ソフトウェア工学 106－3
（1995．11．17）

オブジェクト指向プロトタイピングツール：ROAD/EE

田村直樹、中島毅、柳生理子、萩原正敏
三菱電機（株）　情報技術総合研究所

要求分析作業では、顧客の要求を漏れなく正確にしかも早期に仕様化することが求められる。我々は、プロトタイピング手法とオブジェクト指向分析・設計手法を融合することで、この要求抽出と仕様化の作業を支援するROAD/EE（ Real-time Object-oriented Analysis and Design / Execution Environment）の開発を進めている。本環境では、仕様記述モデルとしてRumbaughらが提案するOMT法を用いる。OMT法で利用される複数の仕様記述モデル間の整合性をチェックするする静的検証を支援すると共に、仕様記述を直接実行する動的検証機能を提供する。さらに、仕様記述を基にオブジェクトを動作させることで、仕様記述と一体化したプロトタイピングを可能としている。

本報告では、ROAD/EEのプロトタイピング機能とその利用方法について述べる。

An Object-Oriented Prototyping Tool: ROAD/EE

Naoki TAMURA, Tsuyoshi NAKAJIMA, Riko YAGIU, Masatoshi HAGIWARA

Information Technology R&D Center
Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura, Kanagawa, 247 Japan

The goal of requirement analysis is identifying all of customers' requirements and describing a consistent specification of those requirements. To archive these goals, we developed a prototyping tool named ROAD/EE (Real-Time Object-Oriented Analysis and Design / Execution Environment). In this tool, requirement specification is described with Object-Oriented specification models of the OMT method. The tool provides the facility of checking the consistency of specification descriptions and testing the behavior of the specifications with executing the specification descriptions directly. Also the tool supports building prototypes from given specifications, to visualize the behavior of the systems.

In this paper, we explain the facilities of this prototyping tool and its usages.

1．はじめに

ソフトウェアシステムに求められる要求は、近年ますます高度化、複雑化している。このため、システムに対する顧客の要求の仕様化を、いかに漏れなく正確にしかも早期に行えるかは、プロジェクトの成否を決定する大きな要因となりつつある。

要求分析・定義の段階の作業は、顧客の要求を抽出する作業と抽出された要求を仕様化する作業からなる。前者には様々なインタビュー法[1]やプロトタイピング手法が、後者にはある程度形式的な仕様記述モデルを用いる分析・設計手法がある。

プロトタイピング手法は、顧客の要求を素早く抽出する手法としてその有効性が広く認識されている。しかしプロトタイプを作成する上で、特殊なプログラミング言語を用いる場合が多く、その場合プロトタイプ開発が実システムの開発に対して付加的な作業となるため、開発現場からは敬遠されがちである。

一方、仕様記述モデルを用いる手法として、近年オブジェクト指向分析・設計手法[2,3]が注目されている。これらの手法が提供する仕様記述モデルは、複数の抽象化の視点からシステムの仕様を記述でき、またダイアグラムであるため比較的記述が容易である点で、仕様を記述する良い枠組みを提供する。しかし、一般にこれらの仕様記述モデルからシステムの動作イメージを理解することが難しいため、顧客参加の仕様のレビューに利用するには大きな壁が存在する。

我々は、プロトタイピング手法とオブジェクト指向分析・設計手法を融合し、上に示したような各々の手法の問題点を解決することを狙っている。具体的には、仕様記述モデルとして、Rumbaughらが提案するOMT法を用い、モデルの入力と実行とプロトタイピングを支援する環境 ROAD/EE （Real-time Object-oriented Analysis and Design / Execution Environment）の開発を進めている。本稿では、現状の仕様化技術が持つ問題を明らかにし、プロトタイピング環境としてのROAD/EEの利用方法を述べる。

2．現状の仕様化技術が持つ問題

2.1 要求仕様に対する検証作業

要求仕様は、その後のシステム開発のベースラインドキュメントとして利用される仕様である。このため、正しい要求仕様を短期間に作成することが、後の開発プロジェクトの成否を決定する大きな要因となっている。

