UDC 681.3

ソフトウェア保守支援ツールセット

Software Maintenance Support Tool Set

野口 正浩*(1) Masahiro NOGUCHI
木村 博昭*(2) Hiroaki KIMURA

抄 録

開発したソフトウェア保守支援ツールセットは，プログラムの保守性向上支援及び修正作業支援のための情報提供を行い，企業のシステム部門にとって重要な課題であるソフトウェア保守業務の効率化を目指すものであり，その考え方，機能，システム構成について紹介した。これは，最新のコンパイラ技術をベースに独自のアルゴリズムを加えた解析モジュールをコアとしており，開発時の品質の作り込みから，プログラム理解や修正時の影響範囲解析に至るまで，一貫した支援を行えるツール群となっている。また，複数の手続き型言語へ比較的容易に対応できるアーキテクチャであることも特徴である。

Abstract

The software maintenance support tool set developed by authors provides software engineers with the information that helps program maintainability enhancement and repair activities, rendering software maintenance more efficient, which is one of the most important topics to the organizations responsible for information systems in enterprises.This tool set includes program analysis engines based on the most recent compiler technology with our own original algorithms, providing integrated maintenance aid for quality built-in in development, and program comprehension and impact analysis during program repair.Further, the architecture of the tool set is designed such that it can be tailored for multiple procedural languages with relatively small amount of efforts.

1. 緒 言

近年，西暦2000年問題を契機に，ソフトウェア保守業務の効率化に関する世の中の関心が急速に高まっている。著者らは，このようなニーズを受けて，ソフトウェア保守業務を支援するツールセットとしてQuality enhancement by Static testing (以後，QSと略記する)を開発した。はじめにソフトウェア保守業務の計算機支援の考え方について述べ，それから，QSの主な機能とそのシステム構成について紹介する。

2. ソフトウェア保守業務の計算機支援について

2.1 ソフトウェア保守の位置づけ

ソフトウェアに関するコストを考えるとき，その開発コストに目を向けがちである。しかしながら，ビジネスアプリケーションソフトウェアの寿命は，平均で約10年[1]と言われており，ライフサイクルコストで考えると，保守だけで40～75%程度を占めるというのが定説である[2]。このため，企業では，ソフトウェア資産の増大に伴い保守予算が開発予算を圧迫している場合が多く，大企業，機関の大半で，両者の割合は1：1程度になっている[3]と言われている。このような状況を踏まえると，保守業務の改善は，保守コスト抑制のための重要課題として捉えなければならない。

2.2 保守業務の計算機支援

保守業務の効率化は，保守担当者に対して，既存プログラムの理解を助ける手段と，与えられたプログラム修正をなるべく正しく行えるようにする手段を提供することによって達成できると考えられる。そのための計算機支援としては，開発時にプログラムの保守品質自体を向上させることを目的とした支援と，プログラム修正作業時の支援に大別できる。以下にこれらを概説する。

2.2.1 保守品質向上支援

保守業務効率化の第一歩は，保守対象となるプログラムの保守性を向上させること，すなわち，理解しやすく変更しやすいプログラムとなるように作り込むことである。そして，そのためには，まずプログラムの複雑さを小さく抑えるような設計を実施することが重要である。なぜなら，複雑なプログラムでは，読み手は一度に多くのことを考えたり記憶しなければその意味を理解することができず，また，データの流れが複雑に交錯するため，1個所の修正の影響範囲が大きく，影響が思わぬところに発現することも多いからで

*(1) エレクトロニクス・情報通信事業部
システム研究開発センター 主任研究員
神奈川県相模原市淵野辺5-10-1 〒229-8551 ☎(0427)68-6075

*(2) エレクトロニクス・情報通信事業部
システム研究開発センター 主任研究員

ある。プログラムの複雑さを表す指標としては，質的複雑さを表す凝集度や結合度[4]，量的複雑さを表すサイクロマティック数[5]などが提案されており，これらと保守性やプログラム劣化との関係についても多くの文献で指摘されている[3,6-9]。従って，これらの数値を機械的に計測，収集してソフトウェア開発プロセスにフィードバックすることにより，保守性の向上に役立てることが可能である。

