プログラミング-言語・基礎・実践 14－8
（1993．10．29）

オブジェクト・ネットワーク生成支援環境

柴田 洋[†] 佐藤康臣[††] 平川正人[††] 市川忠男[††]

[†]広島大学大学院 [††]広島大学工学部

概要

一般的にプログラミングの初期段階では、ユーザは使用する部品や呼出し関係を完全に把握しているわけではなく、それらを組み立てる過程で得ることが多く、オブジェクト指向では呼出し関係がネットワークになるなど複雑で、ユーザにとっての負担が大きい。

本稿では、オブジェクト指向プログラミングの呼出し関係がネットワーク構造になることに注目して、ユーザの関心の高い部品からスタートして、他の部品との依存関係の情報をもとに、ユーザの求める構造を得るための情報を効率良く引出すための質問列を自然言語でインタラクティブに生成し、その答えから得た情報によってユーザの求める構造の生成プロシージャを出力する支援環境を提案する。

An Assistant for Generating Object Network

H.Shibata[†], Y.Sato[††], M.Hirakawa[††], T.Ichikawa[††]

[†]Graduate School of Hiroshima University

[††]Faculty of Engineering, Hiroshima University

Kagamiyama 1-4-1, Higashi-Hiroshima, 724, Japan

Abstract

Generally, at an early stage of programing, the user doesn't know the total view of the objects available and the network for functioning these objects. The user usually gets the reality of objects in the process of coordinating objects. In an object-oriented programming, however, it's not an easy job since the network becomes so complicated.

This paper presents an assistant for generating object-networks. The assistant makes it feasible to generate questions for an efficient acquisition of the information about object-networks interactively and provide the procedure for generating object-networks on the basis of the information obtained in response to the questions.

1 はじめに

近年、コンピュータの普及とその高性能化が進み、ハードウェア進歩とソフトウェア進歩のギャップが広がるにつれ、大規模ソフトウェアへの対応のための、生成・保守サイクルの短縮化などの要求がますます大きくなってきている。その要求を満たすパラダイムとしてオブジェクト指向が注目され、その中で様々な研究が行われ成果を上げている。

オブジェクト指向プログラミングでは、継承機能を用いることによって、ソフトウェア部品をクラスを単位として階層的、統一的に定義できる。この機能により、既存のクラスの定義を変更することなくソフトウェア部品を拡張していくことができ、再利用性の面で優れている。また、各部品の機能が統一的に定義されているため、保守性にも優れている。

しかし、オブジェクト指向プログラミングでは、再利用により巨視的な生産性は向上するものの、クラスやメソッド定義などのために逆にコーディング量が増すなど、直接的な生産性は低下する傾向にある。また、オブジェクト間の構造は一般的にネットワークとなるが、ネットワーク構造ではオブジェクトidを共有しなければならないため、グローバル変数などを用いずに完全に独立したネットワーク構造をコーディングすることは困難であり、ユーザにとって大きな負担である。また、グローバル変数などを多用することは、オブジェクト指向が持つ本来の独立性、保守性を低下させる。本稿では、オブジェクト指向環境における、部品合成に基づく完全に独立したオブジェクトネットワークの生成を支援する開発環境AGONを提案する。これにより、ユーザのオブジェクト・ネットワークの把握、生成のためのコーディングなどを支援し、直接的な生産性の改善を図る。

AGONでは、ユーザの注目する部品から、その実行にあたって必要となる部品とそれらの部品間のネットワーク構造とを、部品機能に関する自然言語の質問列にYesかNoで答えることによって、オブジェクトネットワークの生成プロシージャの形で得る事ができる。

以下、2章でオブジェクト指向の概念と特徴を簡単に説明した後、3章で処理の概要、4章で質問のための知識、5章で質問列生成のための依存関係について述べ、6章で質問列の生成について述べる。