顧客の要求を仕様化するプロセス（図1）は、

・顧客が抱く曖昧な要求の抽出

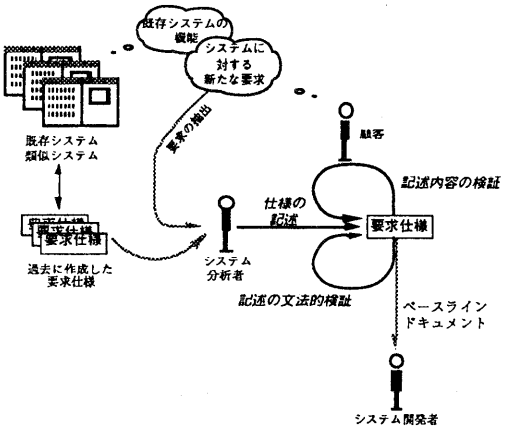


図1　要求の仕様化作業

・仕様の作成

・得られた要求仕様の検証

という3つの作業の繰返しとして定義される。このうち、要求仕様の検証は、得られた要求仕様の正しさを保証するための重要な作業ステップとなる。

我々は、既にこの要求仕様時におけるエラー原因の分類を行っており[4]、この結果から仕様の検証は大きく次の2つの側面から行わなければならないと考えている。

(1) システム開発者による検証

仕様が、利用している仕様記述モデルの持つ文法を正確に利用しているかを、システム開発者が確認出来る範囲で検証する。具体的には、仕様記述モデルの文法に従っているか、一貫性が保たれているか、また仕様記述モデルが必要とする仕様を全て記述しているかを検証する。

(2) 顧客による検証

抽出された仕様が、顧客の要求を漏れなく正確に反映しているか検証する。具体的には、得られた仕様記述が顧客の抱える要件を全て反映しているかどうか、また顧客の抱える要件の間で一貫性が取れているかどうかを検証する。この検証を正しく行えるのは、要求を抱えている顧客自身である。

2.2 現状の仕様記述モデルの課題

現在広く利用されている仕様化技術としては下記のものがある。各々の方法の利点と欠点を、仕様検証の容易性という観点から検討する。

(1) 自然言語を主体とした仕様記述

自然言語による仕様記述は、現在多くのプロジェクトで採用されている。自然言語を用いた仕様記述は、特別な知識を前提とせずに理解することが出来るという特長を持つが、自然言語の持つ曖昧さから顧客

の抱える要件の漏れや矛盾、システム開発者と分析者の間での解釈の相違を取り除くことは難しい。

(2)（準）形式的な仕様記述

ある程度形式的な仕様記述を採用することにより、できるだけ正確にシステムの仕様を記述しようとするアプローチがある。例えばオブジェクト指向分析・設計手法として広く着目されているOMT法[3]はそのひとつである。OMT法が提供する仕様記述モデルは、データの側面、動作の側面、及び機能的側面といった複数の側面からシステムを捉えるため、仕様の記述能力も高く、また図的な記法であるので比較的記述も容易である。さらにCASEツールを用いれば、各モデル記述内での文法チェックも容易となる。

しかし、OMT法では、得られた仕様記述からシステムの具体的な動作イメージを把握することが難しく、従って顧客の立場からこの記述内容を検証する目的で利用することは困難である。

(3)プロトタイピング手法

プロトタイピング手法は顧客の立場から要求を検証する有効な手法である。プロトタイプは、顧客に対して実際のシステム動作イメージを伝え、顧客の潜在的な要求を顕在化する（完全性の検証が行える）からである。

しかし、一般にプロトタイプの作成には、特殊なプロトタイピング言語が用いられ、プロトタイピングの結果得られた記述（プログラム）はそのままでは後工程へのベースラインドキュメントとして利用できない。この結果、プロトタイプの作成は本来の仕様化作業に対して付加的な作業となり、このため現場では敬遠されがちとなっている。

3．オブジェクト指向仕様記述の実行検証系ROAD/EE

3.1 ROAD/EEのアプローチ

我々は、記述能力と記述容易性を両立させた形式的な仕様記述モデルから直接プロトタイプを作成することによって、要求の抽出と仕様化の作業を融合し、要求分析作業の効率と品質を向上させることを狙っている。このための支援系としてROAD/EEの開発を進めている。