次に，各ソフトウェア開発組織が持つプログラミング規約に則った開発を行うことも，保守品質上重要である。このような規約は，通常，保守性の向上を約束すべく十分に吟味されていることが多く，また組織で標準化された記述方法に従ったプログラムは，それに慣れ親しんだ人の理解を助ける。プログラムの規約への遵守度向上は，従来より人が直接プログラムを読んで検査することにより行われてきたが，規約違反の多くは機械的に発見することが原理的に可能であり，この面での計算機支援が期待される。

更に，プログラムは，修正を重ねるにつれて劣化，すなわち，無意味にプログラムの複雑さが増す現象が生ずる。その大きな原因の一つは，プログラムの修正を行う際に他人の書いたソースコード片の意図が正しく理解できず，それを取り除いた場合の他の部分への影響が明確に特定できない場合があることである。そのときには，当該コード片をとりあえずそのままにしておき，その部分を迂回するロジックを組込んだりするため，結果としてプログラム中に無駄なコードが残ってしまうことになる。このような冗長コードの検出支援も重要である。

2.2.2 プログラム修正作業支援

保守業務の主要な作業は，開発時に混入した欠陥の除去や，機能の部分的な向上，あるいは動作環境の変化への適合のためのプログラム修正である。典型的には，数千ページを超える膨大な設計文書をベースに，修正対象の現状機能理解，具体的な修正関連個所の洗い出し，修正部分の設計を行い，最後にプログラムそのものを解読して実際の修正を行う。その際に，通常は，開発者と保守担当者が異なるため，既存のプログラムの内容を理解することに非常に大きな作業負荷がかかっており，一般に保守業務の50%以上がプログラムを理解する作業で占められると言われている[10]。

プログラム理解にあたってはまず設計文書を参照するが，これらの文書は細かい点で実態と乖離している場合が多く，また，設計文書そのものが実質的に存在しない場合もあり，結局はプログラムを直接読んで理解する必要がある。その際には，プログラムの構造に関する情報やプログラム内で使用されている変数情報が最も頻繁に参照されることが実験的にも示されており[11]，これらをわかりやすい形式で保守担当者に示すことが計算機支援として重要である。

また，プログラムの修正の際には，他人の書いたプログラムの中から該当する修正個所とその修正の副次的な影響を正しく特定する必要があるが，これも非常に困難な作業である。従って，計算機により，細かいレベルでプログラム要素間の依存関係を解析し，修正の影響範囲を正しく提示することが期待される。

3. 保守支援ツールセットQS

3.1 QSの概要

QSは，プログラムを動作させずに行う解析，すなわち静的解析をベースに，上記に示した計算機によるソフトウェア保守支援を狙いとしたツールセットである。そして，開発時の品質の作り込みから保守時のプログラム理解，影響範囲解析に至るまで，一貫した支援を行えるようにツールをそろえている。また，解析結果については，情報の量と情報間の関連が多いため，ユーザにわかりやすく提示するためのGUIを備えている(3.3.1参照)。QSには，現在C言語用とFORTRAN用とがあり，ほぼ同等の機能を実現している。以下では，C言語用を例にとり，f2c[*1]といわれるフリーソフトウェア[*2]の解析結果の一部を画面例として示す。

3.2 各ツールの概要

3.2.1 保守品質作り込み支援ツール群

メトリクス計測器

本ツールは，サイクロマティック数，凝集度，結合度といった複雑さを表すメトリクスや，行数(コメント有無別)や語数等のサイズメトリクスを自動計測する。その結果は，一定の形式でファイルに格納され，品質基準値との比較や統計解析が可能である。

プログラミング規約遵守度検査器

本ツールは，対象となるプログラムが，新日本製鐵　エレクトロニクス・情報通信事業部システム研究開発センターで作成したC言語プログラミング規約で定められたスタイルを守っているかどうかを検査するものである。規約には，識別子の命名に関するもの，宣言方法に関するもの，型の整合性に関するもの，定数の使用方法に関するもの，演算子の使用方法に関するもの，制御構文に関するもの，データフロー[*3]に関するもの，及びコメントに関するものがあるが，その中で機械的に遵守度が判断できる約40種類の規約について検査を行う。また，データフローに関する検査では，データフローが異常となる実行パスを，異常を引き起こした変数とともに提示する。

