5S-1

オブジェクトに基づくシステム開発技法
― 状況投影モデルの提案 ―

吉田ゆき　大須賀昭彦　楠井洋一
(株式会社東芝　システム・ソフトウェア技術研究所)

1．はじめに

システム開発において要求仕様を基に機能仕様を作成する段階では、開発者が常に最初からシステム全体をイメージして設計を進めているとは限らず、部分的な設計や対象に関する断片的な知識を積み重ね、それらの相互関係から次第に全体像を明らかにしていくことも少なくない。我々はソフトウェア生産の工業化を行なうIMAP(Integrated software MAnagement and Production support system)の一環として、この様な設計過程を支援するために状況投影モデルを導入し、オブジェクトベースの仕様記述言語で記述した局所的な仕様の集まりからシステム全体の機能を組立てていく開発技法を提案する。

2．概要

要求から機能を獲得する場合には、断片的な知識を直感的に理解しやすい視覚的表現によってまとめていく手法が有効と考えられる。例えばＳＡ手法[1]で用いられるデータフローダイアグラムは、適度に形式的で直感的にもとらえやすく視覚面で多くのメリットを持っている。しかしながら、機能が増えるとフローが複雑になることも多く、多種の情報を一元的に記述する視覚的表現には限界があることがわかる。

そこで提案する技法では、視覚的表現の基となる意味情報を形式的な仕様記述言語で与え、システムの状況というパラメータを付加して必要な情報だけを浮き出させ表現する「状況投影モデル」を用いることにより、一元的記述を多角的表現でとらえるメカニズムを提供する。また、仕様記述言語をオブジェクトベースとすることで、記述の局所性、修正・変更の容易性を実現する。

オブジェクトベースの記述言語と状況投影モデルの二つを融合した本開発技法は以下の特徴を持つ。

(1)開発者は互いに独立なオブジェクトを記述するのみで、システム全体の構造はこれらの関係から自動的に得られる。
(2)状況毎の投影モデルを示すことにより、状況別のシステム機能、状況遷移、動作シーケンスなどの検証が可能。
(3)局所的で修正が容易な記述と豊富な検証機能の相互作用によって、試行錯誤を繰り返しながら段階的にシステム全体の仕様を獲得していくことができる。

なお、ここでいう「システムの状況」とは各オブジェクトの内部状態の集合のことであり、システム状況の動的変化を「システムの動作」、またシステムを動作させる情報の流れを「システムの機能」と考えることにする。

3．提案する開発技法

ソフトウェア開発においては、オブジェクト指向を用いた開発法としてオブジェクト指向設計法(OOD)[2]が知られている。この考え方は本技法が支援する開発段階にも応用できる。しかし、OODが実現という観点から適切なオブジェクト単位を抽出しているのに対し、ここでは機能獲得を目的として、システムについてわかっている(断片的)機能を投影する単位としてオブジェクトをとらえる。

本技法に適した記述言語として、ここではオブジェクト＋内部状態の変化 の概念を簡素に表現できるOBSL(Object-Based Specification Languege)を導入した。一般にオブジェクト指向言語は、メッセージが主体となって内部状態が変化するという考えに基づくが[3]、OBSLは内部状態が主体で、内部状態の変化を引き起こすきっかけとしてメッセージをとらえる。すなわち各要素について、要素のとりうる状態を定め、状態を変化させる原因を見つけ、状態が変わった結果何が起こるかを局所的に考え記述を行なう。ここでは特に、メッセージの送り先指定を不要として、各オブジェクト記述の局所性を高めている。また、この言語は部分的な記述でもある程度の実行ができ、必要に応じてプログラムのような細かい記述も書けるなどの特徴も持つ。

さらに、OBSLで記述した仕様を状況投影モデルを用いて検証するために、次の三機能を考える。それは、システムの状況に対応した機能のみを表示することで状況によってどういう機能が有効であるかを明確にする機能、各オブジェクトの状態の組合せの中から起こりうる状況を解析することで状況遷移を検証する機能、仕様を疑似的に実行し状況を順に変化させることでシステムの動作の流れを検証する機能である。