2 オブジェクト指向

オブジェクト指向プログラミングはモジュール性に優れ、その独立性が高いため、ソフトウェアの再利用性、保守性、信頼性が高い[1][2]。

オブジェクト指向プログラミングの特徴として、

- 継承機能により、クラスを単位として階層的、統一的にソフトウェアの部品化が行なえ、再利用性と保守性に優れている。
- オブジェクトを単位として考えることで、問題をプログラムに反映させやすい。
- 動的束縛を用いた、より柔軟なソフトウェアの開発ができる。

などがあげられる。

2.1 複合オブジェクト

オブジェクト指向プログラミングにおいても従来の手続き型プログラミングなどと同じように、高機能部品は一般に、複数の部品を組み合わせて作られる。

このように、複数の部品を組合せた部品を複合オブジェクトと呼び、複数のクラスのインスタンスを、その構成要素としてインスタンス変数に保持する。これらの構成要素は、そのクラス以下のすべてのクラスを総称した形で与えられるドメインによって規定され、通常、動的に配置される。

複合オブジェクトは、集約化の概念であり、部品の抽象化を行う有力な手法でもある。このように抽象化された部品は、再利用時に、インスタンス変数に保持する構成要素をドメイン内で変えることによって、その機能をカスタマイズして使用できるため、応用範囲は広く、再利用性が高い。

2.2 部品合成

部品合成の方法には、一般に、以下の２つの方法がある [3][4]。

- 要求仕様に合わせて部品を生成する方法
- 要求仕様に合った部品を検索・修正する方法

オブジェクト指向は再利用性に優れているため、要求仕様に合った複合オブジェクトを検索・カスタマイズし、該当する部品がない場合については、要求仕様に合せて部品を生成するという方法が最も適切であると考えられる。

しかしながら、プログラミングの初期段階において明確な要求仕様をユーザから引出すことは一般に困難であり、上記の方法はプログラミングの初期段階に用いることはできない。

3 処理の概要

本章では、AGON における処理の概要について述べる。

図 1は AGON における複合オブジェクト作成時の処理の流れを表している。図中の四角の枠はこれから作成しようとしている複合オブジェクト、○はその構成要素、矢印は要素間の参照関係を表している。

まず始めに、ユーザがプログラムの応用分野を選択し、システムがその応用分野に関する必須部品を示す。次に、ユーザは自分の注目する部品を追加し、逆に不必要な部品があれば削除する。

システムは、それらの選択された部品について、次章で述べる依存関係を求め、それらの依存関係とシステムの持つ知識などを用いて、構造決定のために必要な情報を効率よくユーザから得るための質問を生成する。また、質問に対するユーザの答えに基づいて不足部品を補い、依存関係を更新し、完全な構造が得られるまでこれを繰返す。

そして、最終的に得られた情報をもとに、ユーザの求める構造を作成するオブジェクト生成プロシージャを出力する。

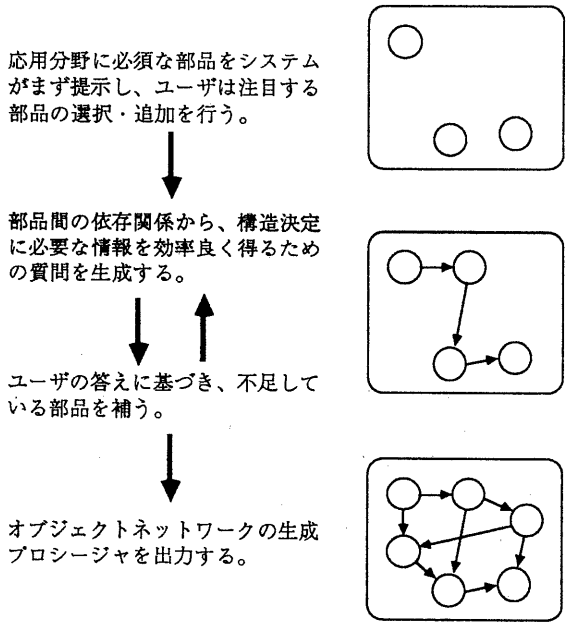


図 1: 処理の概要

4 依存関係

本章では、AGON が用いる質問列生成のための依存関係について述べる。

4.1 クラス間の依存関係

オブジェクト指向では、オブジェクトへのアクセスはメッセージパッシングによってのみ行われる。そのため、呼出す部品を選択する基準は、どんな機能のメソッドを保持しているかということのみである。したがって、クラス間の依存関係は、そのクラスのメソッド内でのインスタンス変数へのメッセージ受渡し状況を調べ、それを受け取り得るクラスを探すことで得られる。