本検査器は，エディタの中から起動できるようになっており，警告メッセージとソースコードの対応を容易にとることができる。また，警告に対するヘルプをWWWブラウザ上に表示させることができる。これらの画面例を図1に示す。

冗長性検査器

前述したように，冗長なコードはプログラムの修正作業を重ねたために生じた劣化の結果である。本ツールでは，手続きの出力に影響を与えない実行文と入力変数を，冗長コードとして検出可能である。これらのコードは，プログラムの実行結果に何ら影響を与えないが，プログラムを理解しにくくし保守性を悪化させるだけでなく，プログラム修正のたびに累積していく傾向がある。本ツールによって，このような冗長なコードを検出し，除去すれば，劣化の防止に役立てることができる。

3.2.2 プログラム修正作業支援ツール群

構造解析器

本ツールは，プログラムの構造をわかりやすく提示することにより，プログラムの理解を助けるもので，手続き間の呼び出し関係を表すコールグラフと手続き内の制御構造を表すフローグラフをグラフィカルに表示するとともに，ソースコードブラウザにソースコードを表示することができる。そして，これらの表示器間は互いに連動しており，グラフ内の各要素とソースコードの対応をマウス操作

---

*1 © 1990 AT&T Bell Laboratories and Bellcore.

*2 フリーソフトウェア：無償で配布されるソフトウェア。

*3 データフロー：プログラム内でのデータの定義～参照関係のこと。例えば，データの値が定義されないまま参照されたりする場合，これをデータフロー異常[12]と呼ぶ。

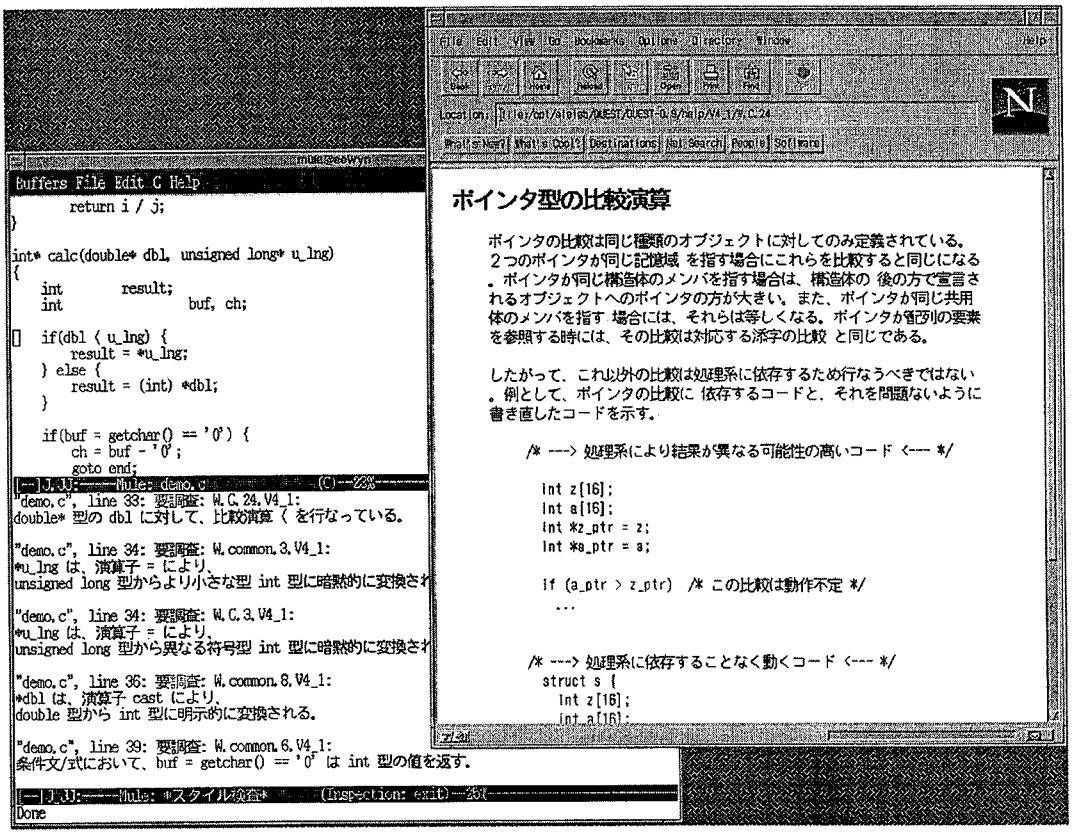


図1 プログラミング規約遵守度検査器の画面例[*4]

