ICOT Technical Report: TR-169

TR-169

オブジェクト表現を用いたプログラム開発の支援環境
—プログラム開発における計画・管理システムの実現—

片山佳則(富士通)

April, 1986

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

オブジェクト表現を用いたプログラム開発の支援環境
－プログラム開発における計画・管理システムの実現－

片山佳則
富士通㈱国際情報社会科学研究所

1．はじめに

この研究は，研究者の意思決定を支援する環境[1]における一つの支援機能の実現を目的とするものである．様々な活動は，意思決定活動として幅広く捕らえることができる[2]．その際に，意思決定を行うためには，その状況を把握していることが必要である．ここでは，研究者が作成しようとしているプログラムを無駄なく効果的に表現させることに焦点を当て，そのための支援機能の実現を目的とする．この場合，その研究者が考えている情報あるいは知識を構造化させることが重要な支援機能となる．つまり，表現しようとするプログラムの構成要素を整理させ，それらの構成要素の関連性を明らかにさせることである．このような支援機能を実現することにより，研究者が効果的にプログラム開発を行えるようになる．

表現としては，現在プログラミング方式として注目されているオブジェクト表現におけるオブジェクトの記述を対象とする．オブジェクトを記述することは，従来のプログラムを開発することに対応する．従って，具体的にはオブジェクト概念を用いたプログラム開発を支援する機能を実現することが，この研究の目的である．この支援システム自身も，オブジェクト表現を使って実現されている．このため支援システムは，オブジェクト表現の持つ特徴や機能を十分に利用している．特にこの支援システムでは，モジュールごとに適宜状況が整理されることから，その状況において適切な対応をすることができ，オブジェクト間の明示的なpart-of 関係を利用することにより効果的な機能分担が実現されている．このシステムは，研究者・開発者と支援システムがインタラクティブな対話を進めることで，開発者自身にこれから行おうとしているプログラムの構成を把握させ，プログラミング活動を効果的に支援するものである．以下では，このような支援システムの支援対象と実現方法を区別するために，「オブジェクト表現を用いたプログラム開発」を「オブジェクト表現開発」と呼び，この「オブジェクト表現開発」を支援するために「オブジェクト表現を使って実現されている計画・管理システム」を「オブジェクト表現上の計画・管理システム」と呼ぶ．

本稿では，まずオブジェクト表現開発のための計画・管理システムの概要を記述し，さらにそのシステムの効果や有効性を議論する．最後に簡単な実行例をまとめて示す．

2．計画・管理システムの概要

研究者が行なうプログラミング活動の支援として，特にシステム開発に関する作業に視点を置くと，日程計画などのプロジェクト管理方式が一つの重要な役割として考えられる．オブジェクト表現上でプログラミング活動を実現する場合でも，基本的な単位が異なるだけで，そのアルゴリズムは一般のプロジェクト管理方式と同様の形式を取ることができる．

そこで，プロジェクト管理を支援するための中心的な手法である，PERT(Program Evaluation and Review Technique)[3]を採用する．これは，CPM(Critical Path Method) をその一部に含むものである．

