計算機アーキテクチャ 83－3
（１９９０．７．１８）

# 超高位図的仕様記述環境（ＡＥＳＯＰ）の プロトタイプの開発

西川洋一郎, 原秀次 (三洋電機 (株)), 稲岡美恵, 山崎哲男, 鳴憲司 (三菱電機 (株)),
芳田真一 (シャープ (株)), 日根俊治 (松下電器産業 (株)), 西川博昭, 寺田浩詔 (大阪大学)

超高位図的仕様記述環境 (Advanced Environment for SOftware Production;AESOP) のプロトタイプは、多様性、了解性に富む図的表現形式を用いた多面的な仕様記述環境、および仕様記述過程と一体化したプロトタイピング環境の実現によって、効率的なソフトウェア開発環境の確立を目指している。開発にあたっては、将来への発展性、および開発の容易性を考慮してファイル中心の設計を行った。本報告では、プロトタイプの機能構成、およびプロトタイプが提供する仕様記述環境について述べ、プロトタイプを使用して得た評価結果をまとめる。

# A prototype of the Diagrammatical Specification Environment AESOP

Youichiro NISHIKAWA,Shuji HARA
(SANYO Electric Co.,Ltd.,1-18-13 Hashiridani Hirakata Osaka 573 Japan)
Yoshie INAOKA,Tetsuo YAMASAKI,Kenji SHIMA
(Mitsubishi Electric Corporation)
Shin-ichi YOSHIDA (Sharp Corporation)
Shunji HINE (Matsushita Electric Industrial Co.,Ltd.)
Hiroaki NISHIKAWA,Hiroaki TERADA (Osaka University)

AESOP is the Advanced Environment for SOftware Production. A programmer can produce the computer system softwares easily with coordinated diagrammatical specifications, and he can also tests the specifications interactively on the AESOP environment. In this report,we will describe functional/software structure of the AESOP prototype that is designed to raise the system tunability and portability, and will estimate the effect for software production on the AESOP environment.

## 1　はじめに

ソフトウェアの設計、開発においては、利用者の要求をシステムに対して確実に伝達でき、かつ、システムに伝達された内容を効果的に確認できる手法の確立が重要である。超高位図的仕様記述環境（Advanced Environment for SOftware Production;AESOP）の研究では、ソフトウェアの記述性、了解性を向上させる手法、および効果的なプロトタイピングを実現するための処理モデルの確立を通じて、上記要求の実現を目指している。

AESOP プロトタイプは、研究の構想段階で明らかにした多面的な図的手法を用いた仕様記述、種々の表記法相互間の変換、仕様記述レベルでのプロトタイピング環境等の着想[1]を具現化して評価することを目的に開発した。

## 2　機能構成

プロトタイプの設計にあたっては、将来への発展性、および開発の容易性を考慮した。即ち、図1に示すようにプロトタイプで取り扱う内部情報を仕様記述水準、論理的なマシンに依存する水準、実マシンに依存する水準で分割し、それぞれの水準における情報の変換を独立の処理系が実現することによって、各水準におけるモデルの変更を局所的な処理系の変更で吸収できる移植性の高い構成を採用した。さらに、これらの情報を一つの統合的なファイルとして扱い、このファイルに対する直接的な操作を1つの処理系（統合ファイル管理系；UF）に集約し、他の処理系からのアクセスはUFが提供するデータ構造操作命令を介して実現する方式とした。この結果、各情報のデータ構造を具体的に決定すれば、それぞれの処理系の開発は並列に進めることが可能となり、併せて、各処理系間のインタフェースが明確なため、個々の処理系の独立性を高めることができる。また、物理的なファイル環境の変更やファイル処理方式の高機能化にも容易に対応できる。

図2にプロトタイプの論理機能構成を示す。入出力系（IO）は、エディタ群（EDs）を介して仕様記述環境におけるユーザインタフェース機能を提供するとともに、EDsと他の処理系との間の通信インタフェースを管理する。変換系（CV）は、複数の表現形式による仕様から抽出される個々の仕様記述情報を統合し、論理水準における処理モデルに対応したコンストラクト情報へ変換するとともに、個々の仕様記述情報間の相互変換機能を提供する。実行系（EX）は、コンストラクト情報をさらに、物理水準における実行モデルに対応した実行形式情報に変換するとともに、それを実行することによって、部分的に完結した仕様の検証を支援する。部品管理系（CP）は、完結した仕様の部品化、再利用を実現するための環境を管理する。ワークステーション上での実現にあたっては、処理速度の問題、プロセス間通信数に対する制限等から、処理系本体部分はIOを中心としてCV、EX、CP、UFが関