で容易に求めることができる。例えば，コールグラフの手続きノードをクリックすると当該手続きのフローグラフとソースコードが表示され，更にフローグラフのノードをクリックすると対応するソースコード部分の表示色が変わる。また，ソースコードの任意個所を指定するとフローグラフの対応ノードの表示色が変わるようになっている。

プログラムが複雑になると，グラフの表示画面も複雑になり容易に理解できなくなる。このため，グラフのレイアウトをユーザがマウス操作で自由に変えたり，関連した部分を切り出す(例えば，コールグラフ上で，ある手続きが呼び出している全手続きを再帰的に切り出す)機能も持っている。構造解析器の画面例を図2に示す。

保守文書生成器

本ツールは，プログラムを解析して得られた様々な情報をハイパーテキスト形式の文書として自動生成するものである。このツールによって生成された文書は，通常のWWWブラウザで参照できる形式になっており，プログラムを構成するファイル情報，手続き情報，変数情報，型情報に関するページをマウス操作でたどることができる。

図3(a)は，pushctlという手続きの入出力情報ページ例である。このページには，当該手続きの入出力変数(引数や大域変数)の情報へのリンクが下線で示されている。そこで，この中からctlstackの部分をクリックすると，大域変数ctlstackに関する情報ページ(図3(b))にとぶことができる。ここから更に型(struct Ctlframe*)の部分をクリックすれば，この型の定義場所や同じ型の他の変数情報へのリンク等が掲載されたページへ進むことができる。また，“変数の宣言場所”，“値が定義された場所”，“値が参照された場所”では該当する手続きの情報ページもしくはプログラムテキストへのリンクが示されている。これらの情報をたどることによって，この変数のプログラム内での使われ方を知ることができ，プログラム修正作業に有用な情報を得ることができる。

影響範囲解析器

影響範囲解析器は，プログラム内のある行の一つの変数に着目し，実行結果がその変数の値に直接的にも間接的にも影響を受けるコード行や，その変数のその行における値を形成したプログラム片(すなわち，そのプログラム片だけで実行しても，当該変数に関しては元のプログラムを実行したときと同一の値が得られることが保証されるようなプログラム部分集合)を見つけることができる。この機能を用いると，プログラムの修正によって影響を受ける部分，受けない部分を確実に切り分けることができる。また，修正後にも，出力変数の値を形成するプログラム片を求めることにより，修正の正しさを確認することが可能である。

3.3 システム構成

QSは，C++言語で開発され，SunOS™[*5] 4.1.3以上もしくは5.5以上で稼動する。その開発にあたっては，オブジェクト指向のアプリケーションフレームワークを構築し，それを基盤として各ツールを開発することにより，保守性はもちろんのこと，再利用性の面で大きな利点を得ることができた。例えば，C言語用のQSを例にとると，ツール平均で，ソースコードの約9割がフレームワークからの再利用であり，その効果は非常に大きい。以下の節では，フレームワークの各構成要素について解説を行う。

---

*1 Netscape，Netscape Navigator，Netscape Communicatorのロゴマークは，米国ネットスケープコミュニケーションズコーポレーションの商標。

