ソフトウェア工学 99−12
（1994．7．8）

データフローによる部品探索機能を持つ仕様エディタの試作

臼井義美

日本電子計算株式会社

我々はビジネスアプリケーションの開発のために有効なオブジェクト指向と構造化技法を組み合わせた設計支援ツール「OBJECT-FLOW」を試作した。このシステムは、宣言的要素を記述するデータ構造と手続き的要素を記述する処理ユニットの２つのオブジェクトを使って、プロセスフロー上にオブジェクトの実行順序を記述できるようにしたものである。本システムでは両者の関係をデータフローを介して検証し、矛盾点を指摘している。設計者が処理内容の記述中に、リポジトリに蓄えた仕様部品からデータフローを利用して必要なプロセスユニットを抽出し、仕様の作成の支援を受けることができる。

Trial Manufacture of Specification Editor Having Part-searching Function by Data Flow

Yoshimi usui

Japan Information Processing service Co.,LTD.

1-9-1 edobori Nishi-ku Osaka 550 Japan

We Manufactured for trial the design-aiding tool "OBJECT-FLOW" in which Object -Oriented Design Technique and Structured-Design Technique that are effective for the development of business application are combined.We enable to describe the order of execution of the objects on Process-Flow, using two kinds of objects, namely the Data-Structure object describing declaratory elements and the Processing-Unit object describing procedural elements.

During the description of the contents of processing, desinger can extract the required Process-Unit by utilizing the Data-Flow from the specification parts stored in a repository, and receive the aid or drawing up spacificatons.

1．はじめに

最近、オブジェクト指向によるシステム分析・設計手法が各種提案されているが、ビジネス・アプリケーションの開発にはまだ十分活用されているとは言えない。現在でもビジネス・アプリケーションの開発は、従来の構造化手法による設計方法が広く受け入れられているが、それは経験の蓄積による成熟した手法であるという理由だけでなく、あらかじめ決められた手順に沿って実行したいシステムの設計に便利であることも大きな要因の１つであると考えられる。

一般にオブジェクト指向設計の利点として、

・現実世界とシステム内部の写像関係が明確である。
・オブジェクトの独立性が高く再利用しやすい。
・オブジェクトは逐次的にも並列的にも実行できる。

などがあげられるが、他方

・何をオブジェクトとして定義するのかの判定が難しい。
・実際にオブジェクトが実行される順序を鳥瞰的に認識するのが困難である。
・複数のオブジェクトが集団としてある機能を果たすような場合の記述が難しい。

というような問題点がある。

我々は、OBJECT-FLOWと呼ぶ独自のダイアグラムを用いた仕様エディタを試作し、オブジェクト指向の利点を活かしつつ上記の問題点を解決しようと考えている。また、アプリケーション設計における支援の１つとして、既存の仕様部品から適切な部品を抽出する機能を開発している。

本稿では、OBJECT-FLOWにおける部品の考え方とその扱い方について述べ、それらを結びつけるデータフローを用いて仕様部品を探索する方法について考察する。

2．OBJECT-FLOWの概要

2．1　OBJECT-FLOWの考え方

一般に、オブジェクト指向による設計において、オブジェクト間の関係は

・クラス間の概念的な包括関係を示すis-a
・オブジェクト間の構造的な集約関係を表すhas-a
・オブジェクト間の参照、利用関係を表す関連

に分類される。

OBJECT-FLOWでは、これらの関係を次のような手段で扱うことにした。まず、ビジネスで用いられるデータを現す要素をオブジェクトとして捉え、リポジトリにis-aのクラス階層に体系化して蓄積する。次に、実際のシステム設計で利用するデータ項目を個々のデータ要素を組合せて定義したオブジェクトとし、has-aの階層構造を構築する。そして、各データ項目間の関係を、宣言的に扱える部分を関係ユニットとして、またメソッドに当たる手続き的な関係を処理ユニットとして、それぞれオブジェクトの形式で表すこととした。このように、OBJECT-FLOWではデータ要素、データ項目、処理ユニット等をいずれもオブジェクトとして取り扱っている。