これらを用いた開発手順は以下の通りである。

STEP1: オブジェクトの記述
STEP2: 各状況における機能の検証
STEP3: 状況遷移の検証
STEP4: 動作シーケンスの検証

4．事例:切符の自動販売機

前章で述べたSTEP1～4の手順を具体例により詳しく説明する。例題として、図1に示す切符の自動販売機システムの設計を考える。ただし、簡単のため、カードによる販売や不正コインのチェック、きわどいタイミングのコイン投入はないものとする。

4.1 オブジェクトの記述(STEP1)

図1の券売機の記述を行なう。記述内容は図2に示すように、外部とのインタフェースと各オブジェクトより成る。外部インタフェースは、外部とやりとりされるメッセージの記述であ

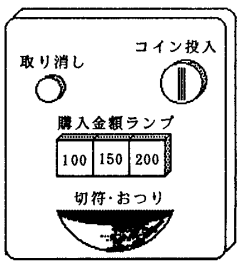

図1 切符の自動販売機

```
package 券売機 has

 interface
    input
      "コイン投入","取り消し要求";
      "切符購入要求(X)" where X is {100,150,200};
    output
      "切符発行","コイン返却(_)";
 end.

 object 投入金額管理 has
     state {累積なし,累積あり}
     initstate 累積なし
     action
       "コイン投入"     |                  ->累積あり | "投入累積";
       "切符発行"       | 累積あり->累積なし | "コイン返却(おつり)";
       "取り消し要求" | 累積あり->累積なし | "コイン返却(全額)";
 end.

 object 購入金額ランプ has
       100円 with (100);
       150円 with (150);
       200円 with (200);
   Instance with (Hny)
     state {ランプ消灯,ランプ点灯}
     initstate ランプ消灯
     action
       "投入累積"       | ランプ消灯->ランプ点灯 when Hny≦投入累積;
       "切符購入要求(Hny)" | ランプ点灯 | "切符購入受付(Hny)";
       "切符購入受付(Req)" | ランプ点灯->ランプ消灯 when Req=Hny;
       "取り消し要求"   | ランプ点灯->ランプ消灯;
 end.

 object 切符発行 has
     action
       "切符購入受付(_)" || "切符発行";
 end.

end.
```

図2 券売機オブジェクトの記述

Object-Based System Development Method
Yuki Yoshida, Akihiko Ohsuga, Yoichi Kusui
Toshiba corporation

り、この例では、外部からの操作に対応している。また、各オブジェクトはとりうる内部状態、初期状態、動作(受け取るメッセージ、メッセージによる状態の変化、送り出すメッセージと適用条件)より成る。図1では、購入金額ランプオブジェクトに三つのインスタンスが設定されており、100･150･200円ボタンに対応するインスタンス変数Mnyを持つ。このように類似のオブジェクトはあるクラスのインスタンスとして容易に記述できる。

4.2 各状況における機能の検証(STEP2)

STEP1の記述を状況別に抽象解釈し、各状況において有効な機能を投影する。例では、状態を持つオブジェクトは投入金額管理と購入金額ランプであり、図3.1～3.3に示す三つの状況について検証してみる。状況1(初期状況)において、投入金額管理オブジェクトが受け付けるメッセージは"コイン投入"のみである。このとき投入金額オブジェクトは"投入累積"メッセージを出す。一方、購入金額ランプオブジェクトは"投入累積"メッセージにより状態変化するがその後メッセージは出さない。従って、図3.1に示す機能が有効となり、この状況で有効な操作はコイン投入のみということが明確になる。同様に状況2の場合が図3.2、状況3の場合が図3.3となる。状況2,3では有効な操作が複数存在するが、個々の操作に対応する機能をシステムが把握しているため、例えば線種を変えて表示することなどでそれらを識別できる。