2.1 PERTとオブジェクト表現

PERTをもとにした計画・管理システムを実現するために必要な作業は，次の３つに分類できる．

(1)プロジェクトのネットワーク化を行う．
(2)プロジェクトの時系列的な計画を立てる．
(3)資源配分を含め，実行可能な計画を作成する．

これらの作業を行うことに対応して，開発者は次のような支援を受けられる．

(1)による支援
- ・開発を進めようとしているシステムの構造を完全に把握する．
- ・各要素（作業）の動作を把握する．

(2)による支援
- ・システム開発上，重要視すべき要素作業を把握する．

(3)の支援では，これまでの計画を実際的に再構成することにより，実現性のある開発計画を作成する．

これらの支援機能は，様々なシステム開発に対してシステムの機能を整理し，開発方針を整理する上で重要である．

プロジェクト管理方式であるPERTをオブジェクト表現におけるプログラム作成に対応させた場合，ある機

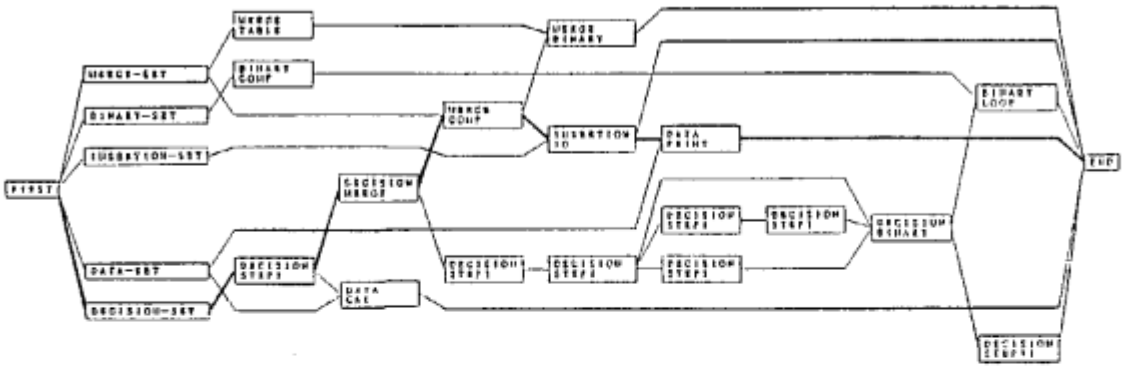

Fig. 1　オブジェクト表現開発の要素作業のネットワーク（後述の実行例のもの）

能を実現するために，いくつかのクラスオブジェクトを作成することが一つのプロジェクトとなる．プロジェクト内の各要素作業は，(1)オブジェクトの枠組みを作ること，(2)オブジェクトが持つメソッドをプログラムすること，と考えることができる．この場合，各メソッドの作成（あるいは，メソッドをさらに細分化したもの）をそれぞれ一つの要素作業とすることが，後に議論するオブジェクト間の関係を決定するための支援機能として重要になる．

例えば，多目的最適化プログラム[4]をオブジェクト表現上に開発する場合の各要素作業とその関係はFig. 1のようになる．この図に示したような要素作業であるオブジェクトのネットワーク構成や，時系列的な計画は，すべてメッセージ通信により動的に実現される．

Fig. 1はＳＫＥＴＣＨシステム[5]によって描いたものである．

2.2 オブジェクト表現開発のための計画・管理システム

PERTシステムをベースにし，計画・管理システムとしてオブジェクト表現上に実現することで，前述のような支援機能に加えて，さらに重要な支援を受けることができるようになる．これは，現実の開発過程にそって適切な工程の手順を示したり，システム開発の各時点で重要視しなければならない要素作業を把握させる，などの開発段階での動的な支援機能である．

プログラム開発を段階により大まかに分類した場合，(a)計画，(b)開発，(c)検証となる．この分類に対応させて支援システムを考えると，それぞれ次のようになる．

(a) PERTをもとにした計画システム

(b) (a)の情報を利用し，オブジェクト表現の効果を利用した開発・管理システム

(c) デバッグ・トレースのための支援機能

計画・管理システムとしては，さらに(a)の結果や(b)の管理情報をもとにした事後評価が必要となる．

これらの支援段階は，それぞれ独立したものではなく，相互に関連がある．実際に研究者に対して支援を行うためには，各段階での情報を共有し，利用しなければならない．

特に，オブジェクト表現では，オブジェクトの作成を個々に独立して実現できるため，この支援システムによって，開発工程などを正確に支援し，経過を統合的に管理できることは，独立的なオブジェクト表現の開発を補い，開発をよりスムーズに進める上でも重要な働きとなる．さらに実際にオブジェクトを表現していく過程において，初めに計画した要素作業（メソッドの動作）を変更することなど，様々な処理が必要となる．特に，要素作業の変更や追加などを，その開発と同時に計画・管理システム上で行うことで，それらの要素を含んでいるオブジェクト間の関係がより明確になり，新しいオブジェクト間の機能分担(part-of)関係を導きだすカギにもなる．このようにして，研究者の意思決定を支援するためのオブジェクト表現を用いたプログラム開発の支援環境が構築される．

本稿で示す計画・管理システムは，論理型言語上のオブジェクト表現・利用システムであるKORE/KR[6][7]を用いて実現されている．KORE/KR は，多種類のメッセージ通信方式やメッセージ処理の内部機構，オブジェクト表現方法，オブジェクト生成時の副作用など様々な特徴を持った表現・利用システムである．このKORE/KR は，それ自身で知識表現システムとなっているが，問題解決支援環境ＫＯＲＥにおける知識表現サブシステムでもある．