仕様の記述にあたっては、処理ユニットを処理の手順を示すプロセスフロー上に配置することによって、自動的にデータフローとデータ項目の状態遷移を表示するようにした。

従って、OBJECT-FLOWの基本的なイメージは、図１に示すようにデータ構造と、その時間的な変化を表すデータフロー、実行の順序に沿って配置された処理ユニットから構成されるもので、我々は「水車モデル」と呼んでいる。

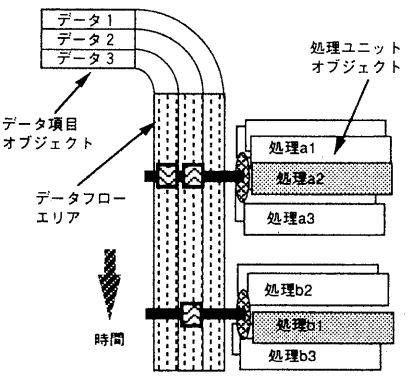

図１　水車モデル

2．2　データ部品の考え方

（１）データ要素クラスの定義

本システムでは、ビジネス・アプリケーションにおける問題解決の主要な対象物であるデータ項目について、その構成要素をデータ要素というオブジェクトとして扱い、物理的クラス階層と論理的クラス階層に分離してis-aの形式で体系化している。物理的クラスは、データ要素をデータの物理的な属性をもとに体系化したもの

で、通常システム提供者が事前に登録する。一方、論理的クラスは、対象となる業務分野に特有のデータ要素を体系化するもので、主にアプリケーションの設計者が登録することとしている。例えば、ファイルや帳票などを物理的クラスとし、預金口座や残高などを論理的クラスとして扱っている。

（2）データ項目の定義

利用者が実際のアプリケーションの設計にあたって用いるデータ項目は、データ要素の論理的クラスの名称と物理的クラスの名称を組み合わせたデータ項目名を作成することにより確定する。このとき、作成されるデータ項目は、双方のクラスの属性を多重継承している。

例えば、「銀行預金口座ファイル」というデータ項目は「銀行預金口座」＋「ファイル」と言う形で作成され、図2に示すように論理的クラス階層に含まれる業務処理用語と、物理的クラス階層に属するシステム処理用語で表され、双方のメソッドが利用できる。

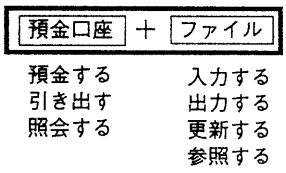

図2　データ項目

（3）データ構造の作成

ビジネスにおけるデータ項目は、現実にはいくつかのデータ項目の集合として用いられることが多い。従って、それらの集合データ項目を作成するために、データ構造エディタを用いてデータ構造図を作成する。データ構造を構成するデータ項目は、内部変数にデータ項目というオブジェクトを持つ複合オブジェクトである。アプリケーションの設計者は、開発すべきシステムで使用するデータ項目をクラス階層から取り出し、データ構造図を作成することによってhas-a形式の集約関係を定義する。

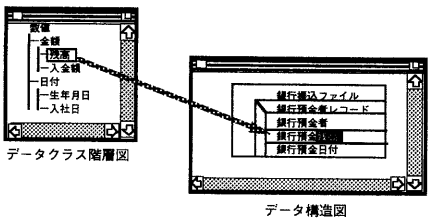

図3　集合データ項目の作成

2．3　処理ユニットについて

本システムでは、各データ項目のメソッドにあたる処理ユニットをオブジェクトとして扱い、それには基本ユニットとその集合である部品ユニット、複数のデータ項目の宣言的な関係を示す関係ユニットの3種類を用いている。

（1）基本ユニット

基本ユニットは、あるデータ項目に定義されたメソッドに対応するもので、日本語仕様を解釈して実行できるようなオブジェクトである。基本ユニットを作成するには、そこで取り扱うデータ項目を指定し、そのデータ項目に対応した構文メニューによって処理の内容を記述する。

