1N－4

オブジェクト図とペトリネットを用いたシステムの表現方法の提案*

中川裕之†　白倉隆雄　冨樫陽太

キヤノンソフトウェア株式会社‡

1　はじめに

オブジェクト指向によるソフトウェア開発手法は、データと手続きをカプセル化したオブジェクト、即ち、「もの」を中心に実世界を表現しようとするもので、たとえば、構造化手法における機能中心のアプローチに比べ自然な問題把握を可能にしている。しかし、オブジェクト指向アプローチは、実世界(システム)の個々の構造は素直に表現できるものの、機能や状態の表現が不十分である。両者を素直に表現する方法について、これまでもいくつかの提案がなされてきたが、開発現場における実用性という点において充分ではなかった。

そこで本稿では、システムの妥当性を容易に確認できることを目標とした、実用的なシステムの表現方法の提案を行なう。即ち、構造はオブジェクト図を、状態はペトリネットを用いてシステムを表現しようとするものである。さらに、これらの記述からのソースコード(C++言語)を自動生成することとした。

2　システムの表現方法

オブジェクト指向の主な概念には(1)抽象化、(2)カプセル化、(3)継承、(4)状態、がある。システムの構造である(1)~(3)をオブジェクト図で、(4)をペトリネットを用いていかに表現するかについて述べる。

2.1　オブジェクト図の記述方法

個々のオブジェクトはその役割を果たすのに必要なデータ(属性)と手続き(メソッド)を内部に持っており、このオブジェクトを戦略で定めた視点に基づいて分類し、抽象化したものがクラスである。オブジェクト図は、クラス間の関係を実体関連図から派生した方法で表現しようとしたものである。クラスを表記するために図1に示した3段のアイコンを使用する。1段目にクラス名、2段目に属性名とその型、3段目にはメソッド名、引数そして返り値を記述する。クラスの継承は、図2に示すように「is-a」関係を用いて表現する。親クラスと子クラスを結んでいる半円形は、この関係が「is-a」であることを明示したものである。

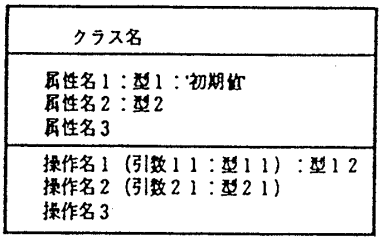

図1．オブジェクト図の記述方法

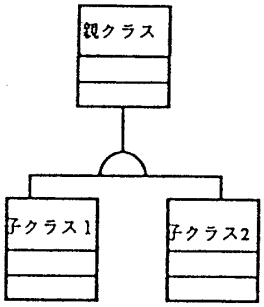

図2．継承の記述方法

2.2　ペトリネットの記述方法

ペトリネットは、システムの構造的、動的性質の検証が可能なグラフィカルなモデル化ツールである。さらに、このペトリネットのトークンに色をつけることでペトリネットを構造的に畳み込んだ、カラードペトリネット(CP-net)[1]がある。CP-netは構成的にはペトリネットと同一で、ペトリネットのアークにアーク式を、トランジションには角型括弧の中に記述されているガード関数という内部記述を付加したものである。

我々はこのCP-netにおける色をプログラミング言語における型とみなし、プレースに割り当てられていたカラー集合の要素(カラー)を、1プレース1カラーとすることで、オブジェクト指向における状態をフローチャートライクに表現できるようにした。アーク式では、C++言語で定義された式とメッセージパッシングを、ガード関数では条件式を、対応するネット構成要素の傍に記述する。

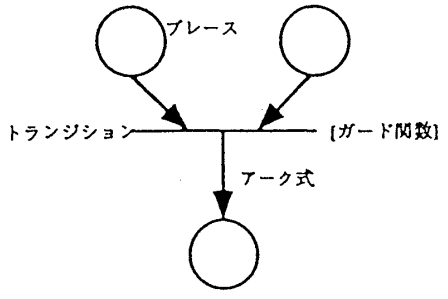

図3．カラードペトリネットの記述方法

*A Practical Object Oriented Design Method

†Hiroyuki Nakagawa,Takao Shirakura,Yota Togashi

‡Canon Software inc.,3-9-7 Mita,Minato-ku,Tokyo 108,Japan

3 ソースコードの自動生成方法

本システムにおけるソースコード (C++言語) 生成は、図式表現で記述されたオブジェクト図と CP-net を、一度、図式規則にもとづいて文字表現に変換、それをトランスレータの入力とすることで行なわれる。ここでトランスレータは、文字表現から C++言語への変換規則を入力としたトランスレータ生成系 PCCTS[2] の出力結果である。本節では、オブジェクト図とペトリネットの図式規則、ならびに、ソースコード生成規則について述べる。