選ばれるべき適切な部品という基準から考えれば、インスタンス変数に保持されたオブジェクトは、送られてくるメッセージ全てを受けとらなければならない (and 制約)。また、そのオブジェクトのクラスは、それらのメッセージ全てを受け取ることのできるクラスのうちいずれかである (or 制約)。

これにより、図2のようなクラス間の依存関係のand-or木ができる。

図2において、□はクラス、○はメソッド、実線はインスタンス変数、破線はその候補を表わす。この例では、Aクラスはインスタンス変数ia1,ia2をオブジェクトとして持ち、そのそれぞれにe1,e2とc1というメッセージを送り出している。そして、e1,e2というメソッドを持っているクラスの候補として、E,E'クラス、c1というメソッドを持っているクラスの候補として、C,C'クラスがあげられている。

なお、図3(a)のようなクラス階層がある時、図2の破線で囲まれた部分のように、d1,d2というメソッドを持つ候補クラスとしては、W,X,Zクラスが挙げられる。図3(a)では、Yクラスもメソッドd1,d2を持つにもかかわらず候補にあげられていない。これは、メソッドd1,d2だけを使用するという立場にたてば、YクラスはWクラスとなんら変らないからである。

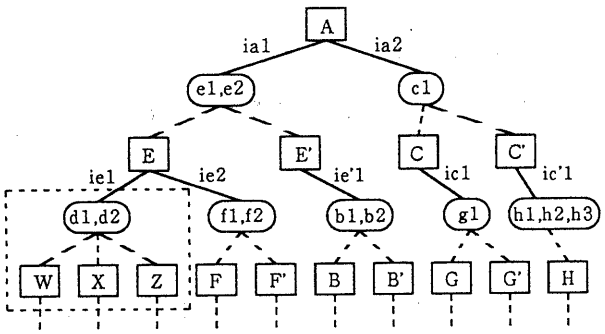


図2: クラス間の依存関係

4.2 メソッド間の連携関係

一般的に同一クラス内のメソッドは、インスタンス変数へのアクセスを通して連係している。例えば、一方のメソッドで代入された値を他方のメソッドで参照するとき、それらのメソッド間には連係関係が存在する。

また、異なるクラス内のメソッド間においても、互いのクラスのインスタンスがそのインスタンス変数に共通のオブジェクトを保持している場合、そのオブジェクトへアクセスしているメソッド間に連係関係が存在する。

5 質問列生成のための知識

本章では、AGONが用いる質問列生成のための知識について述べる。

5.1 メソッド機能に関する知識

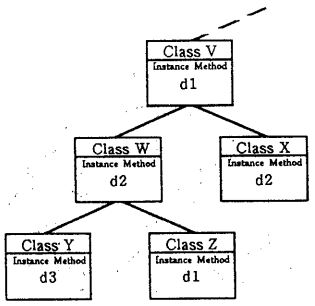
前章でも述べたように、ユーザはメソッドの機能によって部品を選択する。従って、ユーザが求める機能を持つクラスを容易に選択できるようにするためには、質問の中でクラス間のメソッドの機能差を明確に示すことが必要である。

AGONでは、クラス間のメソッドの機能差を自然言語で明確に示すために、メソッドの機能についての知識を、クラス階層に対応する知識階層に、以下の書式で自然言語によって記述する。

