階層構造オブジェクト・エディタの試作 *

4C－1

畠山 正行, 花村 啓介 †
茨城大学工学部情報工学科 ‡
{ masayuki, st94073} @cis.ibaraki.ac.jp

1 はじめに

データ構造とメソッドをカプセル化したオブジェクトをプログラミングの最も基本的な単位 [1] としたオブジェクト指向言語が普及し, オブジェクト指向パラダイムに基づいてプログラムの設計から実装が数多く行なわれる様になってきている. しかしその一方で, そのオブジェクトの汎化·特化階層及び, 集約階層, オブジェクト間の関連の複雑化が問題となってくる. それにともない, プログラムソースコードの可読性の悪化及び, エディットの煩雑化, オブジェクトの再利用性の低下がおこり, 結果としてソフトウェアそのものの品質並びに, メンテナンス性の悪化を招くことも考えられる.

そこで本研究では, プログラミング言語C++をその対象言語とし, オブジェクトの汎化·特化階層, 集約階層, 関連などをGUI(Graphical User Interface) 上に明確に表現し, その上でオブジェクトの編集·管理ができる階層構造オブジェクト·エディタを作成し, ソフトウェアの実装支援の面から, それらの問題を解決することを目的とする.

本研究の対象言語であるC++言語は、対象世界にオブジェクト以外の構成要素も含む不純オブジェクトモデル (impure object model) [2] であり, その言語仕様に、基となったC言語の構造や実行効率の低下を防ぐ為, 大域関数やクラスの公開派生などの様にオブジェクト指向言語として不純な部分を多く含んでいる. それにより, C++言語は多くのC言語のソースやライブラリを利用できるなどの汎用性·柔軟性と引き替えに—あるいは, それゆえに— 複雑性を持った言語となっている. さらに, オブジェクト指向パラダイムに基づいて設計されたソフトウェアをC++言語を用いて実装する段階において, 本来の純粋なオブジェクト指向パラダイムに基づいた設計を, C++言語に適応した不純なオブジェクト指向パラダイムによるかたちに修正せざるを得ない部分が存在する.

そこで本研究では, 階層構造オブジェクト·エディタを用いて, C++言語のオブジェクト指向として不純な点を一部隠蔽あるいは, 補完し, C++言語による実装をGUI上に純粋なオブジェクト指向として表現することも考える.

2 階層構造オブジェクト·エディタの概要

第1節で述べた目的を達成するため, 階層構造オブジェクト·エディタの機能として, 以下の4つの機能（図1）が必要となると考えた.

1. GUIパート
2. プログラム構造解析パート
3. ソースコード·ジェネレータ·パート
4. 構文解析パート

GUIパートは, 階層構造オブジェクト·エディタとユーザ間の接点となるパートであり, 階層構造オブジェクト·エディタのあらゆる操作は, このGUIパート上で行なわれ, ユーザに対するあらゆるメッセージも, このGUIパートを通して表示される. このGUIパート上でユーザは, プログラムの構造をブラウジングしながら変更することができる. これは, マウスなどのポインティング·デヴァイスを主体とした操作で, プログラムの構造を確認しつつ, プログラムにオブジェクトの追加並びに, 削除, 変更が可能であることを意味する.

プログラム構造解析パートは, 階層構造オブジェクト·エディタの核となるパートである. GUIパート上で編集されるプログラムは, プログラム構造解析パートによりメモリ上にそのデータを保持され, その構造を解析され, それに対する変更は常に監視される. GUIパート上でプログラムに変更が加えられると, 構造解析パートにその変更が伝えられる. そのとき, プログラムに矛盾点を生じるような変更が加えられた場合は, プログラム構造解析パートからGUIパートを通じてユーザに対し矛盾点の変更を促す.

プログラム構造解析パートに保持されているプログラムのデータは, C++言語のソースコードにはなっておらず, また, 編集過程でファイルとして保存する場合もC++ソースファイルではなく,

*Hierarchically Structured Object Editor; A Prototype
†Masayuki Hatakeyama, Keisuke Hanamura
‡Department of Computer and Information Sciences, Ibaraki University, 4-12-1, Nakanarusawa, Hitachi-city, 316-8511 Japan.