ROAD/EEでは、仕様記述モデルとしてOMT法の記述モデルを用いる。またOMT法では複数の仕様記述モデルを用いるが、ROAD/EEではモデル間の整合性チェックを行うと共に、これらのモデル間にいくつかの仮定をおくことによって、仕様の静的検証や、仕様記述モデルを直接実行する動的検証を可能にしている[5,6]。さらに、プロトタイピングを行うために仕様記述に現れるオブジェクトのアイコンを定義し、これを画面上で動作させるプロトタイピング環境を提供している。

以下、次節ではROAD/EEの各構成要素がどのような機能を表すかについて述べる。

3.2 ROAD/EEの構成

ROAD/EEは、次の構成要素からなる（図2）。

(1)仕様エディタ群
(2)ツールマネージャ
(3)実行エンジン
(4)プロトタイパ

これらのツールは、環境の機能拡張を容易にする目的で、各々独立したツールとして構成されている。

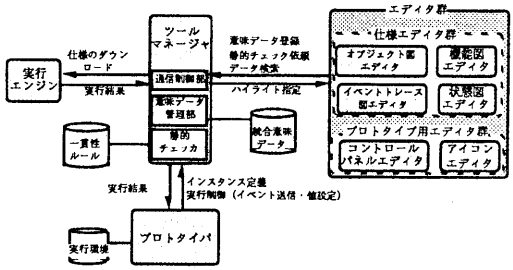


図2　ROAD/EEのシステム構成

(1)仕様記述エディタ群

ROAD/EEでは、OMT法の仕様記述モデルに対応した仕様入力用エディタを提供している。現在、オブジェクト図エディタ、状態図エディタ、イベントトレース図エディタ、機能図エディタを用意している。なお、これらのエディタは全てCASE環境構築用ツールキットCASEmaker[7]を用いて開発されており、仕様記述モデルの拡張にも容易に応えられる構成となっている。

(2) ツールマネージャ

ツールマネージャは、複数の仕様記述エディタから入力される仕様記述から意味データを抽出し管理する。また、エディタから登録される意味データを基に、複数の仕様記述モデルにまたがる整合性のチェックを行う。さらに、実行エンジンの実行を制御し、実行結果をエディタ、プロトタイパに通知する役目を担っている。

(3)実行エンジン

実行エンジンは、仕様記述及びインスタンス情報を基に状態機械を構成し、イベント入力によって動作をするインタプリタである。具体的には、プロトタイパ等から指示したインスタンス定義情報を基にインスタンスを作成し、各々のインスタンスを状態図に定義されたふるまい記述に基づいて動作させる。

この際、状態図中に現れるアクション、アクティビティ等の記述を基に各種演算も行う。更に、複数のインスタンスが並行に状態遷移を起こした時の非決定性の検出（複数の可能な状態遷移の発生、変数の同時更新）といった動的検証も行う。

実行エンジンが解釈する言語（実行言語と呼ぶ）の言語仕様はLISPをベースにしている。仕様エディタを通じて入力するOMT法の仕様記述は、ツールマネージャを通して実行言語に変換され、実行エンジンに送られる。ROAD/EEユーザにとって、実行言語を直接利用するのは、オペレーションの定義に限定されている。

(4) プロトタイパ

ツールマネージャが管理する仕様記述を基にプロトタイピングを行うツールである。プロトタイパは、仕様記述エディタから入力されたクラスや状態に対してアイコンを与え、これらのアイコンを実行エンジンの実行結果に基づいて動かすことで実現する。また、動作中のインスタンスに対して属性値を設定したり、外部イベントを入力するためのコントロールパネルをクラス毎に定義でき、インスタンスに直接働きかけることを可能としている。ROAD/EEでは、これらの機能を提供するため、アイコンの編集エディタ、コントロールパネルの編集エディタを用意している。また、実行エンジンの実行を制御する機能も提供し、仕様記述の詳細な検証を行うことも可能としている。

4．ROAD/EEによるプロトタイピングの方法

本章では、簡単なエレベータシステムを例題に、ROAD/EEを用いたプロトタイプの作成を作業手順を追って示して行く。仕様の記述と実行のメカニズムの詳細は、文献[5]、[6]を参照されたい。