*5 SunOS™は，Sun Microsystems Inc. の登録商標。

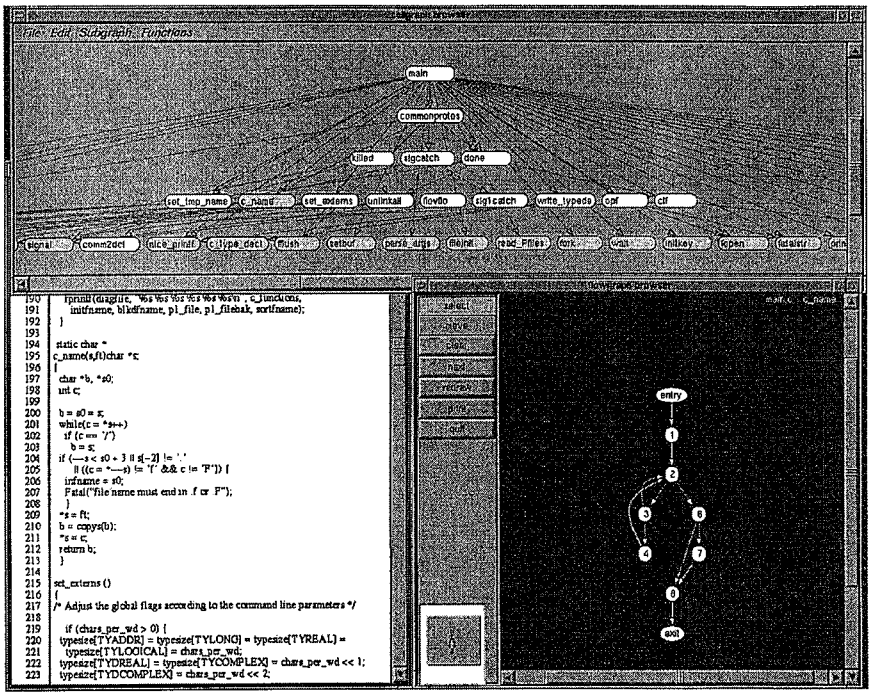
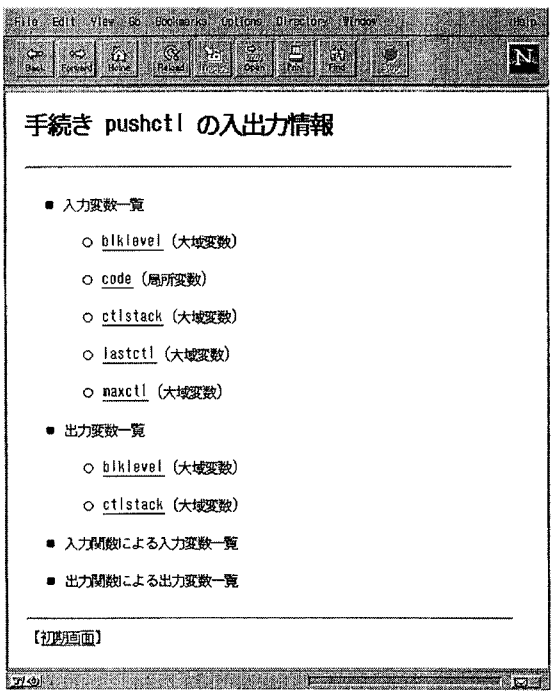
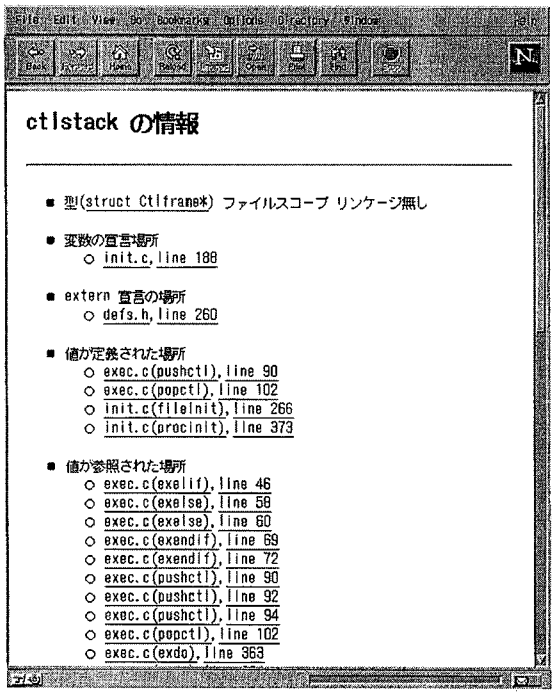


