

# モジュール合成方式による ソフトウェア開発支援システムの構築

納 富 一 宏<sup>1</sup>・榎本 繁<sup>2</sup>・石井 博章<sup>1</sup>

<sup>1</sup> 情報工学科

<sup>2</sup> 工学部情報工学科 3 年

## A Construction of Software Development System with Method of Synthesizing Modules

Kazuhiro NOTOMI<sup>1)</sup>, Shigeru ENOMOTO<sup>2)</sup>, Hiroaki ISHII<sup>1)</sup>

### Abstract

In this paper, we introduce a construction of software development system with method of synthesizing modules. We are construct a new software development system that is deferent from used CASE system. This system has 3 features: 1) small scale, 2) for end-programmers, 3) for Window System. The method of application development is select modules from many kind of modules, which you need to synthesize selected modules. And we implemented this system on Windows NT ver 4.0. If you use this system you can development applications without much time. So, we have constructed this system.

**Key Words:** Module Bind, Event Driven, Software Development Environment, Software Science

### 1. は し が き

ソフトウェア工学における2大目標は、ソフトウェア開発における効率の向上、および製造されるソフトウェアの品質・信頼性の向上である。

これを目指したソフトウェア開発支援環境は、CASEシステムとして知られ、ソフトウェア製造工程のうち、主としてコーディングとドキュメンテーション、およびリビジョン管理を行う。

多くのものは、汎用的なソフトウェア開発に用いることのできるような形態をとる反面、実用レベルのソフトウェア開発においては、実務経験が豊富なシニアプログラマの持つスキルを凌駕するCASEシステムが存在するとは言えない。すなわち、CASEシステムがより知的な支援を提供することは困難であるというのが現状で

ある。

また、CASEシステムが実際のソフトウェア開発に用いられる機会が極端に少ないという事実は、要求仕様定義から外部仕様定義に至るソフトウェアの設計フェーズと、コーディング以降のフェーズとの間に無視できないギャップが存在し、設計の変更を直接反映させることが難しいことの例証であると考えられる。

近年のソフトウェア開発では、ウィンドウシステムをプラットフォームとすることが大前提となっており、コストの50%近くがインタフェース部分の開発に費やされている。また、開発言語や開発環境、あるいはプラットフォーム自体の加速度的な変化は、プログラマ教育自体を困難なものにする。

こうした現状に対処する一つのアプローチは、ソフト

ウェアの部品化・再利用問題として従来から研究開発が行われてきたが、常に開発支援の自由度と制約という障壁が存在してきた。

ここで、①小規模なソフトウェア作成支援、②ウィンドウシステムをターゲットとした実用レベルのプログラム開発、および③エンドユーザによるインタフェース開発、という3点に注目することが重要なヒントであると筆者らは考えている。

そこで、本稿では、ソフトウェアの部品化・再利用を行う手法として、C++言語をベースにしたモジュール合成方式を提案すると共に、簡単なインタフェースにより初心者にも操作可能なアプリケーション開発支援システムの構築について検討する。また、実際にモジュール合成方式によるソフトウェア開発支援システムを Windows NT 上にインプリメントし、その評価結果について考察する。

## 2. モジュール合成によるプログラム開発

### 2.1 モジュール部品とアプリケーションモデル

「ソフトウェア部品」とは、①単独で機能し（機能独立性）、②他に依存せず（無依存性）、③任意のアプリケーションと協調動作可能な（交換可能性）、ソフトウェアであると定義できる。本稿ではこの性質を有するプログラム、またはその集合を、モジュール部品、あるいは単にモジュール (module) と呼ぶ。

次に、任意のアプリケーション (application) とは、データの入出力を含めた「機能」の集合体であるとみなすことができる。ここで、「機能」とは、プログラムを構成する要素として表現されなければならず、しかも他と区別できる形態を有していなければならない。このことから、「機能」とはモジュール部品の抽象表現である、逆に言えば「機能」の具体表象として、モジュール部品を捉えることが可能である。

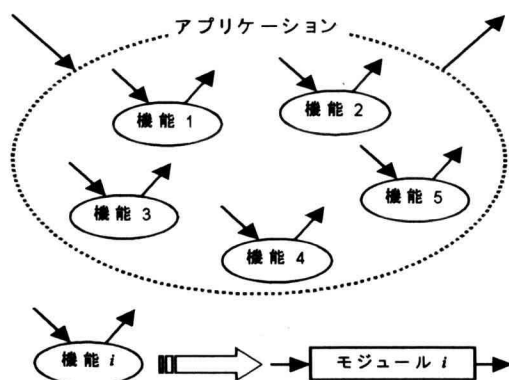


図 2.1 モジュールによるアプリケーションモデル

モジュール部品は、アプリケーションにおける機能部品であり、相互の連携は、入出力データの共有という「弱い依存関係」により保持されると仮定する。これは、個々のモジュールを一つのアプリケーションの中にとどめておくための制限と実際のプログラム表現からの制約である。

