不完全なソースコードからのプログラム情報の抽出

2G－6

大島 満 *　中村 宏明*　安田 和 *
* 日本アイ・ビー・エム (株) 東京基礎研究所

1　はじめに

オブジェクト指向プログラミングの一つの利点として、クラスを部品とする再利用があげられる。これらクラスライブラリの開発時において、クラスの設計とクラスの実装が同時に起こる場合が多く、この設計／実装のサイクルが開発時間にしめる割合が多くなりやすい。したがって、プログラム情報を抽出しユーザーが理解しやすい形で提供することは非常に重要である [1] [2]。我々はこのプログラム情報をデータベースの形で蓄え [3] [4]、それに対するユーザーインターフェイスを用意することにより、より自由度の高いシステムを実現している。

本稿では、不完全なC++のソースコードからの情報の抽出を行ない、• プログラムの解析の高速化、• 開発途中の不完全なプログラムの解析、の可能性について考察する。

2　不完全なソースコード解析の必要性

オブジェクト指向プログラムの開発では、クラスの設計とクラスのコーディングを行き来することが多いため、設計からプログラムという方向の支援だけではなく、プログラムから設計情報という逆向きの情報を得ることも重要である。とくに、フレームワーク [5] などのクラスライブラリを構成する場合は、クラスの階層だけではなく、クラス間の関係の情報も有効である。

プログラムを抽出する方法の一つとして、コンパイルの構文解析プロセスを用いて正確な情報をとる方法が考えられるが、一般的に長い時間がかかる。したがって、ユーザーに対する情報のフィードバックの遅れが生じてユーザビリティが低下する、さらに開発途中の不完全なプログラムを扱うことができないなど、プログラム開発環境として適当でない部分が生じる。

時間がかかる主な原因としては、コンパイルと同等のプロセスを必要とするため

- ユーザが望まない情報 (たとえば osstream class など) まで解析してしまう。
- 複数のコンパイル単位が include の指定によって重複する同じファイルを何度も解析してしまう。

などが考えられる。1における不必要なインクルードを避けようとすると、コンパイル単位での情報が不完全なものになり、必然的に不完全なコードの解析が必要となる。

そこで我々は不完全なプログラムを解析できるParser(以下Fuzzy-Parser) を用いてデータベースに格納する方法をとった。これによって

- 不完全なソースコードの解析が可能
- プリプロセスのインクルード/マクロ展開を避けることによる高速化
- 曖昧だが速く解析する方法と、時間がかかるが正確な情報を扱う方法の両方が可能

などが期待できる。

3　システム構成

システムの構成を図1に示す。正確な情報はコンパイラ、高速に解析した情報は Fuzzy-Parser によって生成され、データベースに書き込まれる。Fuzzy-Parser は不完全なプログラムを解析できるようになっており、これによって include/Macro 展開を行なわないことが可能になっている。結果として Fuzzy-Parser が抽出可能な情報は限られており、今のところ・クラス階層・メンバー関数／変数とその属性のみである。型の正確な解析は本質的に困難と判断し、現在は行なっていない。型を知りたい場合は、シンボル位置のソースを参照することによって、間接的に知ることができるが、型のサポートは今後の課題である。また nested class の一部はサポートしているが、template については現在検討中である。

データの書き込みには、PDB stream と呼ぶ統一されたインターフェイスを用いるので、データベースは Fuzzy-Parser を意識する必要はほとんどない。(プログラムデータベースについては本稿では詳しく述べないので、文献 [4] を参照されたい) データベースには正確

Program Information Retreval for incomplete source codes.

Mitsuru Oshima * , Hiroaki Nakamura † and Kasu Yasuda*

* IBM Research, Tokyo Research Laboratory

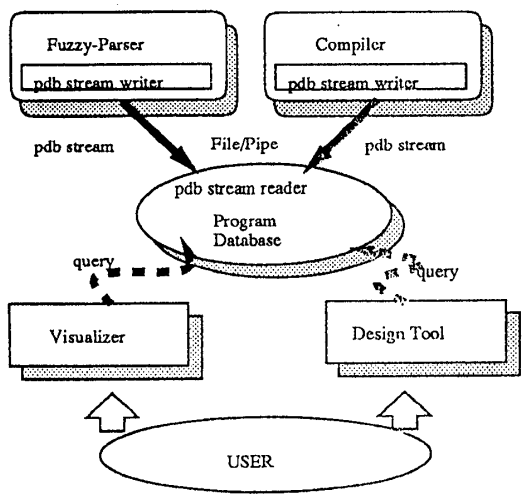

図 1: システム構成

表 1: 実験結果

方法	コンパイル 時間	読み込み時間
Compiler Base	1972.5 sed	479.77 sec
Fuzzy-Parser		15.18 sec

な情報も格納可能なので、必要に応じて正確な方法と、速い方法を選ぶことができる。

4 実験結果

先に述べたシステムを用いて、プログラム解析の実験を行なった。実験対象としたのは、268 ファイル・25962 行・562158 bytes のソースコードである。結果を表 1 に示す。コンパイル時間は、コンパイラがデータファイルを書き出すのにかかった時間、読み込み時間は、吐き出されたデータファイルをデータベースが読み込むのにかかった時間である。Fuzzy-Parser は情報抽出とデータベースに書き込むのを同時に行なっているため、両方の時間を合わせたものである。一度ファイルシステムに書き出す必要がないぶんを差し引いても、十分速いことがわかる。もちろん、抽出しているデータの数・種類ともに少ないが、クラス階層や、メンバーなどはほぼ正確な情報をとることができた。間違いの数・頻度や、不完全なソースコードを読み込んだ場合についてのデータは、現在採取中である。

5 おわりに

不完全なプログラムの解析がプログラム開発の過程で必要であり、情報の抽出が可能であることを示した。関連研究として、Sniff [2] でも同様の手法が行なわれているが、

- 不正確な情報しか扱えない
- 取り出した情報は環境内で閉じており、情報の再利用（たとえば設計に使用するなど）が難しい

などの問題がある。また Richard Helm [1] らはクラスライブラリの解析に IR を用いてドキュメントからの情報抽出を行なっており、ソースコードの情報をそれほど多く必要としないが、クラスライブラリのドキュメントを必要とするため、開発・設計に適用するには問題が多い。

現在得られる情報はクラスの階層やメンバーの情報だけであり、まだ開発には十分ではなく、クラス間にある関係 [6] を知ることが重要になる。今後はより詳しい情報を開発時の負担にらない時間でとり出す手法を研究する必要がある。

参考文献

[1] R. Helm and Y. S. Maarek : Integrating Information Retrieval and Domain Specific Approaches for Browsing and Retrieval in Object-Oriented Class Libraries, OOPSLA'91, pp. 47-61

[2] W. R. Bischofberger : Sniff - A Pragmatic Approach to C++ Programming Environment, C++ Technical Conference, Proceedings, 1992

[3] 中村, 安田, 大平, 三ッ井 : オブジェクト指向ソフトウエア開発におけるプログラム理解支援, 情報処理学会研究会報告,94-PL-4,1994

[4] 安田, 三ッ井, 中村, S. Javey : C++プログラム・データベース構築, 情報処理学会研究会報告,94-SE-18,1994

[5] R. E. Jonson and B. Foote: Designing Reusable Classes, JOOP, June/July, 1988

[6] R. Helm, I. M. Holland and D. Gangopadlhay : Contracts: Specifying Behavioral Compositions in Object-Oriented Systems, ECOOP/OOPSLA '90 Proceedings, October 21-25, 1990