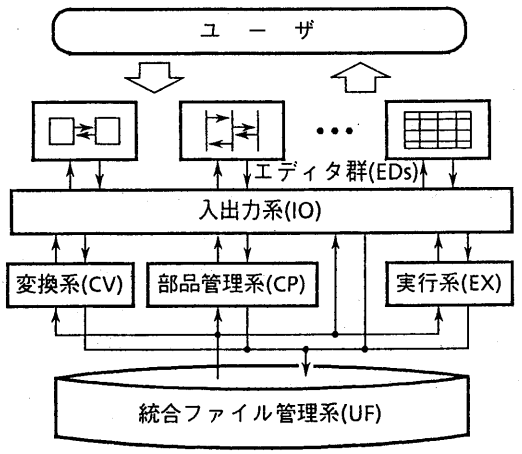



図1 プロトタイプで取り扱うデータ構造

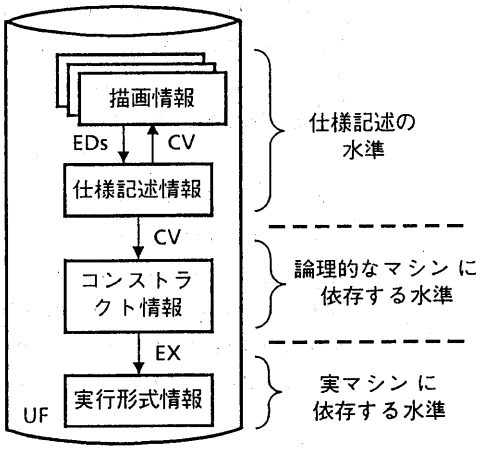



図2 プロトタイプの論理機能構成

表 1 図的表現形式の種類と記述内容

| 表現形式 | 記述内容 |
|---|---|
| 機能ブロック図 | モジュール間の接続関係 |
| シーケンスチャート | 入出力データの因果関係、モジュール間のデータ送受の関係 |
| 表操作図 | （関係演算を中心とする）構造体データ処理 |
| 決定表 | 選択構造 |
| データブロック図 | データの包含関係 |
| 関係表 | 関係データ構造 |

数として機能する一本のプログラムとして作成したが、ファイル操作における UF の位置づけは論理設計通りとし、また、処理系間の情報交換をパケット形式のデータ引数を基本とした宣言的インタフェースで実現するなど、各処理系の独立性は保存した。

一方、複数の仕様記述を同時に参照、更新できる環境を提供するため、エディタプログラム群は個々の独立したプロセスとしてワークステーション上のマルチウィンドウ環境に実現し、1モジュール（あるいは1データ構造）の仕様記述を1つのプロセスに対応したエディタで進められるようにした。

## 3 仕様記述環境

### 3.1 図的表現形式

仕様記述対象の一例として事務処理分野を想定し、表現形式の検討を行った結果、多様性および了解性の観点から表1に示す6種類の図的表現形式を採用した。表操作図は、表形式のファイル処理を中心とする事務処理に合わせて、今回新たに設計した表現形式である。その他の表現形式については、部分的に事務処理向けのルールを設定したものもあるが、基本的に従来から用いられている表現形式を、特別な制約を加えることなく採用している。

### 3.2 仕様記述方法

利用者の仕様記述過程に制約を与えないために、以下の仕様記述方法を用意した。

1. 仕様記述の開始は、システムメニューから記述に利用したい表現形式を選択し、その表現形式に対応するエディタプロセスを起動することによって始められる。記述対象となるモジュールまたはデータに対する名前は記述開始時に与えることも、記述過程の中で設定することも可能である。

2. 独立に記述していった仕様に対して、モジュールの階層関係やデータの定義側と参照側の関係等の相互関係を定義したい場合には、対象となる2つの図的要素をマウスで指定することによって対応付けることができる。

一方、モジュールを階層的に詳細化していく場合等、情報の関連性を追って記述を進めて行きたい場合には、現在記述中の仕様記述内の図的要素を指定して、その詳細記述を開始することが可能である。同様に、前記方法で対応付けを行った図的要素を介して相手を参照することも可能である。