### 2.2 モジュール合成モデル

アプリケーションはモジュールから構成され、相互に弱い依存関係が存在する。この依存関係はデータ共有という制約によるものである。

アプリケーションを実際のプログラミングの次元で捉えた場合、先のモデルには、次の2つの要請が生じる点に注意しなければならない。すなわち、

- 1) 特定の機能群が相互に強い依存関係を有し、逐次処理的な動作が要求された場合、モジュール（「機能」）は、ダイクストラ (Dijkstra) の基本アルゴリズムである「接続」構造を表現できなければならない。
- 2) 特定の機能群が選択的、もしくは排他的依存関係を有し、並立動作が要求された場合、モジュール（「機能」）は、選択分岐構造を表現できなければならない。

よって、上記2つの要請に従い、モジュール合成という概念を先のモデルに取り入れる。ここで、1) を接続合成、2) を選択合成と呼ぶ。

モジュール接続合成とは、パイプ処理と同義である。すなわち、複数のモジュールを序列配置したとき、前置モジュールの出力が後置モジュールの入力となるような連結構造をとり、先頭モジュールへの入力、共有データ空間からなされ、同様に終端モジュールの出力が共有データ空間に送られるというものである。

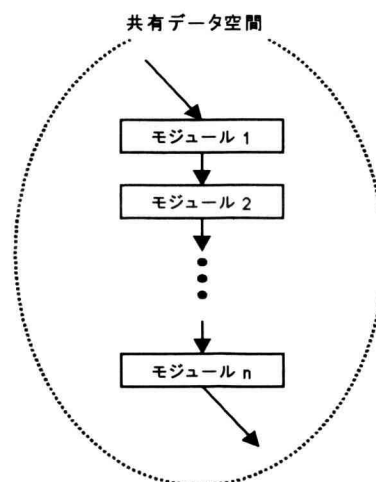


図 2.2 モジュール「接続合成」モデル

また、モジュール選択合成とは、制御システムやウィンドウシステムプログラムなどに見られるイベント駆動 (event driven) 型の構造を有する。すなわち、外部からのイベント (オブジェクト指向のメッセージに相当) をトリガとして特定のモジュールが応答起動するというものである。

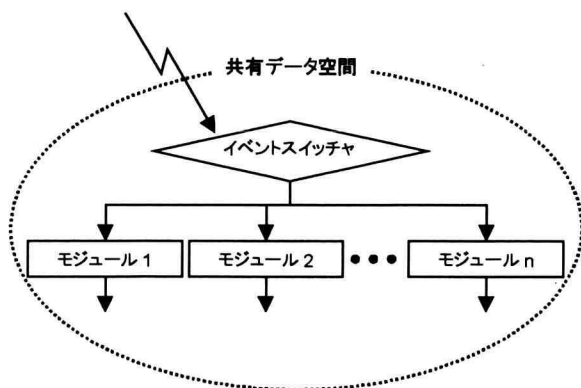


図 2.2 モジュール「選択合成」モデル

### 2.3 Windows アプリケーション開発への応用

先に提案したモデルを、現実のソフトウェア開発へ応用するために、現在の主流プラットフォームである Microsoft Windows の形態とそのプログラミングについて触れる。

Windows 上のアプリケーション開発では、画面設計とイベント応答という2点が重要視されている。従って、現在標準とされる開発環境は、いわゆる Visual 系言語を主体とするものである。Windows には、OS が提供する低水準 API (Application Programming Interface) と、言語系が提供する高水準ライブラリ (クラスライブラリ) 群があり、先の開発環境では、Windows のインタフェース部分に関する設計・開発支援を目的としている。

しかしながら、既存の開発支援システムは、イベント応答処理部分のプレースホルダ (place holder) を提供するのみであり、具体的な処理に関してのソフトウェア部品の提供をサポートしていない。すなわち、高水準ライブラリ群、およびシステムコールとしての API に含まれる多くのプログラムモジュールを利用するためには、プログラマによるテキストエディタを用いての旧来からの開発スタイルをそのまま踏襲することになる。

このことは、最初に述べたように、多くの CASE システム同様、プログラミングの自由度のみを優先し、ソフトウェアの部品化・再利用というガイドラインとしての制約を無視した結果であると言わざるを得ない。