図4に示すダイアグラムは、仕様を表示した処理ユニット群が、プロセスフローに沿って処理の手順どおりに配置された様子である。

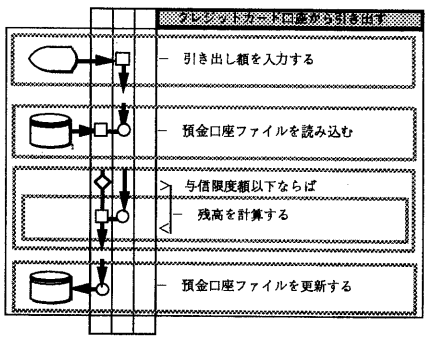

図4　基本ユニットの配列

（2）部品ユニット

複数の基本ユニットが集団で1つの機能を表すとき、利用者は部品ユニットとしてリポジトリに登録することができる。部品ユニットは、図5に示すように関連するデータ項目の処理メニューより選択でき、当該部品ユニットの内部仕様をプロセス・エディタ内に表示できる。

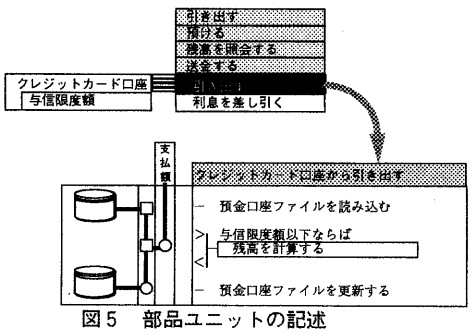

図5　部品ユニットの記述

プロセス・エディタで設計中に、ある部品ユニットの内部を展開すると、図6のように基本ユニットで記述された日本語の仕様書として表示される。

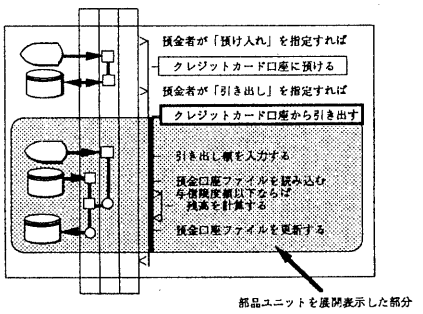

図6　部品ユニットの詳細化

データ項目には集合データ項目と基本データ項目があり、それぞれに関連する処理ユニットがリンクされているので、通常集合データ項目に関連する部品ユニットは、その集合データ項目に含まれる基本データ項目に関する基本ユニットの組合せで表されることが多い。従って、集合データ項目内ではそれらを構成する基本ユニット同士でメッセージ交換が行われ、それらを集約したものが部品ユニットとなる。

その様子を、図7に示す。

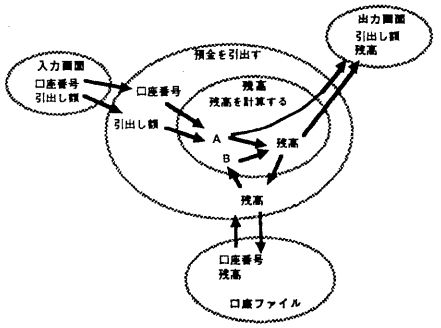

図7　処理ユニット間のデータフロー

この図の「預金を引き出す」という処理ユニットは、他のデータ項目「入力画面」から「口座番号」と「引き出し額」をメッセージとして受信し、「預金を引き出す」という処理を実行する。データ項目「口座ファイル」は、この部品ユニットに含まれる処理ユニットが関連するデータ項目である。また、「出力画面」オブジェクトに対して処理結果としての「残高」などをメッセージとして送信する。

さて、「預金を引き出す」という部品ユニットは、内部に部品ユニットや基本ユニットを含んでおり、これらの間でもメッセージ交換が行われる。これは、1つの部品ユニット内で行われる内部メッセージ交信であり、他のオブジェクトとのメッセージ交信は必ず包含するオブジェクトを通して行われる。

OBJECT-FLOWは、このような集団データ項目の振る舞いの記述についての外部処理ユニットとのやりとり、および包含される基本ユニット間のやりとりなどを、図8に示すようなダイアグラムを用いることで視覚的に表現できる。

（3）関係ユニット

事前にどのタイミングで実行するかを指定しないで、ある条件が満たされると実行されるような処理ユニットや、データ項目間の宣言的な関係を表す処理ユニットを、関係ユニットとしてリポジトリに保持している。