### 3.3 仕様記述情報の相互変換

仕様記述環境で提供される表現形式は、個々に特徴的な情報記述能力を備えると同時に、相互に共通した情報記述能力も備えている。このため、同一のモジュール／データの仕様記述、あるいは階層関係にあるモジュール／データの仕様記述にこれらの表現形式を組み合わせて用いていけば、それぞれに共通な情報を基に各表現形式で固有な情報を付加していくことによって、仕様記述情報の詳細化を進めていくことが可能となる (図 3)。

このとき、各表現形式間で共通となる仕様記述情報、さらに、データ構造定義と参照、モジュールの階層間のインタフェース情報を可能な限り相互変換して提示することができれば、利用者は、それぞれの部分において本質的に追加しなければならない情報のみを記述すれば良く、効率的な仕様記述作業を進めること

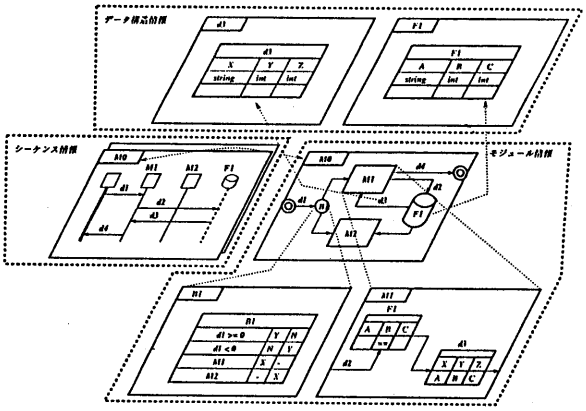


図 3 仕様記述情報の詳細化

ができる。また、相互変換結果を用いて、特定の情報を異なる側面から眺めることによって、それを記述した側面からは認識しにくかった矛盾を発見することも可能である。

プロトタイプでは仕様記述情報の設計において、システムを構成する個々のモジュールの動作およびモジュール間のデータ接続に関する情報（モジュール情報）、モジュールの入出力データの因果関係に関する情報（シーケンス情報）、データ構造の定義に関する情報（データ構造情報）を核とし、それぞれの情報が各表現形式を構成する個々の図的要素から得られる情報を統合していくことによって生成可能であるとともに、仕様記述情報から図的要素を再生成可能なデータ構造を採用し、変換系においてこのような相互変換機能を実現した。

### 3.4 部品参照

効率的な仕様記述環境の実現には、部品化した仕様を効果的に参照できる環境の提供も重要となる。プロトタイプでは、仕様記述過程で部品アイコンを記述することによって、システムから自動的に部品参照メニューが提示され、そのメニューを選択することによって目的部品を決定できるようにした。同時に、部品の入出力インタフェースについても仕様記述における表現形式（入出力ポート情報に関しては機能ブロック図、入出力データの構造については関係表）を用いて確認できるようにした。

### 3.5 プロトタイピング

仕様記述した内容を効果的に確認できるような環境を提供するためには、仕様記述過程と一体化したプロトタイピング環境の実現が重要である。プロトタイプでは、部分的に完結した仕様記述に対して、仕様記述の表現形式上におけるデータ経路記述部分（機能ブロック図のアーク記述等）に直接データ入力点、モニタ点を設定し、入力データの設定や出力データの表示は、入力点、モニタ点の存在するデータ経路に対応付けられたデータ構造定義のための図的仕様記述を介して行えるプロトタイピング環境を用意した。

さらに、利用者が意図したときに、即実行可能な環境を提供するため、仕様記述過程と並行して実行形式情報の生成が行えるようにした。即ち、変換系においては、仕様記述から獲得される情報をまず、連接構造として解釈して変換し、連接構造として解釈不可能な仕様記述が定義された段階で、その情報を制御情報として含み得る新たな制御構造への変換に移行する。この際、新たな制御構造の変換作業は、それまでに生成した連接構造情報に制御情報を追加していく形で実現する。同時に、実行系では、コンストラクト情報中の階層構造を展開し、処理構造の制御を行うための制御プリミティブを挿入し、実行モデルに対応した実行形