4.1 プロトタイプの作成作業の手順

ROAD/EEを用いた場合、プロトタイプの構築作業は図3の手順に沿って行われる。

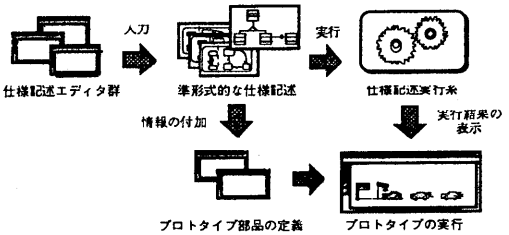


図3　プロトタイプの開発手順

S1: システム分析者は、まず仕様記述エディタ群を利用して、OMT法に沿った形式的な仕様記述を作成する。

S2: 独立に作成された仕様記述は、ツールマネージャによって統合される。ROAD/EEでは、この段階で文法上のチェック（仕様の静的検証）を行うことができる。

S3: 形式的な仕様記述モデルが揃ったら、各々のクラスや状態に対してプロトタイプ用の部品、すなわちコントロールパネルやアイコンを定義する。

S4: 最後にプロトタイパ上で、定義されたクラスを組み合わせてシステムのプロトタイプを構成し、これを実行する。

以下では、これらのステップのうち特に、S1、S3、S4について、作業の進め方に沿ってROAD/EEの利用方法を示す。

4.2 形式的仕様記述の作成

S1では、OMT法に基づいた仕様記述を作成するが、具体的には、以下の3つの仕様について定義を行う。

S1.1 オブジェクト図の作成

S1.2 動作仕様（状態図）の定義

S1.3 オペレーションの定義

(S1.1) オブジェクト図の作成

OMT法では、オブジェクト図を記述することから始まって、各々のクラスの状態図、機能図を記述する手順を基本としている。ROAD/EEでも、全ての仕様記述モデルは、オブジェクト図と対応関係を持って定義される。

オブジェクト図では、個々のクラスが持つ属性と、オペレーション名、及びクラス間の関連を定義する。オペレーションは、後で状態図のアクションやアクティビティとして利用されることになる。

図4にエレベータシステムでのオブジェクト図を示す。エレベータシステムでは、乗客が乗る篭（「箱」クラス）と、エレベータを呼び出す各階のエレベータホール（「階床」クラス）から構成される。ひとつのエレベータは、複数の階から呼ばれることを考慮して、多重性を定義しておく。

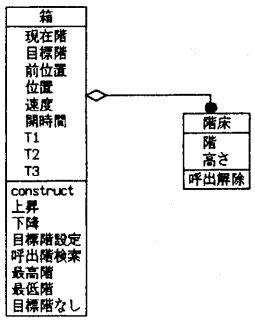


図4　エレベータシステムのオブジェクト図

(S2) 状態図の定義

個々のクラスに対して、各々のインスタンスオブジェクトの動作仕様となる状態図を定義する。図5、6には、エレベータシステムに対して用意した状態図を示す。図5は「箱」クラスに対する状態図を、図6に「階床」クラスに対する状態図を示す。

