

プログラム改造影響範囲分析ツールについて

7E-10

若木 守, 荻田 泰, 瀬川 正昭, 長尾 忠夫
日本電信電話株式会社

1. はじめに

ソフトウェアの開発は新規開発より保有ソフト資産を利用した改造形態が非常に多い(図1)

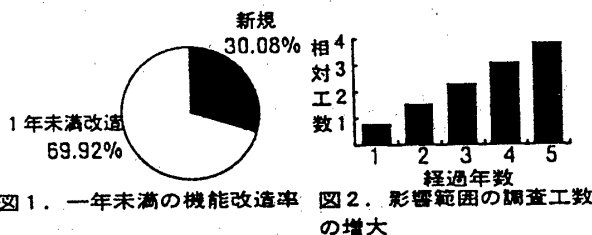
一方、保有ソフト資産の規模は数十MBに膨大化している。このような状況で、

①あるプログラムを改造する際の影響範囲を調査する工数の増大(図2)

②影響範囲を見極めずに製品を出荷し、ディグレードを引き起こして利用者に多大な迷惑をかける危険性の増大

という事態が懸念される。

本論文ではファイル内容やプログラムモジュールの変更をする時の影響範囲を自動分析するプログラム改造影響範囲分析ツールの必要性と機能、及び効果について紹介する。



2. ディグレードの種類

ディグレードの種類には以下のものがある。

(図3)

- ①参照側の論理変更を定義側に反映し忘れたことによるデータの破壊(図4)
- ②定義、参照名の違いによる変更漏れ(図5)
- ③データ構造、部品内容の定義変更による参照側変更の漏れ(図6)
- ④間接呼び出しによる影響範囲の見落とし
- ⑤その他

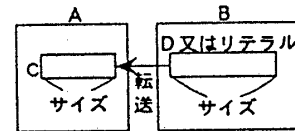
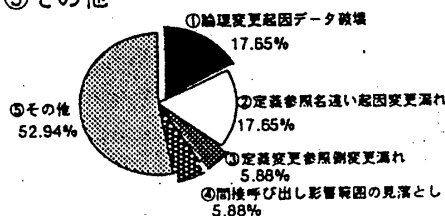


図4. 参照側の論理変更を定義側に反映し忘れた場合

図4はBモジュール内参照エリアDまたはリテラルのサイズを大きくしたのにAモジュール内定義エリアCのサイズを大きくし忘れたことによるデータ転送時の破壊の模様を示す。

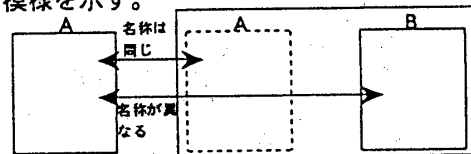


図5. 定義・参照名の違いによる変更漏れ

図5はAモジュール内定義エリアをBasedをかぶせてBモジュールで参照している場合で、Bモジュール内エリアはAモジュール内定義エリア名称とは異なるためBモジュール内参照エリアを変更しても定義側モジュールAに影響があることを見落としている。

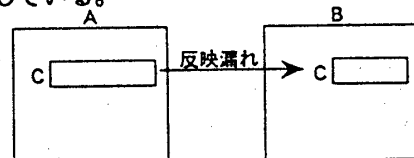


図6. データ構造・部品内容の定義変更による参照側変更漏れ

図6はAモジュール内定義エリアの内容変更を参照モジュールBに反映し忘れた場合である。

これらディグレード中、①・③・④項は改造対象モジュールとそのモジュールに関係したシステム内全モジュールとのインタフェースを見落とししたことにより発生したものである。これらは改造対象箇所と関連を持つ全ての他モジュールの箇所が明確になれば回避できる。

尚、②および⑤は他モジュールで直接定義していないエリアの更新を忘れたり作り込んだ論理の誤りであって、改造内容の論理設計を誤らないようにするしかなく、ここではディグレード防止の対象としていない。