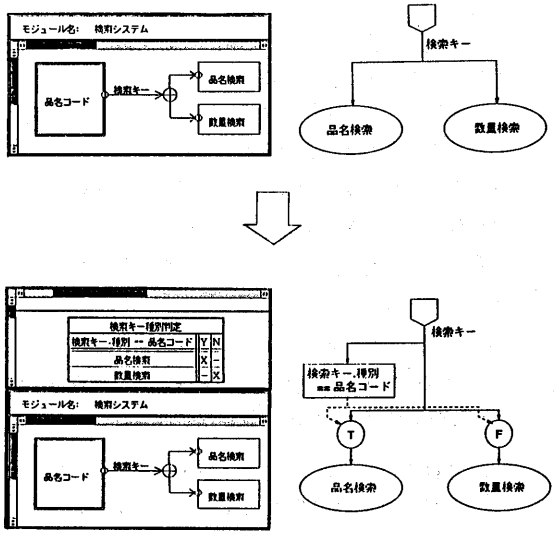



図 4 処理構造の生成過程の一例

式情報へ変換する。実行形式情報は、仕様記述環境における階層性に基づいた部分実行を実現するため、仕様記述上の階層性を保存しつつ展開を行っている。

図4は、仕様記述情報から加法的に処理構造を生成していく一例である。一旦、データコピーを含む連接構造として変換していた仕様記述に決定表の記述が追加されたため、決定表から得られる情報から獲得できる制御情報（図中点線部分）を元の連接構造に付加することによって選択構造を生成している。

## 4 仕様記述例

本プロトタイプによる仕様記述の例を図5に示す。

### 4.1 概要設計

概要設計では、機能ブロック図（FBD）を用いた機能構成の記述やデータブロック図（DBD）、関係表（RT）を用いたデータ構造の設計等が行える。また、記述過程に制約がないため記述可能な部分から仕様記述を進め、後から対応づければ良い。ここでは、購買処理の外部構成や資材係の処理構成をFBDを用いて概要設計している。

### 4.2 詳細設計

発注処理の詳細設計では、FBDとシーケンスチャート（SC）を用いている。FBDでモジュール間のデータ接続は記述できるが、その順序関係までは定義していないため、それをSCで定義する。このように一つの表現形式に無理な拡張を与えること無く、自然な解釈に基づく表現形式を多面的に組み合わせることによって情報を詳細化していける。さらに、FBDで記述したモジュール情報は、相互変換によってSCに反映されるため、SCではデータの順序関係のみを定義すればよい。

また、発注処理中の発注書作成には部品を用いているが、ここに示すように仕様記述と一体化して部品参照が出来る。

不足品検索モジュールは、ファイル処理が中心となるため表操作図を用いて詳細記述している。このように処理内容に応じて表現形式を使い分けると言った、いわゆるマルチパラダイム的な記述も可能である。さらに、表操作記述中の表データも相互変換を通じてRTによるデータ構造定義から引用できる。

### 4.3 プロトタイピング

出来上がった個々のモジュールは、随時実行して検証できる。

発注処理に対するプロトタイピングでは、在庫必要数の入力データ経路に入力点を、発注書の出力データ経路にモニタ点を設定している。また、在庫必要数、発注書の各々のデータ構造定義を介して、入力データの投入、実行結果の表示を行っている。この様に仕様記述とプロトタイピング環境が一体化しており、仕様の検証や拡張が容易に行える。