2．4　データフロー

（1）データフローエリア

データ項目は、ダイアグラムの一部に時間軸に沿ったデータフローエリアをビューとして持っている。実際のシステムでは、データ構造の最上位に位置するデータ項目が代表して内部のデータフローを表示している。このデータフローエリアは、データ項目間で交信されるメッセージのタイミングチャートであり、それ自体は各々のデータ項目の時間的な推移である状態遷移を示している。

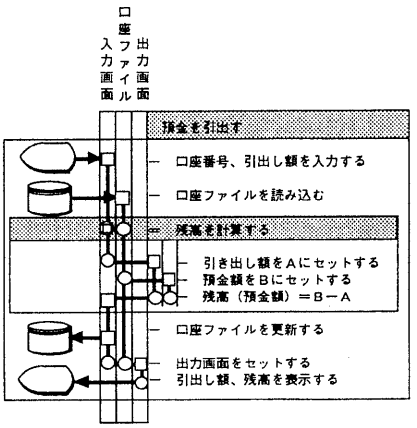

図8　OBJECT-FLOWのデータフロー表示

（2）垂直フローと水平フロー

データ構造を構成する各データ項目の視点か

ら見ると、データフローエリアに表示する縦のデータフローすなわち垂直フローは、各々のデータ項目の状態遷移を示す。従って、データフローが描かれていない部分はオブジェクトの未生成区間もしくは値の参照が不能である区間を示し、データフローの描かれている部分は、有効な値が保持されている区間を示している。

また、横方向のデータフローすなわち水平フローは、集合データ項目に包含されたデータ項目同士の内部データフローと、他のデータ項目との間のメッセージ交信を表す外部データフローがある。

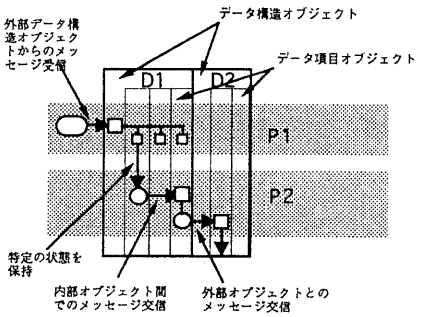

図9　データ項目からみたデータフロー

一方、処理ユニットの視点からデータフローを捉えると、各処理ユニット間のメッセージ交換は垂直フローで表される。この場合も、ある部品ユニットに包含される基本ユニット同士のメッセージ交換を示す内部データフローと、他の処理ユニットとの交信を示す外部データフローが存在する。

また、水平フローは、処理ユニットの内部変数の間で行われるデータ変換などの加工工程を表している。

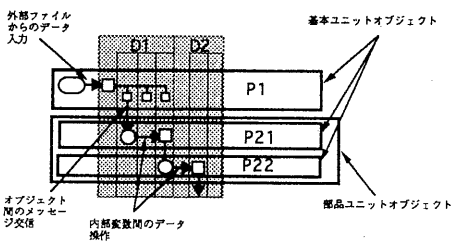

図10　処理ユニットからみたデータフロー

2．5　プロセスフロー

（1）逐次処理

処理ユニット間でメッセージを手続き的な呼び出しとして実現した場合は、オブジェクト間の交信は同期的なものとなり、処理ユニットがプロセスフロー上に一列に並ぶことになる。なお、処理ユニットを配置するためのプロセスフローは、構造化設計における構造化チャートの一種であり、順次、反復、条件の3基本構造を組み合わせて描かれる。

（2）並列処理

一方、処理ユニットを並列プロセスにおけるエントリー呼び出しの形で実現する場合には、交信は非同期的なものとなり、プロセスフロー上に並列の実行数に応じたフローができる。OBJECT-FLOWでは、図11のように並列処理が発生するタイミングで複数のリーフを描き、エディタ上でページ表示を行うことにより、逐次処理と並列処理が混在する仕様も容易に表現することができる。

