## ソースコード解析ツール開発支援システムの試用

高田　智規[†]　　佐藤　慎一[†]
飯田　元[††]　　井上　克郎[†]（正員）

Trial on Development Support Systems of Dependence Analysis Tools

Tomonori TAKADA[†], Shinichi SATO[†], Hajimu IIDA[††], *Nonmembers*, and Katsuro INOUE[†], *Member*

[†] 大阪大学大学院基礎工学研究科，豊中市
Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 560 Japan
[††] 奈良先端科学技術大学院大学，生駒市
Nara Institute of Science and Technology, Ikoma-shi, 630-01 Japan

**あらまし**　ソースコードの解析を行うツールの作成を支援するシステムが利用できるようになってきた．本論文ではそのようなシステムの概要の調査を行う．また，開発にシステムを利用したツールと利用しなかったツールの比較を行う．その結果この種のシステムを用いない場合に比べ，少ない労力でツールが作成できることがわかった．

**キーワード**　Sapid，REFINE，依存関係解析

### 1. まえがき

ソースコード中の変数や文の間の依存関係を調べたりスライス[1]を抽出したりするといういわゆるソースコード解析ツール（ツールと略す）は通常，ソースコードの構文解析，変数や文の依存性の計算，プログラム依存グラフ（PDGと略す）の構築，PDGの表示などを順に行う必要がある．一般にそれらの機能すべてをC言語などで記述するのは容易ではない．

最近，Sapid [2]，REFINE [3]といったソースコード解析ツール開発支援システムが利用できるようになってきている．これらのシステムではツールに必要な基本的な機能を用意しているため，ツールの作成が容易になることが期待される．

本論文では，これらのシステムの機能の概要を調べる．また，実際にREFINEを用いPDGの表示やスライスの計算を行うツールを試作し，その有効性について述べる．

### 2. 開発支援システムの概要

ソースコード解析ツールの開発支援システムとしてSapidとREFINEを取り上げ，依存関係の解析などに有効な機能について述べる．

また，SapidとREFINEの特徴を表1に示す．

[Sapid]　Sapidは，Cで記述されたソースコードの構文解析および意味の一部の解析を行い，その結果（変数の定義，使用などの情報）をデータベースとして出力する．従って構文解析部および意味解析部の一部を作成する必要はなく，依存関係解析を行う際に非常に有用である．

解析結果を図やグラフとして表示する際には外部プログラムが必要である．

相互運用性は通常のCプログラムと同程度である．また，ライブラリを利用することにより，CFG（Control Flow Graph）やPDGなどを作成することも可能である．

[REFINE]　REFINEの構文解析サブシステムでは，構文の定義を与えることにより種々の言語に適用できる．また，C，COBOL，Pascal（サブセット）の構文定義はあらかじめ用意されている．

REFINE言語インタプリタは，木や集合などさまざまなデータ型をもち，それらに対し豊富な種類の演算を行うことができる．図形表示インタフェースでは，木の画面上への配置やマウス操作定義などを行うことができる．

REFINE上で開発したプログラムはREFINE言語インタプリタ上でのみ実行可能であるため，相互運用性はあまり良くない．また，REFINE付属の解析ツール（CやCOBOLを対象）は利用・拡張することによりプログラムの解析を行うことができる[4]が，依存解析などのPDGの計算に必要な解析までは行っていない．

表1　開発支援システムの比較
Table 1　Comparison of development systems.

| | Sapid | REFINE |
|---|---|---|
| 対象 | C | C, COBOL 等 |
| 解析 | 構文・意味の一部 | 構文 |
| 結果 | データベース | 木構造 |
| UI | 外部（athena,Tk） | 内部（INTERVISTA） |
| 相互運用性 | 高（通常のC程度） | 低（REFINE上でのみ実行） |

### 3. ツールの試作

この種のシステムの有効性を調べるために，REFINEを用いてツール（ツールRと呼ぶ）を試作した．

ツールRはPascalのサブセット(注1)で記述されたソースコードを読み込み，そのPDGを表示する．表示されたグラフの節点をマウスで指定することにより，スライスの抽出を行い，その結果をもとのグラフのサブグラフとして表示する．