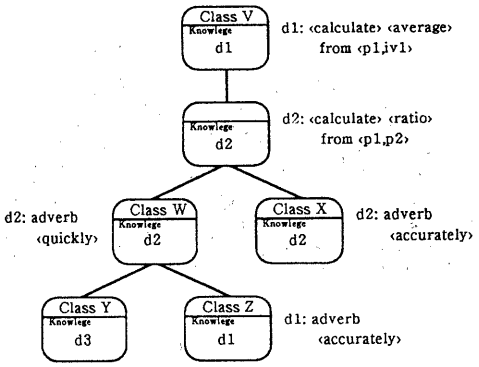
< 動詞 > < 目的語 >
from < *variables* > < 副詞 >

知識階層では、クラス階層の場合と同様に、継承機能が適用されるため、スーパークラスでの記述と異なる項目のみを記述すればよい。

図3にその具体例を示す。図3(a)のようなクラス階層があり、その中でd1,d2,d3というメソッドが定義されているとき、それに対応する知識階層は図3(b)のようになる。Vクラスに対応する知識としてはメソッドd1についての機能説明が与えられ、メソッドd2のように枝分れして2つのクラスに跨がる場合には、集約されたノードにそれらの共通部分に係わる知識が定義され、異なる部分だけが個々のノードに定義される。この例では、Wクラスに対応する知識には「高速に」、Xクラスに対応する知識には「高精度で」という副詞だけが記述されている。また、Zクラスでメソッドd1がオーバライドされているが、この場合も、Vクラスのメソッドd1との機能差である「高精度で」という副詞だけが、Zクラスに対応する知識として記述される。



(a) クラス階層



(b) 知識階層

図 3: メソッド機能に関する知識

5.2 呼出せる部品に関する知識

1章で述べたように、グローバル変数を多用することは、再利用性を低下させてしまう。再利用という観点からみると、理想的には、すべての複合オブジェクトが、他のクラスのインスタンスをその構成要素としてインスタンス変数に保持することが望ましい。AGONは、このような理想的なオブジェクト指向プログラミングを前提にしている。また、複合オブジェクトは、その作成時に、インスタンス変数に持つ構成要素がある機能を持つことを想定して作られている。

AGONでは、複合オブジェクトの構成要素として用いられるインスタンス変数を規定するために、そのインスタンス変数のドメインを知識としてリストの形で登録できる。また、複合オブジェクト内でのインスタンス変数(オブジェクト)の役割を自然言語により記述する。ドメインは部品の適切な動作の保証と依存関係を求めるために用いられ、役割は質問の中で、そのオブジェクトを指し示す際に用いられる。

5.3 応用分野に関する知識

プログラムの応用分野を階層的に定義する。すなわち、応用分野に関する知識についても、各分野での必須な部品を差分的に登録する。この知識によって、初期のユーザの部品選択の負担を軽減し、ユーザは注目する部品の選択のみに専心することができる。

6 質問列の生成

本章では、前章までに述べた依存関係と知識を用いて、どのようにして質問列を生成するかについて述べる。質問列の生成手順を図4に示す。

AGONでは、3章で述べたように、まず始めに、これから作成するオブジェクトの応用分野がユーザによって指定され、システムがその応用分野における必須部品を知識を用いて選択する。そして、ユーザは必要に応じて部品の追加あるいは削除を行う。

AGONは、このようにして選ばれた部品について、知識を用いて依存関係の木を作成する。質問列の生成は、これらの依存関係の木において、同じクラスが複数の依存関係の木に出現する場合、すなわち、依存関係の木が重なる場合と重ならない場合に分けて行なう。

