ツールでどこまでリバースエンジニアリング作業を支援できるか

門井　仁†

概要：仕様書もテストコードもない，いわゆるレガシーアプリケーションをリバースエンジニアリングする場合，その全てを手作業で実施するのは，非常に効率が悪い．既存のツールを利用，また，必要に応じてツールを開発する事で，どこまで作業を効率化できるか検討する．

How efficient reverse engineering can be by using appropriate tools?

HITOSHI KADOI†

Abstract: It is very inefficient to reverse so-called legacy application code manually. In this paper, I will examine how much work can be done efficiently by using existing tools or tools developed as needed.

1. はじめに

いわゆるレガシーと呼ばれているアプリケーションの更改は，現在においても難事業である．また，現在においても，世に出ている様々は開発メソドロジーを何も適用せずに，進めている開発プロジェクトもある．そういったシステムに共通するのは，図 1 にあるように，

- 仕様書がない，あってもコードと乖離している
- テストコードがない，または不足している

等である．

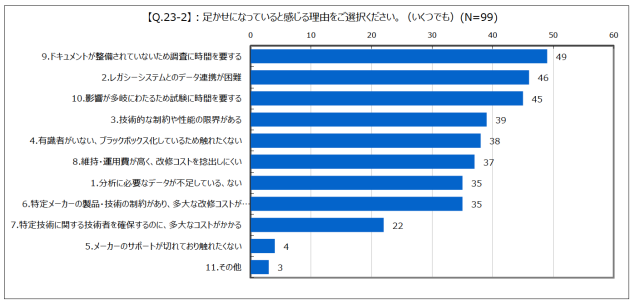


図 1　レガシーシステムが足かせと感じる理由[1]

このような状態になってしまったシステムを，リバースエンジニアリングするに当たり，どこまでツールにより効率化できるか，実際に筆者が携わったプロジェクトでの経験を元に考えてみる．

2. リバースエンジニアリングの工程

リバースエンジニアリングという言葉は，色々な意味合いで使われることがあるが，ここで論じるのは，コードを調査して現行の動作を明らかにする事であり，バイナリコードを逆アセンブルするといったものは含んでいない．筆者が携わったプロジェクトでは，リバースエンジニアリング結果から，要求仕様との整合性確認と，不足しているテストケースを作成することが目的であったため，リバースモデリングにおけるリバース設計[2]に近いものである．

各工程について，内容と難しさを以下に示す．

2.1 概要の把握

まずはコード全体の規模を掴むために，ステップ数（コメント有り，なし）を把握する．ここでは関数単位のステップ数を把握する必要がある．さらに，作業にかかる工数を見積もる目安にするため，循環的複雑度等の内容の複雑度を示すメトリックスを把握する．

また，関数同士のコール関係を明らかにする事で全体の構造を掴む．

さらに，動作にタイミング要素があるかどうか，プログラムをどの様に稼動させているかを調査して把握する．

2.2 入力パターンの把握

各関数に対する入出力変数を明らかにする．関数の引数だけでなく，グローバル変数も対象となる．

入力変数について，同値分割により各々の値域を調査する．ここでは条件分岐の調査が必要となるが，そこに現れる変数が一次変数であった場合，その中身がグローバル変数の内容なのか，計算ロジックの結果が入ったものなのか，といった出自を辿ることが必要となり，調査工数が大きくなる．また，お決まりの Null チェック等のセーフガード的なコードが含まれている場合，それが仕様上重要なものであるのかどうかの判断が難しい．

2.3 入力パターンに応じた出力（振る舞い）の把握

この工程がもっとも時間の掛かるものである．条件分岐単位に，どの様な振る舞いがあり，それによりどの様な出力がされるか，実際にコードを読んでロジックを確認して

† 門井　仁,Advisory Architect
Engineering and Cognitive Innovation
Global Business Services, IBM Japan, Ltd
email: hkadoi@jp.ibm.com

いかなければならない．分岐のネストが深い場合，表にして入力パターン毎の振る舞いを一覧化しないと，全体の振る舞いを把握できないことが多い．

2.4 要求仕様とのマッチング

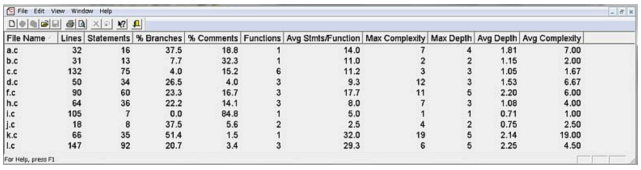
これまでの工程の成果物は，事実を表したものに過ぎない．これが要求仕様と合致した正しいものであるかどうかを確認する必要がある．ここでは振る舞いと変数（パラメータ）について確認する事になるが，仕様書が残っているものは，それとの比較になる．残っていないものについては，有識者に聞くしかないと考えられるが，それでも難しい場合は，コードに書かれていることを正とするしかない．