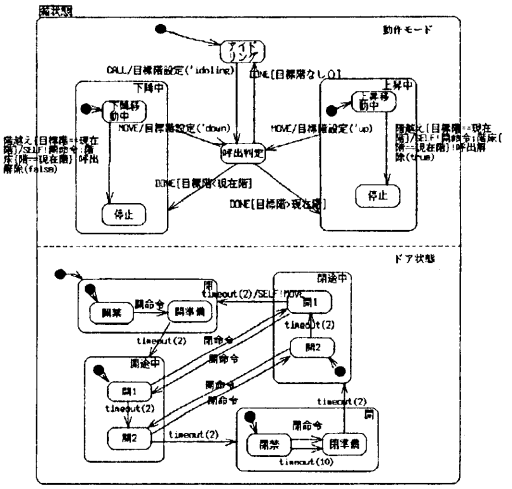


図5　クラス「箱」の状態図

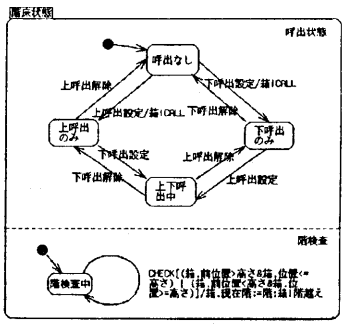


図6　クラス「階床」の状態図

ROAD/EEでは、仕様記述を実行可能とするために、状態図に対して以下の記法の拡張を行っている。

・オブジェクト間の通信メカニズム

・アクションとアクティビティの取り扱い

(1) オブジェクト間の通信メカニズム

ROAD/EEでは、オブジェクト間の通信は、イベントを用いてモデル化する方式を採用した。イベントは、基本的にブロードキャストされるものとして扱う。

特定のインスタンスに対してメッセージを与えたい場合は、明示的に送り先インスタンスを指定する送信相手の同定には、クラス間に張られる関連の「役割」名を用いる。ただし、役割名が指定されないときは、関連名や（クラス名によって一意に決まる場合には）クラス名で指定できるようになっている。

図7に「エレベータ」の状態図の中の状態遷移の典型的な記述例を示す。図7は次のように読むことができる。

「エレベータに対して階越えイベントが発生したとき、目標階と現在階が等しければ、フロアの階がエレベータの現在階に等しい停止階に対して呼出解除イベントを送信し、自分自身に開動作開始イベントを送信した後、状態1から状態2へ遷移する。」

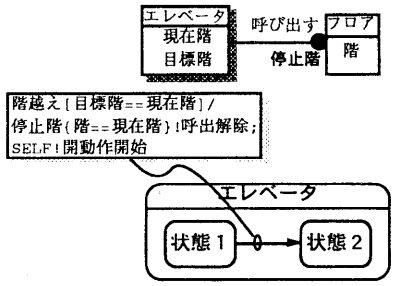


図7　メッセージ送信の例

(2) アクションとアクティビティ

オブジェクトインスタンスは、所属するクラス毎に定義された状態図に従い計算を実行する。その場合の計算とは、受信したイベントに対する状態遷移、それに伴うアクションの実行、状態に入った際のアクティビティの実行である。

状態遷移が発火する直前、アクションが実行される。ROAD/EEでは、アクションには「実行に時間を消費しない」処理が割り当てられることを仮定する。具体的には、イベントの送信、属性値への代入、オペレーションの起動を記述する。ここでオペレーション名を指定した場合、起動されるオペレーションは通常のオブジェクト指向言語と同様のスコープ規則で実行時に決定される。

一方、アクティビティには、時間に対して連続的に変化する属性値の計算を割り当てる。例えば、図5の状態図内の「上昇中」状態には次の計算を割り当てた。

位置 = 位置~ + 速度 × DT

ここで、「位置~」は1単位時間ステップ前の「位置」を保存する値であり、DTは刻み時間幅（単位時間経過イベントの間隔）を表す。これによって線型の微分方程式の近似式を表現している。この計算式は、オブジェクト図のオペレーションに定義することも可能であり、この場合はその状態がアクティブな間オペレーションが継続して実行される。

(S1.3) オペレーションの定義

オブジェクト図で定義されたクラスにおける個々のオペレーションの定義を行う。クラスのオペレーションの定義では、個々のオペレーションの具体的な処理内容を定義する。これは、現在、実行エンジンの言語仕様に従って記述される個々のLISPの関数として記述する方法を取っている。

図8に、「箱」クラスのオペレーション「目標階設定」の定義画面例を示す。

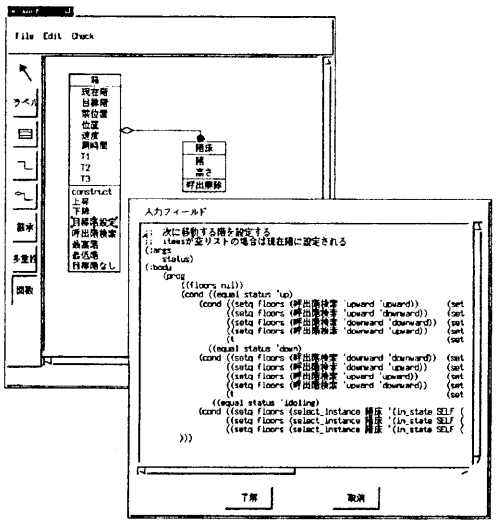


図8　オペレーションの定義画面例

4.3 プロトタイプ部品の定義

ROAD/EEのプロトタイピング機能の基本的なアイデアは、形式的な仕様記述モデルに対してその動作を表現する「アイコン」を与えることで、システムの動作イメージを表現しようというものである。