3．オブジェクト表現上の計画・管理システム

KORE/KR での計画・管理システムの基本的なオブジェクト構成をFig. 2に示す．計画・管理システムをオブジェクト表現上で実現することにより，実際に行う

プロジェクトは，scheduleクラスにつながるインスタンスが管理し，各要素作業はすべてjob クラスにつながるインスタンスとして表現される．これにより，複数のプロジェクトの計画・管理が一対のschedule/jobクラスオブジェクトを使って表現できる．

要素作業間の順序関係は，各 jobインスタンスオブジェクト間の関係として実現される．つまり，各インスタンスオブジェクトがその内部状態としてこれらの関係を持っている．要素作業としては，オブジェクトの枠組みを記述するだけでなく，各オブジェクトが持つメソッドも分割して要素作業とする．これは，オブジェクト表現において，メソッドを記述することが非常に重要な作業となるからである．

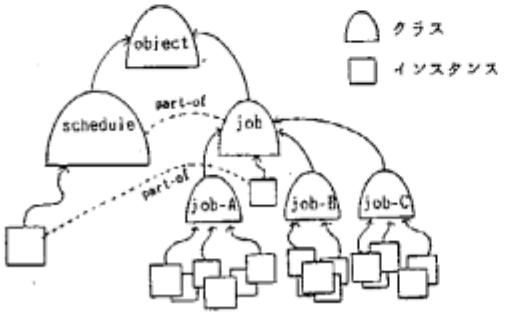

Fig. 2　計画・管理システムのオブジェクト構成

Fig. 2ではscheduleクラスと jobクラスが part-of 関係となっている．これは，すべての jobオブジェクトをscheduleオブジェクトのparts として表現するものではなく，scheduleオブジェクトの部分的な働きを jobの方で受け持つ関係を表しているものである．

各要素作業は，すべて一つのオブジェクトとして表現されるため，オブジェクトの抽象化表現によって容易に要素作業の分類した表現（図のjob-A,job-B 等）が可能となる．Fig. 1 の例では，開発するプログラムがデータの処理部(data オブジェクト),順序づけの計算部(insertion,merge,binary オブジェクト),実際に決定を行う部分(decision オブジェクト）に分類されている．

各要素作業の見積作業時間は，そのオブジェクトの内部状態（属性）として表現される．計画に必要な時間（最早時間・最遅時間）の処理は，すべての要素作業の時間が見積もられた後，計画段階や開発時に要素作業が終了した時点などで，scheduleオブジェクトにメッセージを送信することで行われる．要素作業間のネットワーク関係による時間の伝播は，オブジェクト間のメッセージ通信である．このようにして，支援活動に必要となる基本的な情報は，すべてメッセージ通信で確保される．システム利用時には内部での細かいメッセージ通信の処理動作を考慮する必要はない．

計画段階におけるPERTをもとにした計画システムについては，ここで挙げた表現方法や機能を実装させることで達成できる．開発や検証段階の計画・管理システムについても，オブジェクト構成は，Fig. 2 の形式を取り，さらに各オブジェクトに対して適宜必要な支援機能を付加することで，モジュラリティのある計画・管理システムが実現できる．この計画・管理システム全体の中で開発段階が支援活動の中心となる．

4．プログラム開発の支援環境

2.2 に示した支援段階の分類に対応させて支援環境の機能をまとめる．オブジェクトの基本的な構成は，Fig. 2 に示した通りである．

4.1 計画段階の支援活動

この段階で必要なことは，各要素作業に対応しているjob(インスタンス) オブジェクトを確実に作成し，必要な情報を導出／記録することである．

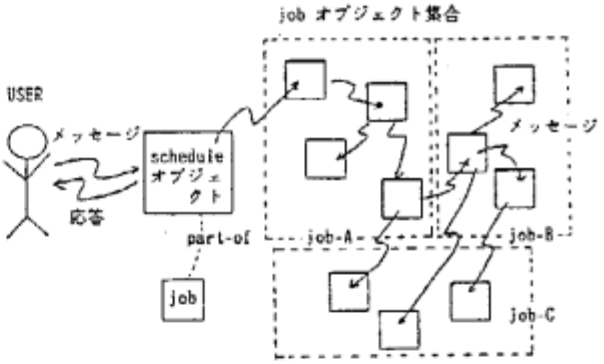

Fig. 3　支援システムのインスタンスの構成

jobクラスオブジェクトのすべてのインスタンス(jobオブジェクト）は，scheduleクラスオブジェクトの一つのインスタンス(schedule オブジェクト）がコントロールする．従って jobオブジェクト間の関係は，このscheduleオブジェクトが管理している．