3. 支援ツール

ここでは，2 章で述べた各工程において，どの程度ツールが作業支援できるのか論じる．2.4 については本質的にツール化が難しいと考えられるため，除外する．

3.1 概要の把握

ここについては，既存の支援ツールが存在する．無償の範囲内でも，SourceMonitor[3]は，コードのステップ数に加えメトリックスも取得することができる（図 2）．また，Doxygen[4]は，関数同士のコール関係を表示することができる．



File Name	Lines	Statements	% Branches	% Comments	Functions	Avg Stmts/Function	Max Complexity	Max Depth	Avg Depth	Avg Complexity
a.c	32	16	37.5	18.8	1	14.0	7	4	1.81	7.00
b.c	31	13	7.7	32.3	1	11.0	2	2	1.15	2.00
c.c	132	75	4.0	15.2	6	11.2	3	3	1.05	1.67
d.c	50	34	26.5	4.0	3	9.3	12	3	1.53	6.67
f.c	90	60	23.3	16.7	3	17.7	11	5	2.20	6.00
h.c	64	36	22.2	14.1	3	8.0	7	3	1.08	4.00
i.c	105	7	0.0	84.8	1	5.0	1	1	0.71	1.00
j.c	18	8	37.5	5.6	2	2.5	4	2	0.75	2.50
k.c	66	35	51.4	1.5	1	32.0	19	5	2.14	19.00
l.c	147	92	20.7	3.4	3	29.3	6	5	2.25	4.50

図 2　SourceMonitor 結果サンプル

3.2 入力パターンの把握

前述のグローバル変数について，変数を直接参照するものについては辿りやすいが，例えば，グローバル変数のアドレスをローカル変数に渡して，さらにそれを関数の引数として渡すといったコードが書かれている場合がある．呼び出し階層が深い場合，これを辿るのが手作業では非常に厄介である．これを扱うツールは，調査した範囲では存在しないため，筆者が携ったプロジェクトでは，新たに開発した．具体的には，gcc -E によるプリプロセス結果を元に変数を辿っていき，関数と変数の Read/Write マッピング表を作るというものである．（図 3）関数はコール関係で示している．

			Var1	Var2	Var3	…
Func1						
	Func1-1			R		
	Func1-2				RW	
		Func1-2-1				
:						

図 3　Read/Write マッピング表

同値分割を行うツールも存在しない．こちらも開発が必要であるが，値域を示す境界値が定数ではなく，何らかの計算結果であったり，関数の戻り値であったりすることまで考慮してツールを作るのは難しい．このため，条件文を抜き出し，作業者が見やすい様に編集したものを提供するに留めた．これだけでもソースを読みながら実施することと比較して十分に効率化を図ることが出来ている．

3.3 入力パターンに応じた出力（振る舞い）の把握

この工程が最も時間が掛かるものであるが，ツール化ということでは，コードを完璧に解釈して，それを作業者が見やすい様に整形することが求められるため，大変難しい．入力パターンをインプットにして動的に解析することも考えられるが，なぜその出力結果となるか，コードを見て判断する必要があり，あまり効率化には繋がらないと考えられる．

このため，プリプロセッサによるコードの整形のみ実施し，作業者がそのコードを見ながら解釈していく，という程度しかできないのではないかと考える．

4. 議論

本議論においては，振る舞いの把握についてのツール化について，あまり深く掘り下げることが出来ていない．同様に，動作にタイミング要素がある場合のツールについて検討できていない．

また，アプリケーション全体の振る舞いという観点では，例えば，今回取り上げた，Read/Write マッピングを発展させて，アプリケーション全体のデータフローを明らかにする事で，ある程度把握できる可能性がある．

この様な点について，効率化の可能性を深く議論できると幸いである．

参考文献

[1] 一般社団法人日本情報システム・ユーザー協会「デジタル化の進展に対する意識調査」（平成 29 年）
[2] SESSAME WG2.組み込みソフトウエア開発のためのリバースモデリング.翔泳社,2007,p.52
[3] SourceMonitor
http://www.campwoodsw.com/sourcemonitor.html
[4] Doxygen
https://ja.osdn.net/projects/sfnet_doxygen/