図 2　構図解析器の画面例



(a)手続き入出力ページ例



(b)変数情報ページ例

図 3　生成された保守文書例

3.3.1　GUI(Graphical User Interface)

ＧＵＩ用のコアライブラリとしてフリーソフトウェアのInterViews[13]を採用し，更にこれを用いてダイアグラム描画ライブラリを開発した。ダイアグラムとは，ある文法に則って図形と文字列で構成するグラフ構造で，複雑な情報を視覚表現するために用いられる。このライブラリでは，図形の種類，図形同士の接続関係・包含関係等の論理構造に関する規則，図形の配置規則等を自由に定義し，ダイアグラム表示を行うことができる。本ライブラリは，コールグラフやフローグラフの描画に用いられている。

3.3.2　通信

構造解析器の項で説明した複数の表示器の連動には，プロセス間通信が用いられているが，このような連動機構は，以下のような特徴がある。

・各プロセスには主従の関係は無く，すべて対等な立場で単方向のメッセージ通信を行う。

・通信の際には，送信側プロセスと受信側プロセスとでコネクションは張らない。

・メッセージ送信時に，その受信側プロセスが未起動であることがありうる。この場合，受信側プロセスが自動的に起動されて，初期

化の後に当該メッセージを受信することが期待される。

・ツール開発時には，メッセージ送受信の関係の変更がしばしば発生する。

このような機構を実現するため，LINDA[14]と呼ばれる共有メモリ型の通信モデルを採用し，これを実装したライブラリを開発した。また，メッセージタイプと送受信プロセスの関係を集中管理するクラスをこのライブラリに付加し，受信プロセスの自動起動の機能を実現するとともに，メッセージ送受信関係の変更を容易にする工夫を行った。

3.3.3 解析機能

QSのコアは，コールグラフ，フローグラフ，プログラム依存グラフ[15]等のデータ構造群と，データフロー解析[16]やプログラムスライシング[17]をベースにエイリアス解析[18]等を含めた最新のコンパイラ技術を駆使したプログラム解析モジュール群からなっており，更に解析精度の向上や処理の高速化を目的とした独自のアルゴリズムを考案して使用している。

ところで，手続き型言語*6は新日本製鐵社内で使われている主要なものだけでも数種類あるが，QSのようなツールに対する将来的なニーズを考えると，これらのいずれの言語への対応も比較的容易にしておくことが望ましい。しかしながら，プログラミング言語の文法は言語毎に異なるため，パーサ*7が生成する構文解析木*8の構造も異なるものになり，構文解析木の情報を利用する解析モジュールも言語毎に開発する必要がでてくる。

そこで，QSでは，**図4**に示すように，構文解析木へのアクセス方法を抽象化して同一のインタフェースで統一的に行わせるアクセッサと呼ぶ機構を開発し，解析モジュール群はすべてアクセッサを通してのみ，構文解析情報を利用するようにした。このようにした結果，言語毎の文法の違いは，アクセッサでほぼ吸収されるため，解析モジュールにほとんど変更を加えずに，複数の言語への対応が可能となった。例えば，C言語用のQSの中で，アクセッサを利用するいくつかのツール*9を調査したところ，それらのソースコードの約7割は，対応するFORTRANのツールと共通であり，残りの3割のほとんどは，パーサとアクセッサの実装部分であった。