2.1 で述べたプロジェクトのネットワーク化や時系列的な計画などの操作は，Fig. 3 に示すようにすべてこのscheduleオブジェクトに対してメッセージを送信することで行われる．

この段階の支援活動を行うために，各クラスオブジェクトは次のような表現と機能を持つ．表現がオブジェクトの属性定義（内部状態），機能がオブジェクト

の動作定義（メソッド記述）に対応する．

(1)scheduleクラスオブジェクト

［表現］
- jobの集合
- 実行可能な jobのレベル構造
- critical-path

［機能］
- 計画内にあるすべての jobのレベルづけを行う
- 計画において考えられるすべての jobのパス（初めから終わりまで）を求める．
- 計画内のすべての jobの最早開始／終了時刻，さらに最遅開始／終了時刻を job間の順序関係に従って各 jobオブジェクトに導かせる．
- プロジェクトの状況に応じたcritical-path を導く．

(2) jobクラスオブジェクト

［表現］
- その jobの所要時間
- job間の関係(before,after)
- 実行により導かれる時間値

［機能］
- jobオブジェクト間の順序関係において，関係する jobとのメッセージ通信により，その jobの最早開始／終了時刻，さらに最遅開始／終了時刻を導く．
- 対象としている jobの余裕時間（総余裕・自由余裕）を導く．

総余裕：作業の許容計画時間域－その作業の所要時間
自由余裕：前後の作業との関連により，次の作業に影響を及ぼさない余裕

ここに挙げた各クラスオブジェクトの表現は，2.2で分類したすべての段階に共通するものである．

この段階で，システム側から受ける支援は 2.1で述べたように，研究者の考えを整理させることが主である．従って，支援活動のために必要な機能として各クラスオブジェクトが持つものは，プログラム構造の情報を整理して確保することである．

この段階のシステム利用形態をFig. 4 に示す．

情報の確保のためには，Fig. 2 に示したオブジェクトの基本構成の他に，Fig. 4 のようにインターフェースとなるクラスオブジェクトを利用することになる．このようなオブジェクトを作成することで，すべてメッセージ通信だけで入出力を行うことができる．

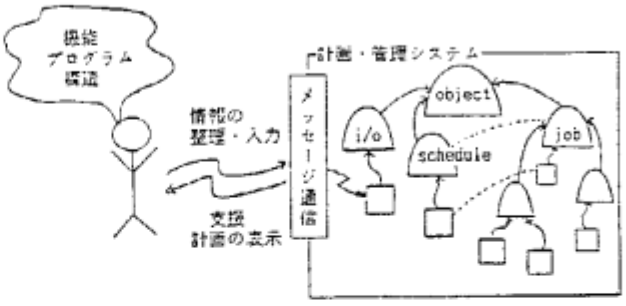

Fig. 4　計画段階のシステム形態

4.2 開発段階の支援活動

この段階での支援が，オブジェクト表現の開発を進める支援活動として重要な働きをするものである．計画段階で導かれた情報をもとにして，研究者からのメッセージにより，オブジェクトの開発工程を支援する．［後述の実行例(4)(7)(8)］システム側から与えられる支援情報は，計画段階での jobの順序関係に従った開発手順が中心である．4.1 に示したscheduleクラスが持つ機能を用いて，その状況に合ったcritical-path が導かれるため，重要視しなければならない jobの情報も，研究者に伝えられる．また，並行して作業する場合でもその情報が伝えられる．［実行例(7)］これらの情報を参照して，研究者がjob(オブジェクト内のメソッド記述やオブジェクトの作成）の処理を進める．