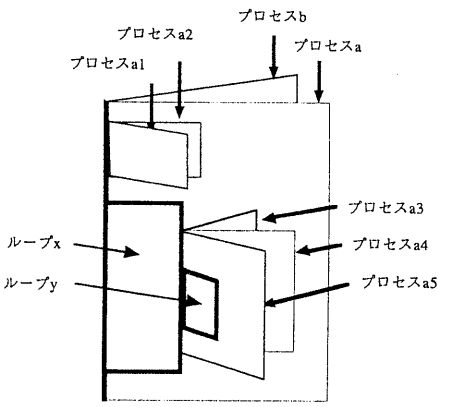

図11　並列処理の表現イメージ

（3）随時処理

このような手続き的なプロセスだけでなく、データ項目に付随させて、データ項目間の関係を表すプロセスや、プレ条件を満たした時に実行されるプロセスなどの関係ユニットを、宣言的に記述することも可能である。

3．部品探索処理

3．1　処理ユニットのデータフロー端子

前述のように、データフローを処理ユニット間のメッセージ交信と捉え、各処理ユニットのデータ項目を結合用の端子とみなすと、OB-

JECT-FLOWにおける垂直フローはそれらの端子をデータフローで結びつけることである。

処理ユニットにおけるデータフローの接続用端子には、

・他の処理ユニットとの外部データ交信用端子
・処理ユニットに含まれる外部ファイルなどとのデータ入出力用端子
・処理ユニットの内部データ交信用端子

が考えられ、図12に示す6種類の端子から成るデータフローの組合せで表現することができる。

なお、複数の基本ユニットから構成される部品ユニットは、包含される基本ユニットのデータフロー端子を集約したもので、部品ユニットと外部の処理ユニット間のデータフロー端子は基本ユニットのそれの論理積となる。

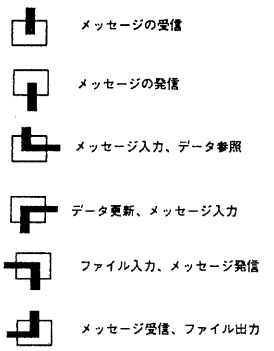

図12　データフロー端子

3．2　データフローによる整合性検証

OBJCT-FLOWでは、各処理ユニットのデータフロー端子を結んで得られるデータフローをもとに次のような整合性の検証を行っている。

（1）水平フローの検証

水平フローについては、処理ユニットの構文を記述するとき、接続の対象となるデータ項目同士の属性を次のような手続きで検証する。

・使用したいデータ項目の含まれるデータフローエリアを指定し、対応するデータ構造図を表示させる。
・目的のデータ項目を指定すると、データ項目の属性を参照して使用可能な構文テンプレートが表示される
・テンプレート中に埋め込むべきデータ項目をデータ構造図から指定する。このとき、データ項目の属性と構文中の仕様で認められた属性を考慮して、使用可能なデータ項目以外は選択できないようにプロテクトがかけられる。

（2）垂直フローの検証

一方、垂直フローの検証は、処理ユニット間のメッセージ交信の整合性を検査することによって行っている。ここで用いるプロセス・ダイアグラム上では、処理ユニット間で交信可能なデータ項目は、同一データフローエリア上に表示され、データの流れる道筋だけでなく、時間軸に沿ったデータの状態遷移を読みとることができる。

そこで、本システムでは図13に示すように、各データ項目の状態を4値に設定し、各処理ユニットにおけるデータ項目のアクセスタイプとして7種類を定め、状態の遷移ルールを設定した。そうして、各処理ユニットでアクセスするデータ項目の状態によって処理の妥当性をチェックし、矛盾があればデータフローのアクセス記号を着色することよって、その旨を利用者に知らせている。

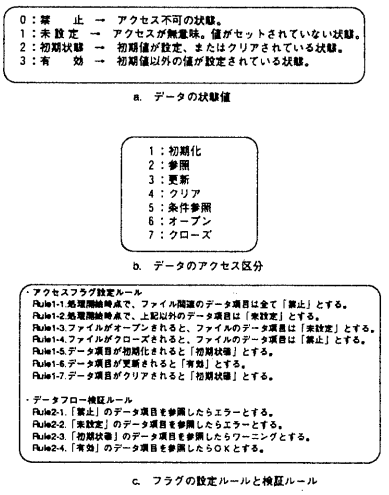

図13　状態の検査ルール

3．3　部品探索の考え方

設計中の利用者にとって、仕様の記述中に有効と思われる処理ユニットが自動的に検索され、適当な位置に表示されると便利である。我々は、仕様の未完成な部分に対して、リポジトリに登録された部品ユニットから、データフローを追跡して再利用可能な処理ユニットを抽出することにした。ただし、この機能を利用する場合には、少なくとも探索すべきデータ項目が特定できるような処理ユニットが予め記入されていなければならない。