## 5 評価

本プロトタイプが実現した仕様記述環境の使い勝手を評価する。

a. 仕様記述

仕様記述過程に制約がなく多面的な記述が可能であるため、仕様記述が進めやすい。また、相互変換や部品機能も記述性を向上させている。

b. プロトタイピング

仕様記述環境の中で、プロトタイピングが実現できるため、設計、開発、テストの処理手順を踏む必要がなく、効率的なシステム構築が可能である。

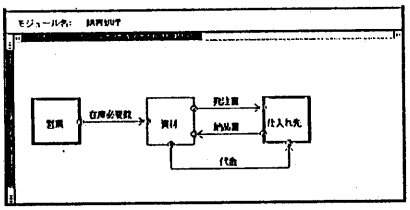
c. ドキュメント性

仕様記述そのものがドキュメント的効果を備えている。また、仕様の部分的変更を、相互変換を介して他の記述にも反映できるため、ドキュメントに一貫性がある。

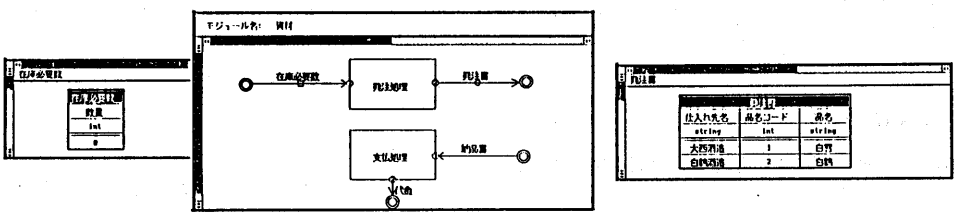
さらに、仕様記述、プロトタイピング、ドキュメント作成等の工程を意識しないで、システム開発が進めていける点が本プロトタイプの最大の特徴と考えられる。

## 6 おわりに

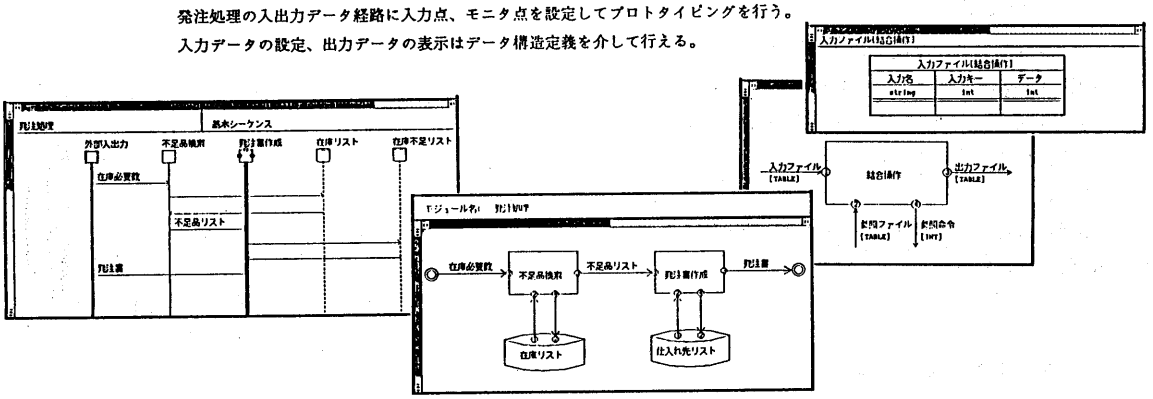
本プロトタイプによってAESOPの構想に基づく仕様記述環境が、効率的なソフトウェア開発を実現する上で効果的なものであることが確認できた。今後は、相互変換機能の拡張や異なる処理分野を対象とした表現形式の追加等を通じて、実用的なシステムの開発を目指して行きたい。



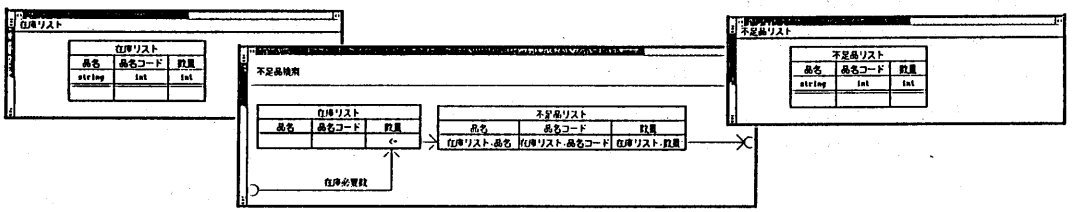
トップレベルの機能構成を機能ブロック図で記述。



発注処理の入出力データ経路に入力点、モニタ点を設定してプロトタイピングを行う。
入力データの設定、出力データの表示はデータ構造定義を介して行える。



発注処理の詳細を機能ブロック図とシーケンスチャートを用いて多面的に記述。
発注書作成処理は、部品を引用。



不足品検索処理の詳細を表操作図で記述。
在庫リスト、不足品リストのデータ構造は、関係表記述から引用できる。

図 5 記述例

本研究をご指導、ご支援頂いた関係各位に厚く感謝いたします。なお、本研究は、（財）大阪科学技術センターに研究会を設置して行われたものです。

## 参考文献

[1] 西川、寺田他,"超高位図的仕様記述環境（ＡＥＳＯＰ）の構想",情報処理学会研究会報告,90-ARC-83-2,1990