研究者が開発を行う過程においては，この他に様々な操作が必要となる．特に，計画・管理システムとしては，動的な jobの変更や追加などの操作が必要である．そこで，この開発段階では，これらに対応した柔軟な支援活動が行われなければならない．このシステムでは具体的に次の三種類の変更が可能である．

1)　job の順序関係の変更
2)　新しい jobを追加する
3)　job の削除

1)は，一通りの計画段階が終了し，全体を見通すことで，job 間の順序関係を変更する場合である．特に計画の内容が概略的な場合に必要となる．Fig. 1 の構成において，merge-compとmerge-binaryを入れ換えた例を示す．

```
| ?- simplex <- change_job,[mer_comp,mer_binary].
*** CRITICAL-PATH ***
      first
      decision_set
      d_step2
      d_merge
      d_step3
      d_step4
      d_step6
```

```
      d_step7
      d_binary
      bi_loop
*****************
yes
```

この場合にはcritical-path が変化している．（変更前のcritical-path は実行例(2)を参照）

2)は，計画に従って開発・実行する上で出てくる変更である．jobオブジェクト自体の作成は，単にオブジェクトを生成するメッセージ(new) を送信すればよい．その後，現在の計画の中でその jobの適切な順序を決める必要がある．これらはすべてメッセージ通信で処理される．

新しい jobオブジェクトが計画内に付加された場合には，計画段階の動作を再度起動する必要がある．研究者は，開発の進捗状況の情報を支援システムに入力しているため，すでに終了した jobは，計画の中から外されている．従って，再度計画段階の動作を起動する場合には，処理すべき jobだけが対象とされる．新しい jobが付加されることで,job間の critical-pathとして導かれていたものが，再構成される場合もある．［実行例(9)］

3)は，順序関係の変更としてもとらえられるが，計画に従って開発をすすめることで，終了した jobの処理としても扱われる．［実行例(6)］これは，開発の進展に合わせて計画・管理システムを更新する処理である．これを行うことで，支援システム側が計画の進行を確認し，次の処理への準備を行う．

Fig. 1 の構成でdecision-step2を取り除いた例は次のようになる.(critical-path が変化している)

```
| ?- simplex <- kill_job,d_step2.
*** CRITICAL-PATH ***
      first
      decision_set
      d_merge
      mer_comp
      in_io
      data_pr
******************

yes
```

これら三種類の変更機能を持ち，開発段階の支援が実現される．

この段階のためには，計画段階に示したscheduleクラスオブジェクトの機能に次の機能を付加している．

(1)scheduleクラスオブジェクト

［機能］
- 計画に従って，開発工程の支援を行う（必要な情報を研究者に与える）．
- job間の順序関係を変更させる．
- 現在の計画に新しい jobを追加する（同時に順序関係を更新する）．
- 終了した jobを現在の計画から除く（同時に次の支援活動を受けるための処理を行う）．
- 対象としている jobを開発状態とする．

この段階のシステム利用形態をFig. 5 に示す．この場合は，計画段階と異なり，Fig. 2 に示したオブジェクトの基本構成だけを利用する（入力のためのオブジェクトを使用しない）．

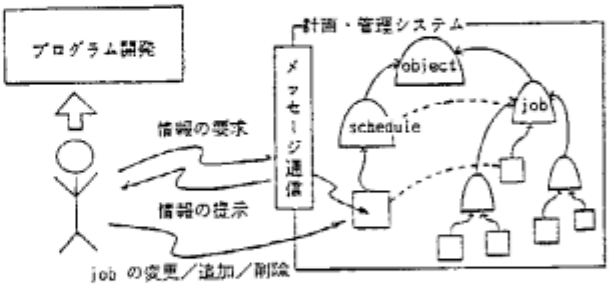

Fig. 5　開発段階のシステム形態図

4.3 検証・事後評価段階の支援活動

この段階では，開発段階において開発したプログラムの編集作業が行えることが望ましいが，現在の計画・管理システムとしては，これまでに進められてきた開発の過程をチェックする機能や，実行された計画の事後評価などの支援活動を中心としている．