3. ツールに要求される機能

本ツールには以下の4つの機能が必要となる。

(1) 定義・参照関係表示機能

ディグレードの種類①の回避のため、FILE名やデータ名等の定義・参照関係を明確にする機能。

(2) モジュール単位部品使用状況表示機能

ディグレードの種類③の回避のため、指定モジュールが使用しているマクロ、COPY句の一覧をクロスリファレンス形式で明確にする機能。

(3) 間接呼び出し関係表示機能

ディグレードの種類④の回避のため、CALL・マクロのネスト関係をトップダウンあるいはボトムアップ形式で明確にする機能。

(4) 文字列検索機能

ディグレードの種類①の回避のため、指定リテラル定数が定義されている箇所を明確にする機能。

表1 ツールの機能一覧

機能	表 概要	処理例				
定義・参照関係表示機能	FILE、データ、サブルーチン、CALL、COPY句、マクロに関して定義箇所と参照箇所を表示	ファイル名	モジュール名			
		AAA	PRO1	PRO2	PRO3	
		BBB	PRO1	PRO4		
CCC	PRO2	PRO3	PRO4			
間接呼び出し関係表示機能	CALL、マクロのネスト関係を表示する	第1次元呼び名	第2次元呼び名			
		SUB1	SUB11	SUB111		
				SUB112		
		SUB12				
モジュール単位部品使用状況表示機能	指定モジュールが使用しているマクロ、COPY句の一覧を作成する	モジュール名	マクロ名			
		PRO1	A\$AAA	B\$BBB	C\$CCC	
		PRO2	A\$AAA	C\$CCC	D\$DDD	
		PRO3	B\$BBB	F\$FFF		
文字列検索機能	指定文字列を持つリテラル情報を定義しているモジュールを表示する	文字列	行番号・モジュール名・行内容			
		+001	001000	PRO1	MOVE +001 TO AAA	

5. ツールの機器構成

本ツールはCOBOLソースをコンパイルした出力リスト情報を入力とし、必要なキー情報を分析情報ファイルへ抽出する抽出機能と、分析情報ファイルに格納したデータを指定形式に編集・出力する編集機能から構成される。

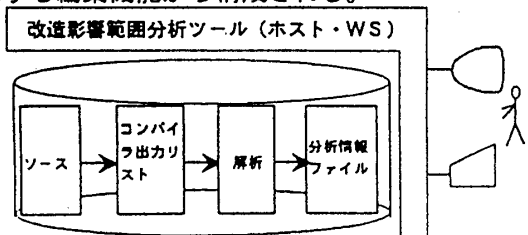


図7. 機器構成図

6. ツール作成上のポイント

本ツールは数MBにもものぼるAPのキー情報を取り扱うため、性能上のネックが予想されること及び将来における多様な機能追加を想定して以下に示す配慮がなされている。

(1) 抽出機能

①インラインマクロ、マクロネスティングを自前で展開・分析することによる性能劣化を防止するためCOBOLコンパイラの出力リストをMT/DK経由で入力とする。

②システム全体の分析情報ファイル中、更新したモジュールのキー情報だけ差し替えを可能とし、分析情報ファイルの更新が短時間で済ませる。

③アクセス速度及び更新を容易にするため、分析情報ファイルは直接編集ファイルとし、ファイルの先頭に管理情報部を設け、そこから各モジュールのキー情報部へポインタを張ってランダムアクセスを可能としている。

④COBOLソースはMIA仕様COBOLやベンダ個別仕様COBOLなどキー情報及びキー情報の位置が仕様毎に異なるため、キーサーチ処理を個別部分と共通部分に分け、サポートするCOBOLの種類の追加を容易に可能とする。

(2) 編集機能

①使用メモリ量を少なくして実装メモリ台数の少ないWSでも動けるよう、分析情報ファイルのデータは必要モジュールの部分のみメモリ上へ展開する。

②HMIを善くするため可能な限り運用者とインタフェースを持つ部分はグラフィックを用いる。

③将来の機能追加に備えて共通処理のサブルーチン化を徹底する。

7. 特徴

MIA仕様COBOLで本ツールを作成しているためマルチベンダ上で動作可能

8. 効果

標準的な実測結果を表2. に示す。

表2. 工程別稼働削減例

基本検討〜プログラム設計工程での改造箇所抽出に要する工数削減	90%減
試験以降のトラブル対応に要する工数削減	47%減

9. 終わりに

本ツールは実用化が終わった段階で多くのシステムで使用してもらい、特にHMIの改善を行っていきたい。