(注1)：変数はスカラ型のみで，条件文（if），代入文，繰返し文（while），入力文（readln），出力文（writeln），手続き呼出し文，複合文（begin-end）からなる．

表2 ツールRとツールCの比較
Table 2 Comparison of tool R and tool C.

| | ツールR | ツールC |
|---|---|---|
| 構文解析部（行） | [730] | 900 (yacc,lex) |
| PDG構築部（行） | 910（REFINE言語） | 2710 (C) |
| 表示部（行） | 570（REFINE言語） | 460 (Tcl/Tk) |
| 構文解析（秒） | 5 | 0.11 |
| PDG構築（秒） | 10 | 0.87 |
| PDG配置・表示（秒） | 45 | – |
| 作成期間（月） | 1.5 | 6 |

PDGの構築およびスライスの計算には，再帰を含むプログラムを対象としたアルゴリズム[5]を用いた．

ツールRを以前作成したツール[6]（ツールCと呼ぶ）と比較した（表2）．ツールCはツールRと同一のPascalのサブセットを入力とし，PDGの構築，スライスの計算等を行う（但し，UIは多少異なる）．なお，両ツールともSun SPARCstation 10（メモリ64MB）上で実行し，実行時間の比較には約500行の同一プログラムを入力として与えた．

ツールRの構文解析部はREFINEが提供するものを用いている．また，表示部ではPDGの配置，表示およびマウス操作を記述している．PDGの配置には，REFINE言語の木を配置する関数を利用した．一方，ツールCの構文解析部はyacc, lexを用いている．また，表示部はウィンドウ上にテキストの表示を行う．

PDG構築部の大きさはツールRの方が大幅に小さい．これは，REFINE言語のもつさまざまなデータ型や関数を利用したためだと考えられる．

実行時間はツールCが短くなっている．これは，REFINE処理系がインタプリタ型であることが一つの要因であると考えられる．更に，ツールRではシステムの起動，サブシステムの読込みに約2分要した．

作成期間はツールRが短くなっている．これは，記述量と同様にREFINE言語のデータ型・関数を利用したためだと考えられる．但し，REFINE言語およびREFINEシステムの学習には数か月要した．

## 4. むすび

ソースコード解析ツールの開発支援システムとしてREFINEとSapidを取り上げ，PDGの計算やその結果出力といった用途に有効な機能について調査した．

また，これらのシステムの有効性を評価するためREFINEを用いてツールを作成し，C言語で記述されたツールとの比較を行った．その結果，REFINEを利用することにより記述量や作業時間を小さくすることができた．またツールの出力も，C言語などでは実現が困難である見やすいグラフィックスにすることが容易にできた．一方，システムの起動や実行にやや時間がかかることがわかった．

以上から，これらのシステムはソースコード解析ツールの試作やその評価を効率良く行うのに有効であると考えられる．

## 文献

[1] M. Weiser, "Program slicing," Proceedings of the Fifth International Conference on Software Engineering, San Diego, CA, pp.439–449, 1981.
[2] 山本晋一郎，阿草清滋，"細粒度リポジトリに基づいたツール・プラットフォームとその応用，" 情処学ソフトウェア工学研報，vol.102, no.7, pp.37–42, 1995.
[3] "REFINE User's Guide," Reasoning Systems, 1990.
[4] L. Markosian, P. Newcomb, R. Brand, S. Burson, and T. Kitzmiller, "Using an enabling technology to reengineer legacy systems," Communications of the ACM, vol.37, no.5, pp.58–70, 1994.
[5] 植田良一，練 林，井上克郎，鳥居宏次，"再帰を含むプログラムのスライス計算法，" 信学論（D-I），vol.J78-D-I, no.1, pp.11–22, 1995.
[6] 佐藤慎一，飯田 元，井上克郎，"プログラムの依存関係解析に基づくデバッグ支援システムの試作，" 情処学論，vol.37, no.4, 1996.
[7] D. Atkinson and W. Griswold, "The design of whole-program analysis tools," in Proceedings of 18th International Conference on Software Engineering, pp.16–27, Berlin, March 1996.

（平成8年4月30日受付，9月24日再受付）