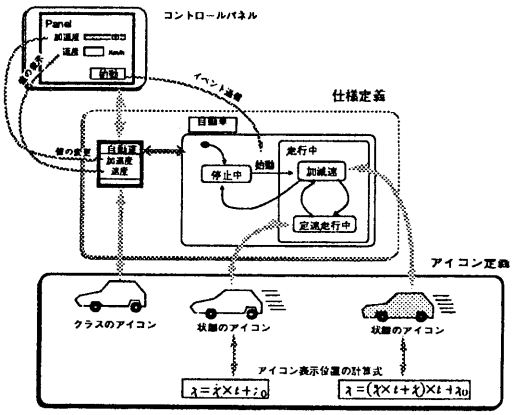


図9　仕様記述とアイコン・コントロールパネルの対応

ROAD/EEでは、図9に示すような対応関係で仕様記述モデルに対してアイコンとコントロールパネルを設定することが出来る。すなわち、ひとつのクラスに対して、オブジェクト図におけるクラス定義に、状態図、アイコンの定義、コントロールパネルの定義をまとめて、蓄積をすることが出来る。我々は、ROAD/EEを用いて、このように各クラスの仕様記述を仕様ライブラリとして蓄積して行くことによって、仕様記述の再利用を図ることを狙っている。特定の業務分野（ドメイン）に対して複数のシステムの分析を繰り返す過程でこの仕様記述ライブラリを充実させることで、ドメイン知識の集積を行ない、顧客の業務知識を反映した要求仕様化の作業をより円滑に進められると考えている。

S3プロトタイプ部品の定義では、次の2つの作業を行う。

S3.1 アイコンの定義

S3.2 コントロールパネルの定義

(S3.1)クラス、状態に対応したアイコンの作成

プロトタイパ上で動作を確認したいオブジェクトに対してアイコンを設定する。図10に「箱」クラスにおけるアイコンの定義を示す。まず「箱」クラスに対して1個のアイコンを設定する。また、「箱」クラスの中で注目すべき状態に対して、その状態を視覚的に表現するアイコンを定義する。「箱」クラスに対しては特に扉の開閉状況を把握したいため、「箱」クラスの状態図のうち「ドア状態」を視覚的に表現するアイコンを用意した。

また、各アイコンに対して表示位置を決定する計算式を定義する。プロトタイプ実行時にはこの計算式を用いてアイコンの表示位置を決定する。

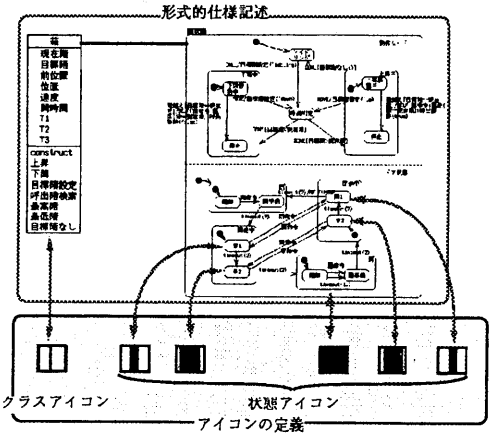


図10 アイコンの定義

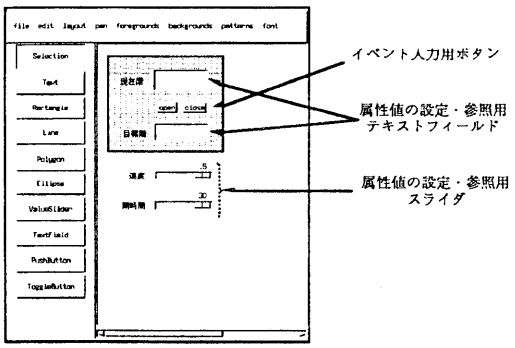


図11 「箱」クラスに対するコントロールパネルの定義

(S3.2) コントロールパネル画面の定義

プロトタイプでは、システムを取り巻く環境からの様々な入力がシステムの動作に対して与える影響を知ることも重要である。ROAD/EEでは環境からの入力をユーザが代表して行うものとして、「コントロールパネル」と呼ぶユーザインタフェースを用意する。個々のクラスによって利用できる属性値やイベントが異なるため、このコントロールパネルは各クラスに対応して定義する。ユーザは、コントロールパネルを通じて、このクラスのインスタンスの属性値を参照、設定し、またインスタンスに対してイベントを与える。