部品探索は、利用者が指定した区間に対し

て、必要なデータ項目の送信元が確定していない処理ユニットに対して、必要とする処理ユニットの候補を次の３つの方法で生成もしくは抽出しようというものである。

（1）基本ユニットの生成

未確定データ項目が存在する処理ユニットについて、当該データ項目と同じ名称の他のデータ項目で、同じタイミングで参照可能な状態のものが存在すれば、そのデータ項目を未確定データ項目に転送するような基本ユニットを生成する。

（2）関係ユニットの抽出

リポジトリに登録された関係ユニットは、複数のデータ項目間で常に成立する関係を記述したものであるが、未確定データ項目を出力している関係ユニットが存在すれば抽出する。

（3）部品ユニットの抽出

リポジトリに登録された部品ユニットを探索し、未確定データ項目を出力とする部品ユニットを見つけだす。

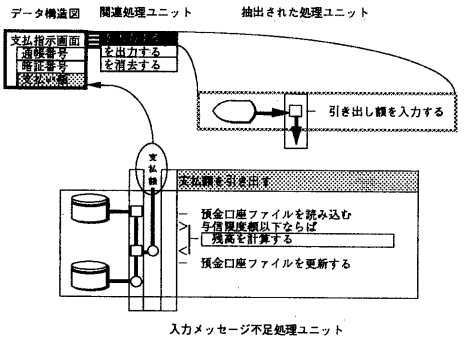

図14　処理ユニットの探索

図14の例で、「支払額を引き出す」という処理ユニットの「支払額」が未確定データであるとする。システムは、データ構造図内の「支払額」に関連する処理ユニットを探し、なければそのデータ項目を含む集合データ項目「支払い指示画面」に関する処理ユニット「支払い指示画面を入力する」を探し出す。この処理ユニットの出力データ項目に「支払額」が含まれているので、候補として採用する。

このような部品ユニットの探索を繰り返すことによって、データフローの検証を利用して最も整合性の高い組合せを見つけだし、指定されたプロセスフロー上に仮配置する。利用者はこれらの仮配置された部品ユニットを参考にしながらシステムを確定していくことになる。

３．４　プレフィックスとインスタンス

本システムではデータ要素とそこに含まれるクラスメソッドをis-aのクラス階層としてリポジトリに登録しており、設計時にこれらを利用する場合には、データ項目の名称の前にプリフィックスを付加してインスタンスを表すことにしている。

データ項目を仕様に取り込むに当たっては、has-a形式の集合データ項目を定義するが、その時点でデータ項目が所属する集団、すなわちマスターやトランザクション、データベースという物理的な属性の識別を行うため、データ項目にプレフィックスを付加することによってインスタンスを表している。

一方、処理ユニットもクラスメソッドとしてリポジトリに登録されている間は、プレフィックスのない抽象化したデータ名称を用いている。そして、実際の設計に当たってプロセス・ダイアグラムに書き込まれた時点で、プレフィックスを付加することによってインスタンス・メソッドとなる。

３．５　探索方法

本システムにおける部品ユニットの探索方法をまとめると次のようである。

- 指定された部品ユニットのうち、結合先の未確定なデータ項目を見つけだす。
- 未確定データ項目に対して、プレフィックスを除いたデータ項目名で参照可能なデータ項目が存在すれば、そのデータ項目を転送するような基本ユニットを生成する。
- ここで、転送の対象となるデータ項目の的中率向上のため、対象処理ユニットからできるだけ近い位置で出力されるデータ項目を優先するなどのルールを適用する。
- 転送で得られないデータ項目については、求めるデータ項目を出力している関係ユニットをリポジトリから探し出す。
- 適切な関係ユニットが存在しない場合は、求めるデータ項目を出力している部品ユニットをリポジトリから探し出す。
- 複数の部品ユニットの候補があれば、対象となる処理ユニットの位置におけるデータ