そこで、Windows アプリケーションにおけるイベント応答のためのプレースホルダに対して、より具体的な「機能」をソフトウェア部品として割り当てることを考える。

Windows アプリケーションの機能の呼び出しは、すべてがイベントとして表現される。たとえば、マウスの移動やクリック、キーボードの押下、あるいはメニューの選択やダイアログ中のコントロールの操作などである。ここで、「機能」があらかじめ列挙可能なもの、基本的なもの、あるいは定型的なものとして表現できるならば、これをモジュールデータベース (module database) として一括管理することが可能である。

よって、Windows プログラムの開発とは、データベースで管理されたソフトウェア部品を、そのままプレースホルダに割り当てるか、あるいは接続合性、もしくは選択合成した後に、割り当てることに相当するものであると結論できる。

### 3. システム設計

この節では、モジュール合成システムを Windows 上にインプリメントするためのシステム設計について述べるとともに、その実装方法について検討する。

#### 3.1 システムの形態

複数のモジュールを必要な順番で呼び出すために今回作成したシステムはインタプリタ方式を採用した。そのシステムを Windows 上で動作させるためのシステム構成図を図 3.1 に示す。

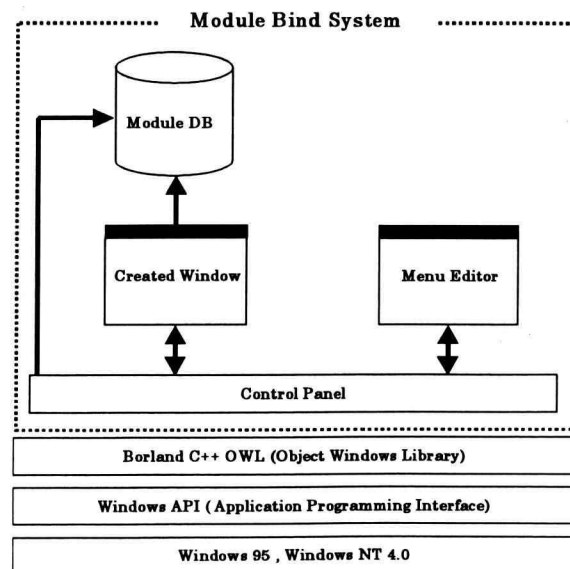


図 3.1 システム構成図

図 3.1 で示したモジュールバインドシステムは Windows NT 上にインプリメントされており、Windows API 及び OWL の上に存在している。

モジュールはこのシステムのベースルーチンである「コントロールパネル (control panel)」から選択する。前節で述べたように Windows アプリケーションはイベントが発生することによって処理が呼ばれる。ここで、コントロールパネルからモジュールを実行するとは、ユーザが選択したモジュール群を内部に含む Window を生成し、その Window に対するイベントが発生したときにイベントに応じたモジュールを実行することである。

### 3.2 モジュール部品の表現

2.1 で述べたソフトウェア部品の定義に基づいて、モジュールをデータベースとして一括管理するにはインタフェース部分を統一する必要がある。まず最初にプログラムへの実装を行うためのモジュールの C 言語による定義を行い、これを C 言語表現モデルとして提案する。

```

int    foo (void)
{
    .
    .
    .
}

```

図 3.2 モジュールの定義

図 3.2 の形を取る関数をプログラム上ではモジュールと呼ぶ。先に述べた 3 つのソフトウェア部品の条件を満たすために、モジュールの形式をただ一つに決めた。

図 3.2 のように引数を void とすることの意味は、複雑な引数を取る関数を使うことによって、モジュールのプログラムやそれを制御する構造の複雑化を防ぐためである。複数の複雑な引数を取ればより細かな制御が可能になり、自由度の高いアプリケーション開発が可能になるが、今までなされてきた CASE ツール研究で自由度が高い開発環境と知的な動的モジュール合成が可能な開発環境が両立でき得なかった事実により、今回の研究は、最も単純な構造での実装を試みることにする。

また、このモジュール定義では、戻り値として int 型

を返しているが、これはプログラム側でのモジュール動作チェックを行うため、モジュールの正常動作を確認するためである。

### 3.3 部品合成

部品合成手段として本稿で作成したプログラムでは関数ポインタ配列を使用した。関数ポインタ配列を使用する理由として、先に述べたように合成には接続性と選択合成があり、その 2 つを同時に満たすことが可能であるからである。以下に関数ポインタの具体的な実装方法とその意味について述べる。