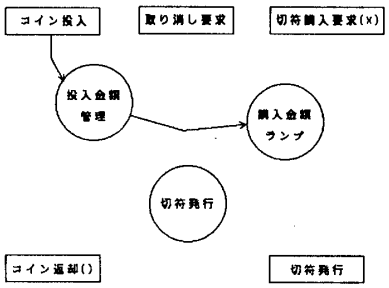

図3.1 状況1:{累積なし}{ランプ消灯}

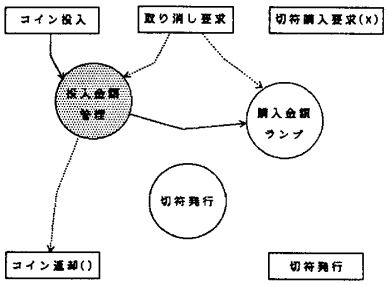

図3.2 状況2:{累積あり}{ランプ消灯}

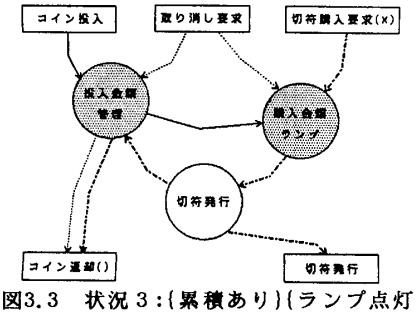

図3.3 状況3:{累積あり}{ランプ点灯}

4.3 状況遷移の検証(STEP3)

STEP3では各状況間の制約関係をSTEP1の記述より解釈し、初期状況から遷移可能な状況を明らかにすることで、起こりうる状況間のつながりを検証する。図4は、STEP2の各状況を単位として状況の遷移を投影したものである。これを見ると、{累積あり}{ランプ点灯}から{累積なし}{ランプ点灯}という開発者が期待していなかった状況に至っている。このように、状況間のつながりの誤りが発見できる。この原因を調べるためには次のSTEP4が有効である。

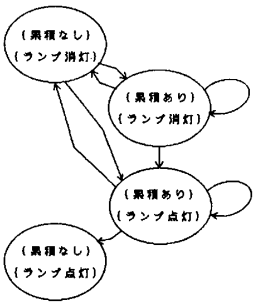

図4 状況の遷移

4.4 動作シーケンスの検証(STEP4)

STEP4ではSTEP1の記述を疑似的に実行し、各状況を動的に変化させることで、STEP3の状況遷移の順序を検証する。ただし、仕様において具体的なデータを処理すると記述が細かくなり過ぎる恐れがあるので、ここでは抽象的なデータを流し、処理の分岐する箇所は必要に応じて開発者に選択させることにする。例えば、状況1でコイン投入の操作を行ない、「投入累積の大きさが購入金額ランプの相当金額より大きい」という条件を選択すると状況3に遷移する。このように状況の移り変わりを順に調べていくと、状況3で切符購入要求の操作を行なった場合に{累積なし}{ランプ点灯}の状況となることが発見できる。従って、切符を発券した後に購入金額ランプオブジェクトの状態{ランプ点灯}を{ランプ消灯}に変える機能、つまり、

"切符発行"|ランプ点灯==〉ランプ消灯|

なる動作が必要なことがわかる。

5. まとめ

部分的な知識の記述(STEP1)、任意のシステム状況における機能の検証(STEP2)、状況遷移の検証(STEP3)、および疑似実行による動作シーケンスの検証(STEP4)により、局所的な記述から次第にシステム全体の仕様を獲得していく開発技法について述べた。このように、状況投影モデルを用いて仕様記述と表示機能を独立させたことにより、仕様をわかりやすく表現することが可能となった。この技法により獲得したシステムの仕様は、視覚的表現によらないので、他のモデルへ変換が可能となる。具体的な応用としては、個々の機能がわかっているが全体の動きがわかりにくいシステムの仕様化に適している。また、記述と表示の機能を独立させた概念は他の表現手法においても有効である。

6. おわりに

現在、この開発技法に基づく支援系の試作と記述実験を行なっている。今後の課題としては、記述言語の表現能力や時間の概念の扱いのほかに、OBSLにインヘリタンスなどのオブジェクト指向言語の特徴をうまく取り入れる試みや、記述をSSDダイアグラム[4]などのシーケンスダイアグラムへ変換する技法の研究などが挙げられる。

参考文献

[1]DeMarco,T.:Structured Analysys and System Specification,Yourdon,New York(1980).
[2]Booch,G.:Object Oriented Development,IEEE Trans.Softw.Eng.,Vol.SE-12,No.2(1986),pp.211-221.
[3]上谷:統合化プログラミング環境－Smalltalk-80とInterlisp-D,丸善(1987).
[4]楠井他:システム動作設計支援技法SSDESIGNの開発(1)～(3),情報処理学会第35回全国大会(1987).
[5]本位田他:推論型システム記述言語 MENDEL,情報処理学会論文誌,Vol.27,No.2(1986),pp.219-227.