図11は、「箱」クラスに対するコントロールパネルの定義である。コントロールパネルには、そのインスタンスオブジェクトが外部から受け取る（ユーザが送り込む必要のある）イベントを生成するためのボタンと、ユーザが参照したり、設定したりする必要のある属性値と対応付けたスライダやテキストフィールドを用意する。例えば、「箱」オブジェクトに対して、ユーザが扉の「開命令」「閉命令」イベントを入力するボタンを用意する。また箱の属性である速度と開時間を設定する目的でスライダを用意している。さらに、現在階、進行階という属性の変化を表示する目的で、テキストフィールドを用意した。

4.4 プロトタイプの作成と実行

仕様に対してアイコンとコントロールパネルを作成した後、S4ではシステムの動作を模擬するプロトタイプの作成と実行を行う。このステップの作業は以下の手順に従う。

S4.1: インスタンスの生成

S4.2: リンクの設定

S4.3: 場の定義

S4.4: プロトタイプの実行

(S4.1) インスタンスの生成

プロトタイプの構築は、これまでに定義したクラスのインスタンスを作成することから始まる。

インスタンスは、各々個別に状態値と属性値を持つ。状態値はそのインスタンスのクラスの状態図上でのアクティブ状態を一意に決めるものである。属性値はそのインスタンスのクラスに対して、オブジェクト図上で定義された属性に対して設定された値である。スーパークラスで定義された属性に対する値も個々のインスタンスが管理する。

インスタンスの作成は、生成するインスタンスのクラスをアイコンパッドから選択し、実行画面上に配置することで行われる。画面上にアイコン（インスタンス）をひとつ置くと、このインスタンスに対するコントロールパネルが自動的に生成される（図12）。ここで個々のインスタンスの属性値の初期値設定を行う。

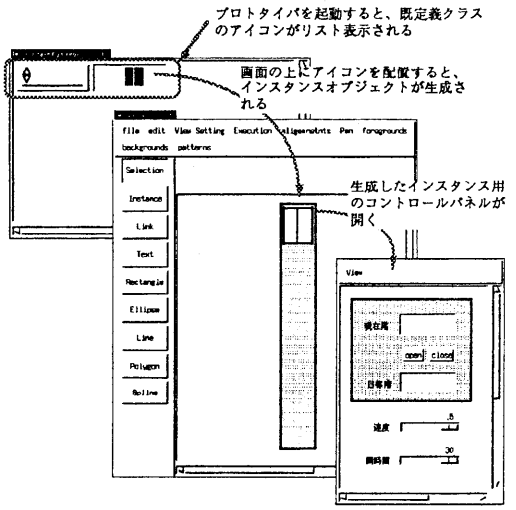


図12 インスタンスの生成

(S4.2) リンクの生成

プロトタイパの画面上にインスタンスを配置し終わったら、各々のインスタンス間にリンク（オブジェクト図における関連のインスタンス）を設定する。

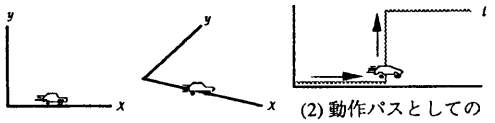
ROAD/EEでは、関連のインスタンスであるリンクの生成は、インスタンスの生成と独立に行う。これは、あるクラスのインスタンスは複数同時に存在し、例え1:1の関連であろうとリンクの接続先インスタンスを一意に決定できないことによる。また、インスタンス間でのイベントの送信には、リンクがアクセスパスとして行われる。

(S.4.3) 場の定義

「場」とは、インスタンスに対応するアイコンが置かれる座標系である。個々のインスタンスに対して、その属性値と2次元座標値との対応関係を計算式として与える。その計算式よって求められた座標値は、この「場」に定義された座標軸上に写像される。

例を図13に示す。図13(1)は、座標変換の目的で「場」を定義した例である。このように座標系を定義した場

合、アイコンの運動方程式は2次元の運動として定義する。図13(2)は、オブジェクトの動作パスを「場」として定義した例である。この場合、動作パス上での距離を算出する式として、アイコンの運動方程式を定義する。