フローから判断して最も適合率の高い処理ユニットを選択する。

・選択した処理ユニットを、データフローの検証ルールを適用して、最も確定するデータ項目が多く未確定データ項目が少なくなる位置を算出する。

・算出結果をもとに、抽出した処理ユニットを最適な位置に仮配置し、利用者の判断によって確定する。

3．6　処理ユニットの最適配置

探索した処理ユニットを挿入する位置を算出するに当たって、いくつかのルールが必要である。例えば、ある条件に含まれる処理ユニット群が存在するときは、その条件を満たした時と満たさなかった時の双方についてデータフローの検証を行っている。また、ループに含まれる処理ユニット群では、前回のループ時点で設定されたデータ項目の値を利用するときは時間軸に沿って描くデータフローが一意に定まらないことがあるため、初期ループと２回目以降のループを組み合わせて検証している。

このように最適な処理ユニットの配置については、その精度を上げるためにルールの拡張やいくつかの解析方法についてテストを行っている。

4．おわりに

OBJECT-FLOWは、従来構造化設計のために開発した構造化エディタ「Show-CASE」を発展させて、オブジェクト指向の要素を取り込んだものである。現在のプロトタイプでは、処理ユニットは表示した日本語の仕様をCOBOL言語またはオブジェクト指向COBOLに変換する機能を持たせているが、本来の処理ユニットは自らの仕様を解釈して直接実行すべきである。図15にOBJECT-FLOWのシステム構成を示す。

本稿では、データ項目をオブジェクトとして定義すると共に関連するメソッドを処理ユニットと呼ぶオブジェクトとして扱うことにより、ビジネス・アプリケーションの設計に便利な仕様エディタについて紹介した。また、仕様部品の再利用の方法とデータフローを用いた各種の検証についての考え方を説明し、データフローの追跡によって利用可能な処理ユニットの候補を探索する方法について考察した。

今後の課題として、効果的な探索方法や的中率の高いルールを実験し評価する必要がある。また、構造化設計とオブジェクト指向設計の最も効果的な融合の方法を探っていきたい。

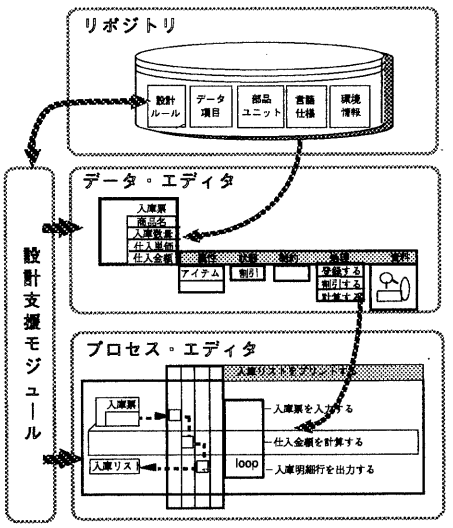

図15　システム構成図

参考文献

［1］Kurt J.Schmucker:"Object-Oriented Programming for the Macintosh"Hyden Book.Co.(1986)

［2］Ian他著，佐野美知男他訳：ソフトウェアエンジニアリング」フジテクノシステム，(1993)

［3］Grady Booch:Object Oriented Disign,The Benjamin/Cummings Pub.Co.(1991)

［4］大槻泰則，榎本進：アクタ・モデルを介したデータフロー・モデルによるオブジェクト指向の実現，情報処理学会論文誌，Vol.35,No.5,pp.875-886(1994)

［5］本位田真一，山城明宏：オブジェクト指向分析・設計，情報処理，Vol.35,No.5,pp.392-401(1994)

［6］臼井義美：Show-CASEにおける仕様部品の再利用支援環境，情報処理学会ソフトウェア再利用技術シンポジウム論文集，(1992)

［7］NOAH S.PRYWS,AMIR PNUELI: Compilation of Nonprocedural Specifications into Computer Programs,IEEE Trans.Softw.Eng.,Vol.SE-9,No.3,pp.267-279(1983)