6.1 依存関係の木が重なる場合について

依存関係の木が重なる場合については、さらにその重なり方によって、図5に示すように、以下の2通りに分けられる。

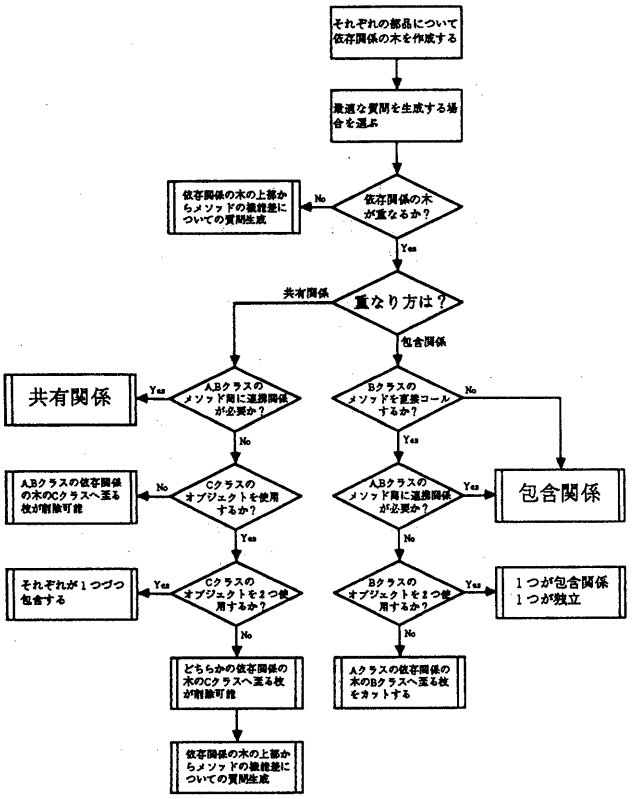


図 4: 質問列の生成

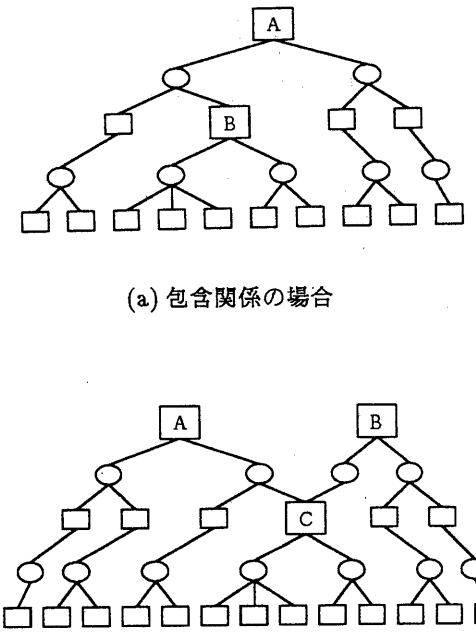


図 5: 依存関係の木の重なり

1. 包含関係の場合

図 5(a) のように、包含するクラスを A クラス、包含されるクラスを B クラスとする。

まず、B クラスのメソッドをユーザが直接呼出すかどうかを問い合せる。この質問は、ユーザがどのような目的で B クラスを選択したのか、すなわち、直接呼出すために選択したのか、他の部品の構成要素として選択したのかという意図を探るものである。

この質問にユーザが No と答えた場合、A クラスのオブジェクトが B クラスのオブジェクトを実際に包含することが決定される。Yes と答えた場合、システムは A クラスと B クラスとが仮に包含関係となった場合、A クラス内のユーザが使用するメソッドと B クラス内のユーザが使用するメソッド間に、実際に連係関係が存在するかどうか調べた上で、ユーザにメソッド間の連係関係が必要であるかどうかを問い合せる。この質問によって、システムは A クラスと B クラスが実際に包含関係となるべきであるかどうかを決定する。

すなわち、Yes と答えた場合、A クラスのオブジェクトが B クラスのオブジェクトを包含することが決定され、No と答えた場合は、さらに、B クラスのオブジェクトを 2 つ使用するかどうかを問い合せる。この質問に対して No であれば、A クラスの依存関係の木の B クラスへ至る枝をすべてカットし、Yes であれば、B クラスの 1 つのオブジェクトは A クラスのオブジェクトに包含され、もう 1 つのオブジェクトはユーザが直接呼出すために用いられる。