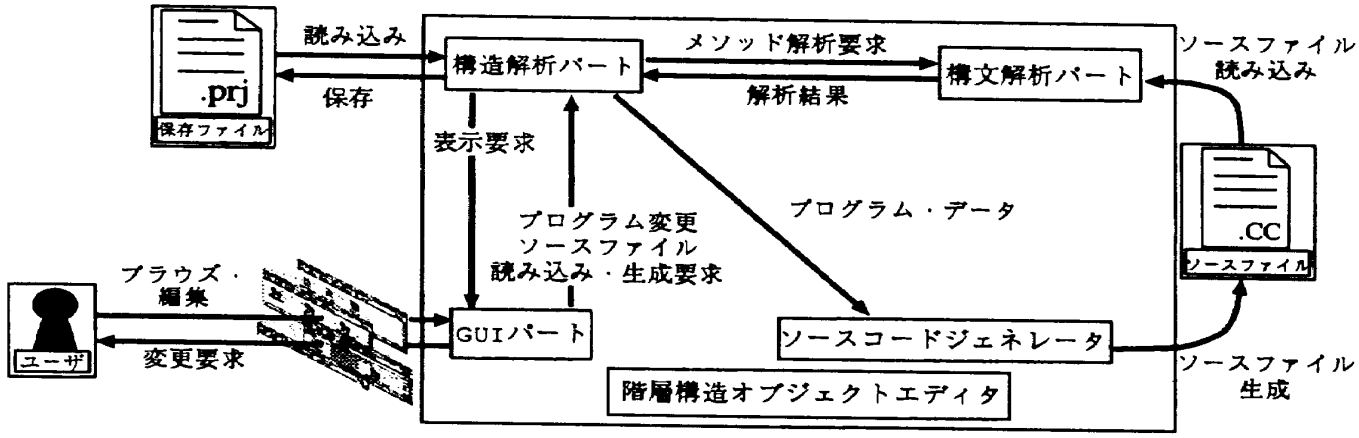

図 1: 本研究のエディタの主な機能集合とその関係

通常, 階層構造オブジェクト・エディタの独自のバイナリ・ファイルとして保存される（図 1 の左上の.prj ファイル）. そこで, C++ソースファイルとしてプログラムを出力する場合には, ソースコード・ジェネレータ・パートを使い C++言語のソースコードを作成し, 出力する.

C++言語のオブジェクト指向として不純な点を隠蔽する為, GUI パート上では全ての変数や一部のヘッダファイルなどが全てオブジェクトとして表現され, 変数に対する演算やヘッダファイル内で宣言された関数などをメソッドとして扱う. しかし, これらを C++ソースファイルに変換する場合には, C++言語の仕様もしくは実行効率などを考えて変数や関数として出力する必要がでてくる. そこでこの GUI パート上で編集された情報を基に, ソースコード・ジェネレータ・パートは GUI パートで一部のメソッドやクラスを自動生成する. また逆に, オブジェクトとして GUI パート上で表現されていたものを, 変数や関数に変換することも行なう. 例としては、GUI パート上でクラスのインスタンスとして扱われていた変数をソースコードに出力する際に、通常はただの変数として出力されるが、その変数クラスのメソッドがオーバーロードされていたり、他のクラスのスーパークラスとなっている場合は、クラスとして生成される。

階層構造オブジェクト・エディタでは, メソッドの実装部分に関してはテキスト・エディタを用いて作成・編集する. このとき, メソッドの実装部分とそれに関連する他の部分との整合性を調べる為に, C++の構文を解析する必要がでてくる. そこで構文解析パートを用いて, C++言語の構文を解析し, その結果をもとにプログラム構造解析パートによって, その構造・整合性などが解析される. また, 構文解析パートは既存の C++ソース・ファイルを読み込み, 利用する際にも, その構文を解析する為に用いられる.

3 本研究の現状

本研究は現在, 階層構造オブジェクト・エディタの設計段階を終了し, GUI パート並びにプログラム構造解析パート, ソースコード・ジェネレータ・パートの 3 つのパートの実装が C++言語を用いて進められている. ソースコード・ジェネレータ・パートに関しては、メソッドやクラスの自動生成が無い、ごく単純な構造を持つプログラムに関してはソースコードを生成できる。しかし、それ以上のソースコードの生成に関しては、GUI パートとプログラム構造解析パートの試用が可能となってからとなる。また現段階では構文解析パートに関しては、実装は行なわれていない。よって、メソッドの実装部の記述の監視やソースファイルの読み込みなどはまだである。

4 おわりに

本稿では, 階層構造オブジェクト・エディタの設計について述べてきたが, 現段階では, まだ解決すべき問題が多く残されている. 例えば, C++ソース・ファイルからの読み込みによって生じる, 階層構造オブジェクト・エディタとの表現の不一致, デバッグの難解性などがあげられる. 今後はこれらの問題を解決したい.

参考文献

[1] 春木 良且 「オブジェクト指向への招待」近代科学社, 1995

[2] 所 真理雄, 松岡 聡, 垂水 浩幸 編 「オブジェクト指向コンピューティング」岩波コンピュータサイエンス（岩波書店）, 1993