3.1 図形表現から文字表現への変換方法

オブジェクト図の図式規則を以下に示す。

OM : { **icon** *classes* } { **arc** *relations* } .
classes : **class** ({*attribute*}) ({ *method* }) .
attribute : (**var type access**) .
method : (**funname** [*argment*] **rettype** *CPN*) .
relations : ([**superclass**],[**subclass**]) .

オブジェクト図の図式規則は、クラスの集合 (*classes*)、クラス間の関係を表した集合 (*relations*) で構成されている。さらに、クラスの集合は、クラス名 (class)・属性の集合 ((*attribute*))・メソッドの集合 ((*method*)) で構成されており、クラス間の集合は、親クラスと子クラスの名前を組とした集合 ((superclass ,subclass)) で構成されている。

オブジェクト図の図式規則中のメソッド (*method*) の要素の *CPN* は、CP-net を用いてメソッドが実装されることを表している。この CP-net の図式規則 (*CPN*) を以下に示す。

CPN : **plc**{*places*}**arc**{*arcs*}**trs**{*transitions*}.
places : (**place-id kind varname type**) .
arcs : (**arc-id , snode , enode ,** *expression*)) .
transitions : (**transition-id** *guard* ([*CPN*])) .

CP-net は、プレースの集合 (*places*)・アークの集合 (*arcs*)・トランジションの集合 (*transitions*) で構成されており、アークの集合はアーク式 (*expression*) を、トランジションの集合はガード関数 (*guard*) と *CPN* を含んでいる。ここで、トランジションの要素の中で再帰的に *CPN* を含んでいるのは複雑な CP-net を 1 つのトランジション (サブページトランジション) に置き換えることが可能であることを意味している。

図式表現で記述されたオブジェクト図と CP-net は、この図式規則にもとづいて文字表現に変換されている。

3.2 文字表現からソースコードへの変換方法

上述の文字表現は図式表現をそのまま変換したものであるため、直接ソースコードの生成は行なえない。たとえば、オブジェクト図の文字表現においては、クラス間の関係は 1 つアークを中心にそのアークに付着している (親子) クラスを表現しただけのものとなっている。この場合、複雑なクラス継承、たとえば、多重継承のように 1 つのクラスに複数の親クラスが存在する場合のコード生成がうまくいかない。そこで、このクラス間の関係をアーク中心からクラス中心へ変更することで、クラス間の関係を得ることができるようにする。変更された文字表現を形式的に表すと以下のようになる。

OM : **class** { **classname**
superclasses attribute method } .
superclasses : { (**super-classname** ',') } .

つまり、1 つのクラスに関係のある親クラスを全てそのクラスの集合の要素とするよう変更を行なった。

プログラムの基本的な制御構造は、繰り返し文 (WHILE-DO 命令) と、判断文 (IF-THEN-ELSE 命令) である。CP-net からソースコードを生成するにあたり、CP-net の (文字) 表現より WHILE-DO 命令、IF-THEN-ELSE 命令に相当する構造を探索し、CP-net の形式記述を以下のように変更する。

CPN : { *assign.* | *while-do* | *if-then-else*} .
while-do :**while** (*guardexp*) *true-statement* **do** .
if-then-else : **if** (*guardexp*) *true-statement*
else *false-statement* .

4 おわりに

オブジェクト図とカラードペトリネットを用いたシステムの実用的な表現方法を提案した。そして、この表現からソースコードを自動生成する方法を提案した。今後は開発現場に適用を図り、本システムの改良につとめていくことでソフトウェアの高生産性、高信頼性を確保していきたいと考えている。

謝辞 本システムの開発に協力していただいた当社ネクストプロジェクトの木村裕一氏に感謝します。

参考文献

[1] K.Jensen : Coloured Petri Nets 1,Springer-Verlag,1992.

[2] T.J.Parr,H.G.Dietz,and W.E.Cohen.PCCTS 1.00:The Purdue Compiler Construction Tool Set.*SIGPLAN Notices*,1992.