この段階で受ける支援活動は，4.1,4.2 を終了したことにより，各インスタンスオブジェクト内に蓄えられた情報を参照し，計画と実行の相違等を検討することである．

計画・管理システムとして，特に，研究者の開発しようとしている対象の整理・確認，開発工程の支援に重点を置いているため，検証については，現在それほど考慮していない．

5．支援環境の効果

ここでは実際にこのシステムを利用して，オブジェクト表現を行う際の効果をシステムの評価とともにまとめる．

オブジェクト表現を用いたプログラム開発の際，計画・管理システムを利用する様々な形式が考えられる．一般に，

(a)：各オブジェクトの機能を，研究者がすべて明確に整理できている場合に利用する形式．

(b)：作成するシステムに必要な機能が明確には整理できていないが，ある程度の枠組だけを決めることができる場合に利用する形式．

に分けられる．さらにこれらを開発状況に合わせて分類すると次のようになる．

(1)個人が複数のオブジェクトから成るシステムを作成する場合．

(2)複数の個人が，いくつかのオブジェクトから成るシステムを作成する場合．

これらのどの場合でも，作成するシステムのオブジェクト構成をあらかじめ明確に規定しておく必要がある．本稿の計画・管理システムでは，これらのどの形式でもプログラム開発を行うことができる．(1)，(2)の形式の区別は，開発段階での支援の受け方によって行う．

さらに，大規模なプログラム開発プロジェクトの場合は，その内容をいくつかのサブプロジェクトに分割し，それらの個々についてこの計画・管理システムを利用することも可能である．

この時には，本支援システムでは同一のscheduleやjobのクラスオブジェクト表現を使ってすべてのサブプロジェクトを管理するインスタンスオブジェクトを容易に作成することができる．また，jobクラスオブジェクトのような階層関係(Fig.2)をscheduleクラスオブジェクトに導入することによって，プロジェクト全体を統一的に表現できる．

オブジェクト表現を用いてプログラムを開発する場合，一般には研究者個人が開発するプログラム全体の機能を常に正確に把握していなければならない［(1)の形式の場合］．複数で開発する時でも，各個人が開発するプログラムの機能を常時正確に把握している必要があり，さらに，プロジェクト開発の全体の状況などを管理している必要がある［(2)の形式の場合］．これらが実現されていないと，プログラム開発過程において無駄な手続きや操作が現れたり，最終的に期待通りのプログラムにするために，何度も検討し直すことにも成り兼ねない．また，実際に開発を進める場合，その経過を正確に記録することによって，複数の個人による開発作業の協調を取ったり，一人の場合でも別の作業と並行させて開発を進めることができる．

これらは，計画・管理システムを利用することにより解決され，実現することが出来る．

2.2 で分類した支援段階に対応させてこのシステムの有効性や効果についてまとめる．

［計画段階］

- 開発するオブジェクトの機能を支援システムを使って整理できる．
- オブジェクトの機能の利用関係（内部での利用や外部からのメッセージによる利用）を明らかにすることで，開発するプログラム全体の機能を常に把握できるような計画が構成される．
- 開発するプログラムが，どれだけモジュラリティのあるものかを容易に判断できる（例えば，要素作業の並行性に関しては，job数に比較して job間の並列パスが多い程プログラム開発における要素作業の独立性が高いと考えられる）．

［開発段階］

- どこまで終了したかを支援システム側で記録しているため，開発途中の場合でも，常に正確に把握することができる．
- システム開発が進むと同時に，計画・管理システムの内部状態も更新されるため，開発するプログラム全体の機能を常に把握できるだけでなく，開発の状況・次に行うべき最適な作業などを容易に確認することができる．
- 開発工程において，計画全体を見通しながら，計画を自由に変更でき，その変更に応じた適切な指示を受けられる．
- 計画段階で，オブジェクトが持つ各機能の利用関係によって開発工程のネットワークを決め，それにもとづいて開発を進めていくため，機能の分担関係(PART-OF) などを表現するオブジェクト間の関係表現の記述を素直に導くことができる（開発を進めることで新たに適切なPART-OF 関係を導くことができる）．これは，各メソッドの作成をそれぞれ一つの要素作業としたことによる．

［事後評価段階］

- 開発が一通り終了し，最終的な計画と初めの計画とを比較する事後評価により，別の開発を行う際の計画策定の参考となる．

この計画・管理システムは，オブジェクト表現を用いて実装されているため，研究者が支援を受け易く，開発者と支援システムとのインターフェースにそれほど重点を置く必要がない．すべてメッセージ通信である．

6.まとめと今後の方針

計画・管理システムのベースとして，PERTを用い，オブジェクト表現を用いたプログラム開発の支援に対して応用した．PERT自身は検査手法であるが，5.でまとめたように，オブジェクト表現の開発段階において非常に効果的な支援システムとなることがわかる．この計画・管理システムが持つ次の機能を，重要な特徴として挙げることができる．

(1) 開発対象がどれだけオブジェクト表現としてモジュラリティのあるものかの判断ができる.