(1) 表示座標の定義例　　(2) 動作パスとしての定義例

図13 場の定義例

(S4.4) プロトタイプの実行

ここまでの作業を行うことによって、プロトタイパの画面上に構築されたシステムのプロトタイプが動作可能となる。インスタンスの実行は、実行エンジン上で行われる。実行エンジン上ではインスタンスは、対応するクラスの状態図に従って動作し、インスタンスの状態値と属性値を更新する。この結果はプロトタイパに伝達される。プロトタイパは、インスタンスの状態値が変更されれば、インスタンスのアイコンをその状態に対応するアイコンに置き換える。これによって、プロトタイパ画面上で個々のインスタンスの状態を確認することが出来る。状態にアイコンが設定されていない場合は、クラスに設定されたアイコンが表示される。またインスタンスの属性値が変更されれば、これに対応してアイコンの表示位置やコントロールパネル上での表示内容が更新される。これによって、システムの具体的な動作イメージを顧客に伝えることが可能となる。

また、実行の途中で個々のインスタンスのコントロールパネルを通じて個々のインスタンスの属性値を任意に変更することも可能である。これにより、システム外部とのインタラクションによってシステムがどう動作するかを検討することも可能としている。

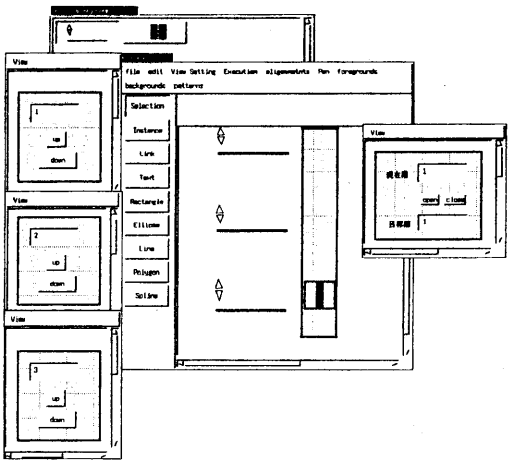


図14　プロトタイプの実行

図14にエレベータシステムでのプロトタイプの実行状況を示す。ここでは、3階のエレベータホールで「下降」を押すことによって、エレベータが3階まで移動しドアを開いて閉じる、という一連の動作を行わせた。また、エレベータ移動中に「箱」から開命令を送付した場合の動作も確認した。今回利用した記述では、エレベータの扉が開く結果になり、これにより図5の状態図の定義が誤っていることも確認出来た。

5．まとめ

本報告では、現在開発を進めているオブジェクト指向仕様記述の実行・検証系ROAD/EEの狙いとプロトタイピングの方法を例を交えながら示した。

今後は、様々な分野での適用を試み、プロトタイピング環境としての機能の充実を図るとともに、特定ドメインに対応した仕様記述部品ライブラリの充実化を図り、仕様レベルでの部品化・再利用の効果を検討して行く必要があると考えている。また、仕様記述部品を充実することにより、プロトタイプの画面設計から始めて仕様記述モデルを作成することを支援することも将来的な拡張として考えている。

参考文献

[1] C.Potts, et.al, "Inquiry - Based Requirements Analysis," IEEE Software Vol.11 No.2, pp.21-32 (1994-3).

[2] G.Booch, "Object-Oriented Analysis and Design with Applications," The Benjamin/Cummings Publishing (1994).

[3] J.Rumbough, et.al, "Object-Oriented Modeling and Design," Prentice-Hall (1991).

[4] T.Nakajima and A.M.Davis, "Classifying Requirements Errors For Improved SRS Reviews," Proc. of REFSQ'94 (1994-7).

[5] 田村他,"オブジェクト指向仕様記述の実行検証系：ROAD/EE," 情処学会 SE-102-11 (1995-1).

[6] 中島他, "新しい要求分析スタイルの構築～オブジェクト指向仕様の実行検証系 ROAD/EE～," 情処学会 サマーワークショップ・イン・立山, pp. 73-80 (1995-7).

[7] 田村他, "CASE環境構築用ツールキットの開発," 第44回情報処理学会全国大会 3J-3 (1992).