作成したアプリケーションに対するイベントが発生したとき、応答関数が呼び出したいモジュール名の配列を作成し、イベントの内容によって、動的に配列の順番を入れ替えたものを、モジュールデータベースクラス (module database class) 内に存在するモジュール実行関数に渡す。すなわち、応答関数は

- 1) 呼び出すモジュールの選択が可能
- 2) 呼び出すモジュールの内容および順序の動的な入れ替えが可能

といえる。この 2 つはそれぞれ選択合成、接続合成に相当し、先に述べた 2 つの合成方法を C 言語として表現可能となる。

以上のことからモジュールの合成は関数ポインタ配列を使用することにより実現可能である。

### 3.4 動的実行

モジュール実行関数の引数として受け取った配列をモジュールポインタ配列として格納し、順番どおりモジュールポインタを実行して行くことにより配列に格納された任意の組み合わせのモジュールは動的に実行される。ゆえに、この方法でモジュール合成を行うことにより、

- 1) コンパイル不要
- 2) 動的に呼び出し可能
- 3) アプリケーションの安全性向上

が同時に満たせるといえる。

## 4. システムの評価

### 4.1 コントロールパネル

必要なモジュールを選択する方法として、より単純に、

より短時間で行うために以下のような選択形式を取ることによって、その実現を試みた。

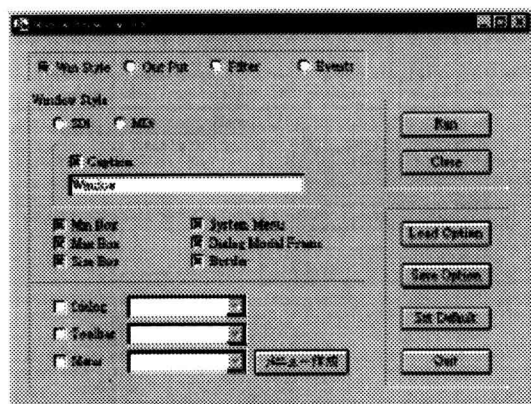


図 4.1 コントロールパネル

図 4.1 はモジュールを選択するコントロールパネルである。このコントロールパネルからモジュールを選択し、アプリケーションの開発を行う。モジュールの選択方法はきわめて単純になっており、以下の3通りの方法で行える。

- 1) 重複して選択可能なモジュールに関してはチェックボックスになっているので、必要なモジュールにチェックを入れる。
- 2) 5つ以上ある複数のモジュールの内、一つだけ選択可能な場合は、コンボボックスにモジュール名が格納されているので必要なモジュールをコンボボックスの中から選択する。
- 3) 2) の条件でモジュールの数が5つ未満のものはラジオボタンになっているのでラジオボタンにチェックを入れる。

## 4.2 メニューエディタ

イベントが発生する要素の一つとして、メニュー選択が挙げられる。

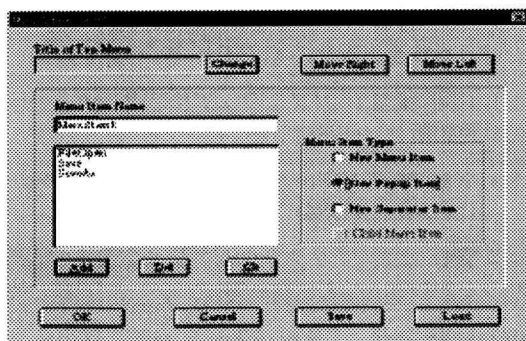


図 4.2 メニューエディタ

今回のシステムではユーザがメニューを自由に設計し、そのメニューアイテム一つ一つについて、コントロールパネルからモジュールを割り当てられるものにした。

図 4.2 はユーザがメニューを作成するときのメニューエディタである。

## 4.3 モジュールを含んだ Window の生成

3.1 節で述べたように、モジュールをユーザが選択しても Window システム上ではすべてをその場で実行するわけではない。ここではモジュールを内部に含んだ Window を例に挙げる。

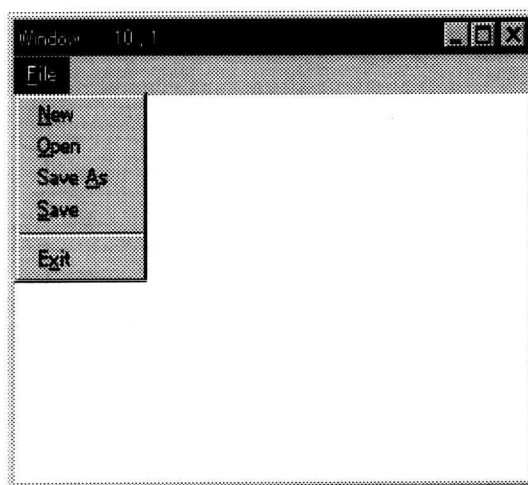


図 4.3 作成したアプリケーション

図 4.3 は図 4.2 で示すメニューエディタで作成したメニューに図 4.1 で示すコントロールパネルで以下のモジュールを選択し、合成して生成された Window である。以下に内部に含まれているモジュールの種類を挙げる。