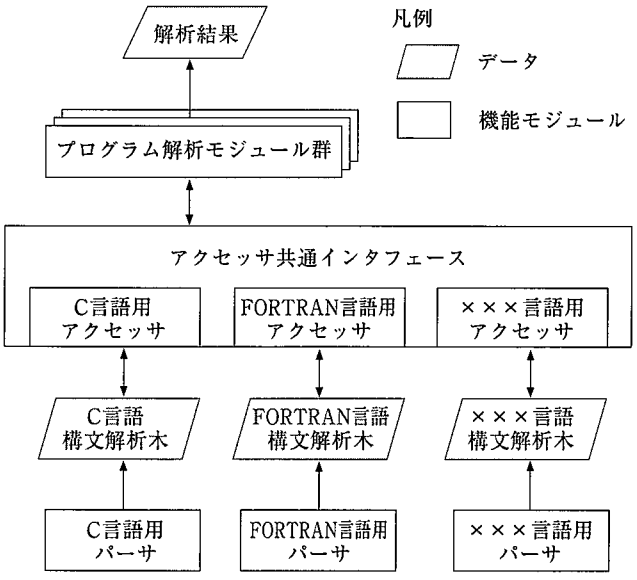


図4 多言語対応が容易なアーキテクチャ

4. 結　言

本稿では，保守業務を支援するツールセットであるQSについてその概要を紹介した。QSは，プログラムの保守品質を向上させるとともに，プログラム修正時に保守作業者に適切な情報を提供し，その作業を効果的に支援することができる。現在，QSは，新日本製鐵内の数箇所に配布され利用されている。特に製鉄事業のプロセス制御分野では，その標準開発環境であるNSCASE[19]*10の中にC言語用のものが組み込まれて利用されている。

今後は，データフロー情報の更なる厳密化のための機構の再設計及びアルゴリズムの開発を進めるとともに，使い勝手の改善や他の言語への対応も進める予定である。

参照文献

1) Tamai,T., Torimitsu,Y. : Proceedings of International Conference on Software Maintenance. 1992-11, p.63-69
2) Vessy,I., Weber,R. : Communications of the ACM. 26(2), 128 (1983)
3) Jones,C. : Assessment and Control of Software Risks. Yourdon Press, 1994
4) Yourdon,E., Constantine,L.L. : Structured Design. Yourdon Press, 1979
5) McCabe,T. : IEEE Transactions on Software Engineering. SE-2(4), 308 (1976)
6) Card,D.N. et al. : Proceedings of 8th International Conference on Software Engineering. 1985. p.372-377
7) Korson,T.D., Vaishnavi,V. K. : Empirical Studies of Programmers. 1986-6. p.168-186
8) Troy,D.A., Zweben,S.H. : The Journal of Systems and Software. 2, 113 (1981)
9) Selby,R.W., Basili,V.R. : IEEE Transactions on Software Engineering. 17(2), 141 (1991)
10) 竹下　亨：プログラムの保守・再開発と再利用．共立出版, Sep. 1992
11) Mayrhauser,A. et al. : IEEE Transactions on Software Engineering. 22(6), 424 (1996)
12) Fosdick,L.D., Osterweil,L.J. : ACM Computing Surveys. 8(3), 305 (1976)
13) Linton,M.A. : InterViews Reference Manual. Stanford University, 1992
14) Gelernter,D. : ACM Transactions on Programming Languages and Systems. 7(1), 255 (1985)
15) Ferrante,J. et al. : ACM Transactions on Programming Languages and Systems. 9(3), 319 (1987)
16) Aho,A.V. et al. : Compilers : Principles, Techniques, and Tools. Addison-Wesley, 1986
17) Weiser,M. : IEEE Transactions on Software Engineering. SE-10(4), 352 (1984)
18) Landi,W., Ryder,B.G. : Conference Record of the 18th Annual ACM Symposium on Principles of Programming Languages. 1991, p.93-103
19) 西田剛司：電気学会 金属産業電力電気応用合同研究会．1997-2

*6 手続き型言語：手続き(例えば，C言語では関数，FORTRANではサブルーチン)を基本単位としてプログラムを記述するプログラミング言語。代表的なものに，COBOL，PL/I，FORTRAN，C言語等がある。

*7 パーサ：ここでは，字句解析と構文解析の両方を実行する部分を指す。

*8 構文解析木：プログラミング言語で記述されたプログラムを、その言語の文法に照らし合わせて解釈することを構文解析と呼ぶ。そして、その際に生成されるデータ構造は一般に木構造となるので、これを構文解析木と呼ぶ。ここでは，シンボルテーブルも含む。

*9 例えば，メトリクス計測器は，言語の文法をかなり意識して実装する必要があるため，構文解析木に直接アクセスし，アクセッサはほとんど利用しない。このようなツールは比較対象から外した。

*10 NSCASEは，新日本製鐵(株)の登録商標。