(2) オブジェクトが持つ機能の開発工程が, その利用関係によって決められることから, オブジェクト間の関係表現を素直に導くことができる.

(3) 計画をメッセージ通信によって動的に変更でき, 各時点に合った支援が受けられる.

これらは, 対象をオブジェクト表現のプログラム開発に限定したことから言えるものである.

この計画・管理システムは, KORE/KR が持つほとんどの機能を利用しているため, KORE/KR 自体の表現能力を検討する上でも効果的なものであった. 特に job クラスとscheduleクラスの関係が, 統制関係と機能分担の関係になっている点において, PART-OF 関係を効果的に利用できている.

現在の計画・管理システムでは, 検証段階における支援活動が十分でないため, この点をさらに改良し, 今後は, これをKORE/KR 上のプログラム開発支援環境において, KORE/KR が持つ機能を十分生かせるような開発支援環境として発展させることを考えている. また, オブジェクト表現を用いたプログラム開発において, プログラムの機能を要素作業に分割する過程自体を支援する機能も充実させる必要がある.

本研究は, 第五世代コンピュータプロジェクトの一環として行ったものである.

謝辞

本研究に対して, 適切な御指導・助言を下さった戸田部長, 情報社会学室の新谷, 平石研究員に深謝いたします.

[参考文献]

1)新谷, 片山, 平石, 戸田：問題解決支援環境ＫＯＲＥ（その１）－概要－, 情報処理学会第32回全国大会, '86 年 3月 No.5L-8.

2) H.A.Simon (稲葉他訳) ：意思決定の科学, 産業能率大学出版部, '79 年.

3)五百井：ネットワークプランニング, 日刊工業新聞社, '74 年.

4) S.Kobayashi, A.Ichikawa : Interactive Binary Choice Method to Estimate Decision Maker's Utility Function in Multi-Attribute Decision Problem, Proc. of the Int. Conf.on Cybernetics and Society,pp1368-1372,1978

5) K.Sugiyama, M.Toda : Structuring Information for Understanding Complex Systems : A Basis for Decision Making, FUJITSU Scientific and Technical Journal,Vol.21,No.2,1985

6)片山, 新谷：知識テーブル（その利用）－知識ベースにおける対象指向表現とその処理機構の試作－, 情報処理学会第31回全国大会, '85 年 9月 No.1P-7.

7)片山, 新谷, 平石：問題解決支援環境ＫＯＲＥ（その３）－知識表現サブシステムKORE/KR とその概要－, 情報処理学会第32回全国大会, '86 年 3月 No.5L-10