- 作成したメニューのメニュー項目「Open」にフィルター処理実行モジュールを割り付けた
- キャプション付き Window
- キャプションの内容は Window
- Maximize Box 付き Window
- Minimize Box 付き Window
- Sizechange Box 付き Window
- マウスの座標をキャプションに表示する
- 出力結果はクライアント領域に表示する
- 出力結果は answer.txt に出力する

以下は Window 内部に含ませたモジュールを接続合成したものである。すなわち、実行時にフィルタとして動作するもので、順番はモジュール登録順になっている。

- 1) ファイルを読み込む
- 2) スペースで区切られた単語を抽出する
- 3) 単語群にソートをかける
- 4) ファイルに書き込む

#### 4.4 実行結果

データベースに登録したモジュールの内、いくつかはフィルタとして動作するものを登録した。フィルタの場合、目に見える結果が出る場合が多く、その出力をいくつかの方法で視覚的に見る方法を考え、出力方法の一つとして生成した Window 上に表示するものを作成した。



図 4.4 フィルター実行結果

図 4.4 は図 4.3 の生成 Window のメニュー項目「Open」を選択し、イベント駆動によりモジュールが動作した結果である。4.3 節で出力結果をクライアント領域に出力するモジュールを組み込んでいるため、このような出力結果になる。

#### 4.5 アプリケーションへのダイアログの貼り付け

選択したモジュールを含む Window にあらかじめ作られている大きな部品を組み込む例として作成済みのダイアログボックスを表示する例を以下に挙げる。

Windows アプリケーションにおいて API が提供するインタフェース以外のものを開発することは困難である。ゆえに開発するソフトウェアのインタフェース部分はある程度の限定が可能である。インタフェースの限定ができるのであれば、ある程度の機能の集合モジュールを用意することでより緻密なアプリケーション開発が可能に

なると考えた。

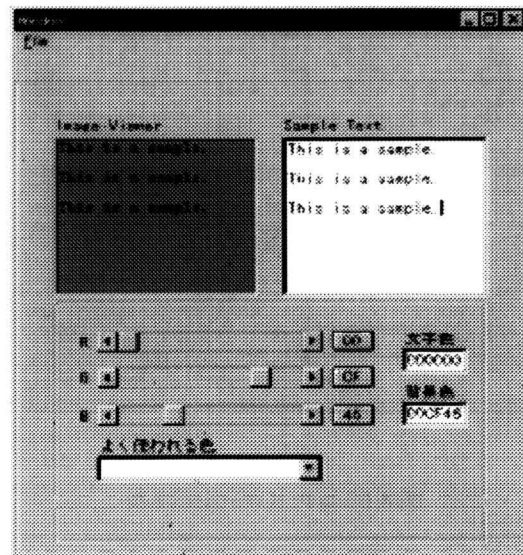


図 4.5 ダイアログを貼り付けた例

図 4.5 は生成 Window にダイアログを貼り付けた例である。張り付けているダイアログボックス上はそのダイアログボックス独自の処理を行い、メニューなどはコントロールパネルから選択したものが動作する。

#### 5. むすび

今回の研究で作成したシステムにより、小規模なソフトウェア開発支援システムながら Windows System 上で動作する実用レベルのプログラム開発が可能になった。今後の目標として、ユーザが作成したモジュールをシステムに登録可能にすることである。モジュールの登録が可能であれば、自由度の制約がなく、知的な動的部品合成が可能な開発環境を提供することができるといえよう。

#### 参考文献

- 1) 片岡 雅憲：『ソフトウェアの開発パラダイム』，日科技連出版社，(1992)。
- 2) 片岡 雅憲：『ソフトウェアモデリング』，日科技連出版社，(1988)
- 3) 有沢 誠：『ソフトウェアプロトタイピング』，近代科学社，(1986)
- 4) 菅野 文友：『ソフトウェア製品生産工学入門』，日科技連出版社，(1992)
- 5) 水野 幸男：『最新ソフトウェアプロダクト・エンジニアリング』，ジャテック社，(1979)