD98-101, CPSY98-121, 1998 年