[実行例] (Fig. 1のオブジェクト表現開発の実行)

```
| ?- file <- load,'schedule.o'.
>>  schedule.o  loaded.                    (1)

yes
| ?- file <- load,'simplex.i'.
>>  simplex.i  loaded.

yes
| ?- simplex <- critical_path.
*** CRITICAL-PATH ***                      (2)
     first
     decision_set
     d_step2
     d_merge
     mer_comp
     in_io
     data_pr
*****************
yes
| ?- simplex <- pp.

**********simplex object **********        (3)
     classname schedule
     >value set<
      value  job_list
               [[first,decision_set,data_set,
                insertion_set,binary_set,
                merge_set,d_step2,d_merge,
                d_step3,d_step4,d_binary,
                d_step5,d_step6,d_step7,
                d_step71,data_cal,data_pr,
                in_io,bi_comp,bi_loop,mer_comp,
                mer_table,mer_binary,end]
      value  job_level
               [[first],[merge_set,binary_set,
                insertion_set,data_set,decision_set],
                [d_step2,bi_comp,mer_table],
                [data_cal,d_merge],[mer_comp,
                d_step3],[d_step4,mer_binary,
                in_io],[data_pr,d_step6,d_step5],
                [d_step7],[d_binary],[bi_loop,
                d_step71],[end]]
      value  critical
               [first,decision_set,d_step2,
                d_merge,mer_comp,in_io,data_pr]
 end of object
yes
```

```
| ?- simplex (- next_schedule.

(4) Critical Path =
        [first,decision_set,d_step2,
                d_merge,mer_comp,in_io,data_pr]
    Next Job List =
        [merge_set,binary_set,insertion_set,
                data_set,decision_set]
    Job List Being Executed =  []
     ****************
    Next Job =  decision_set
        required time =  20
    Free Float =  0

    yes
| ?- simplex (- selected_job,decision_set.

(5)     required time =  20
     Comment = make object
     This job is selected !!

    yes
| ?- simplex (- finished_job,[decision_set,15].
    This job is Finished!!
(6) *** CRITICAL-PATH ***
        first
        d_step2
        d_merge
        mer_comp
        in_io
        data_pr
    ******************

    yes
| ?- simplex (- next_schedule.

(7) Critical Path =
        [first,d_merge,mer_comp,in_io,data_pr]
    Next Job List =
        [d_merge,mer_table,bi_comp,insertion_set]
    Job List Being Executed =  [d_merge,insertion_set]
     ****************
    Next Job =  bi_comp
        required time =  120
    Free Float =  240

    yes
| ?- simplex (- selected_job,bi_comp.

(8)     required time =  120
     Comment = compare with middlepoint
     This job is selected !!

    yes
    | ?- simplex (- finished_job,[insertiopn_set,15].

     This job is not on 'Job List Being Executed' !!

    yes
    | ?- simplex (- finished_job,[insertion_set,15].
    This job is Finished!!

    yes
| ?- simplex (- add_job,
                [d_step31,15,beforemd_step3]).
(9) *** CRITICAL-PATH ***
        first
        mer_comp
        in_io
        data_pr
    ******************
    yes
| ?- simplex (- selected_job,data_cal.

    required time =  200
 Comment = data * weight, total
 This job is selected !!

yes
| ?- simplex (- finished_job,mer_table.
This job is Finished!!

yes
| ?- simplex (- next_schedule.

(10)  Job List Being Executed = [data_cal,mer_comp]
     Wait for completion of these jobs, please.

    yes
| ?- simplex (- finished_job,[mer_comp,240].
    This job is Finished!!
(11) *** CRITICAL-PATH ***
        first
        d_step31
        d_step4
        d_step6
        d_step7
        d_binary
        bi_loop
    ******************

    yes
```

＜実行例の説明＞

(1)計画・管理システムのオブジェクトの読み込み.

(2)支援活動に必要な情報を導き，critical-path を導出する［simplex(プロジェクト名) はscheduleクラスにつながるインスタンスオブジェクト］.

(3)simplex オブジェクトの現在の内部状態の参照.

job-list : 計画上の jobインスタンスのリスト

job-level: 実行可能な jobのレベルリスト

critical : critical-path の jobリスト

(4)開発工程の支援を受ける.

次に実行可能なjob の集合(Next Job List) や現在実行している作業の集合，などを示し，自由余裕時間(Free Float)を計算して次に行うべき最適なjob（自由余裕時間が最小のjob)を示す(Next Job)

(5)ユーザがあるjob を実行することをシステムに示す.

(6)実行中のあるjob が終了したことを，実際の所要時間（この例では15）と共にシステムに示す．この時，critical-path が変わる場合には，新しく計算されたpathが表示される.

(7)並列に処理しているjob は開発工程の支援時にJob List BeingExecutedのところに示される.

(8)各job の内容は，job の選択時にそのjob オブジェクトの内部状態として記述されているComment が表示される.

(9)開発過程において新しいjob を自由に付加することができる.(この場合は,d-step3の後ろにd-step31というjob が付加されている．ネットワークは自動的にシステム側で更新される）この時,critical-pathに変更があれば，新しいpathが表示される.

(10)複数のjob を並行して実行する場合には，job のネットワーク関係によって待たなければならない時も出てくる.

critical-path は各job の実行が終了していくことで適宜更新される.