2. 共有関係の場合

図 5(b) のように、A クラスの依存関係の木

とBクラスの依存関係の木が、Cクラスにおいて重なっているとする。

まず、システムはCクラスが共有されるとき、Aクラスのメソッドと Bクラスのメソッドとの間に、実際に連係関係が存在するかどうか調べた上で、ユーザにメソッド間の連係関係が必要であるかどうかを問い合せる。

この質問に Yes と答えた場合、Cクラスの共有が決定され、No と答えた場合、Cクラスのオブジェクトを使用するかどうか問い合せる。この質問に対して No であれば、A,B クラスの依存関係の木のCクラスへ至る枝をすべてカットし、Yes であれば、さらに、Cクラスのオブジェクトを2つ使用するかどうか問い合せる。

これに対してユーザが Yes と答えれば、A,B クラスが共にCクラスのオブジェクトを独立に持ち、No であれば、少なくともどちらか一方のCクラスへ至る枝を削除できるので、A,B クラスのいずれかの依存関係の木の上部からメソッドの機能差について質問を生成し、Cクラスへ至るどちらかの枝を削除する。

6.2 依存関係の木が重ならない場合について

依存関係の木が互に重ならない場合には、依存関係の木の上部の or 分岐から順に、メソッドの機能に関する知識をもとに、その機能の差を抽出し、どの機能のメソッドを使用するかについての質問を生成する。

6.3 質問の順序

AGON では、前節の依存関係の木が重なる場合を優先的に質問を生成してゆく。しかし、一般にはこのような重なりは同時に複数存在すると考えられる。その中で、どの質問からユーザに問い合せるかについて、AGON では、各質問ごとに、その質問によって削除できる候補の期待値を求め、その期待値の最も大きい質問から優先的に問い合せる。このことによって、ユーザの求める構造を最も効率良く引出すことができる。

7 まとめ

本研究では、オブジェクト指向環境における生産性の向上を目標に、直接的生産性と再利用性の2つの観点から考察し、プログラム設計時に生じる曖昧な要求からの構造設計と部品選択、また、それにともなう部品検索や構造生成のためのコーディングなどのプログラミングにおいて本質的でない作業のそれぞれを支援する環境を実現した。

この環境では、プログラマは既存の部品を適切に選択でき、また、メソッド定義などの本質的な作業にのみ従事することができる。その結果、直接的生産性と再利用性の両面から、生産性を向上させることができる。

今後の検討課題として、

- 型付き言語への応用
- 質問列の多様化
- 実験的検証

  どの位の規模のクラス階層で、どの位の規模の構造を得るために、どの位の質問列が必要か？

などがあげられる。

参考文献

[1] A. Goldberg and D. Robson, "Smalltalk-80 -The Language and It's Implementation-," *Addition-Wesley*, 1983.

[2] B. Meyer, "Object-Oriented Software Construction," *Prentice-Hall*, 1988.

[3] 古宮, "部品合成による自動プログラミング・システム PAPS-プログラミングの知識とその使い方について-," 電子情報通信学会, 信学技報, vol.88, No.176, pp.19–26, 1988.

[4] 岡本, 橋本, "制約指向の概念モデルを用いた高次部品化によるプログラム合成" 電子情報通信学会, 信学技報, vol.89, No.276, pp.1–10, 1989.

[5] G. Fischer, A. Girgensohn, K. Nakakoji, and D. Redmiles, "Supporting Software Designers with Integrated Domain-Oriented Design Environments," *IEEE Trans. Software Engineering*, vol.18, No.6, 1992.

[6] G. S. Novak,Jr., F. N. Hill, M. Wan, and B. G. Sayrs, "Negotiated Interfaces for Software Reuse," *IEEE Trans. Software Engineering*, vol.18, No.7, 1992.

[7] 水野, 重永, "部品合成による日本語からのLisp 関数自動生成," 情報処理学会研究報告, 90-AI-69, vol.90, No.26, 1990.