レガシーコードの蘇生術
～リバースモデリングツール REXSTM for Cのご紹介～

2017年7月12日

技術本部応用技術調査委員会/
状態遷移設計研究WG
山本　椋太

一般社団法人 組込みシステム技術協会 Japan Embedded Systems Technology Association

状態遷移設計研究ワーキンググループとは

http://www.jasa.or.jp/TOP/activity/technology/state/

状態遷移設計研究ワーキンググループ

状態遷移設計研究WG

［状態遷移設計研究WG主査］
青木 奈央
キャッツ(株)

平成29年度事業
既存ソースコードから、状態変数を抽出し状態遷移表をリバース生成する手法の研究を継続する。
・ リバースモデリング手順のガイドの作成とツール化の検討。
・ セミナー、講演会などの広報活動　他
なお、今年度は事例拡充のための、プロトタイプツールの作成を、産学連携（情報技術人材育成のための実践教育ネットワーク形成事業：ｅｎPiT）により推進する。

- 状態遷移設計の普及啓蒙活動
- 「状態遷移表のリバースモデリングへの適用」

・既存ソースコードから、状態遷移表を逆生成する手法の研究、ツール化。
・ツールの多言語対応化
・セミナー、講演会などの広報活動を含む論文作成　他

①状態遷移設計研究会定例会議

状態遷移設計研究WGの方針

1. 状態遷移設計の普及啓蒙活動
2. 「状態遷移表のリバースモデリングへの適用」
   - 既存ソースコードから、状態遷移表を逆生成する手法の研究、ツール化。
   - ツールの多言語対応化
   - セミナー、講演会などの広報活動を含む論文作成
   他

RExSTMの開発

現在の組込みソフト開発の状況

■ 組込みソフトウェア開発の傾向

- 派生開発による短納期・高品質の要望
- レガシーコードの肥大化・複雑化→メンテナンス性低下

→ 設計資料なし、担当者もすでにいない、
修正したら予想外の問題が出る…

ソースコードのブラックボックス化が進行中！

■ 効率化のための手法導入が進まない

- 派生開発・レガシーコードのせいで従来のものを踏襲せざるを得ない…

レガシーコードの存在が足かせになっている！

仮説と研究テーマ

■ リバースエンジニアリング
- レガシーコードを解析することで仕様を明らかにする

■ 仮説
- 状態遷移設計は普遍的なモデル
- レガシーコードにも状態遷移は必ずあるはず！

フラグのあるところに、状態がある！

■ 研究テーマ
- 「状態遷移表のリバースモデリングへの適用」
- レガシーコードから状態遷移表をリバースモデリングする

状態遷移表でレガシーコードを蘇生！

状態遷移表

関数	条件	スロットの状態			
		全て回転中	1つ停止	2つ停止	3つ停止
Entry4	L、M、Rがすべて一致	状態遷移：1つ停止	状態遷移：2つ停止	状態遷移：3つ停止 総額にBET金額の5倍を加算	状態遷移：全て回転中
	L、M、Rの2つが一致	状態遷移：1つ停止	状態遷移：2つ停止	状態遷移：3つ停止 総額にBET金額加算	状態遷移：全て回転中
	else	状態遷移：1つ停止	状態遷移：2つ停止	状態遷移：3つ停止	状態遷移：全て回転中

状態変数

状態

イベント

遷移・処理

状態遷移表

関数	条件	flag_b1_c			
		0	1	2	3
Entry4	L、M、Rがすべて一致	flag_b1_c = 1	flag_b1_c = 2	flag_b1_c = 3 Total += Bet*5	flag_b1_c = 0
	L、M、Rの2つが一致	flag_b1_c = 1	flag_b1_c = 2	flag_b1_c = 3 Total += Bet	flag_b1_c = 0
	else	flag_b1_c = 1	flag_b1_c = 2	flag_b1_c = 3	flag_b1_c = 0

状態変数

状態

イベント

遷移・処理

サンプル検証1（意味不明→仕様明確化）

関数	条件	flag_b1_c			
		0	1	2	3
Entry4	L、M、Rがすべて一致	flag_b1_c = 1	flag_b1_c = 2	flag_b1_c = 3 Total += Bet*5	flag_b1_c = 0
	L、M、Rの2つが一致	flag_b1_c = 1	flag_b1_c = 2	flag_b1_c = 3 Total += Bet	flag_b1_c = 0
	else	flag_b1_c = 1	flag_b1_c = 2	flag_b1_c = 3	flag_b1_c = 0

状態変数名が意味不明

状態を0～3の直値で指定

条件は 状態2 のときのみ有効

リバースモデリングの作業手順

①レガシーコードの整形

- コメント削除、＃ifdefを設定
- 解析範囲の決定

③条件処理表（仮の状態遷移表）の作成

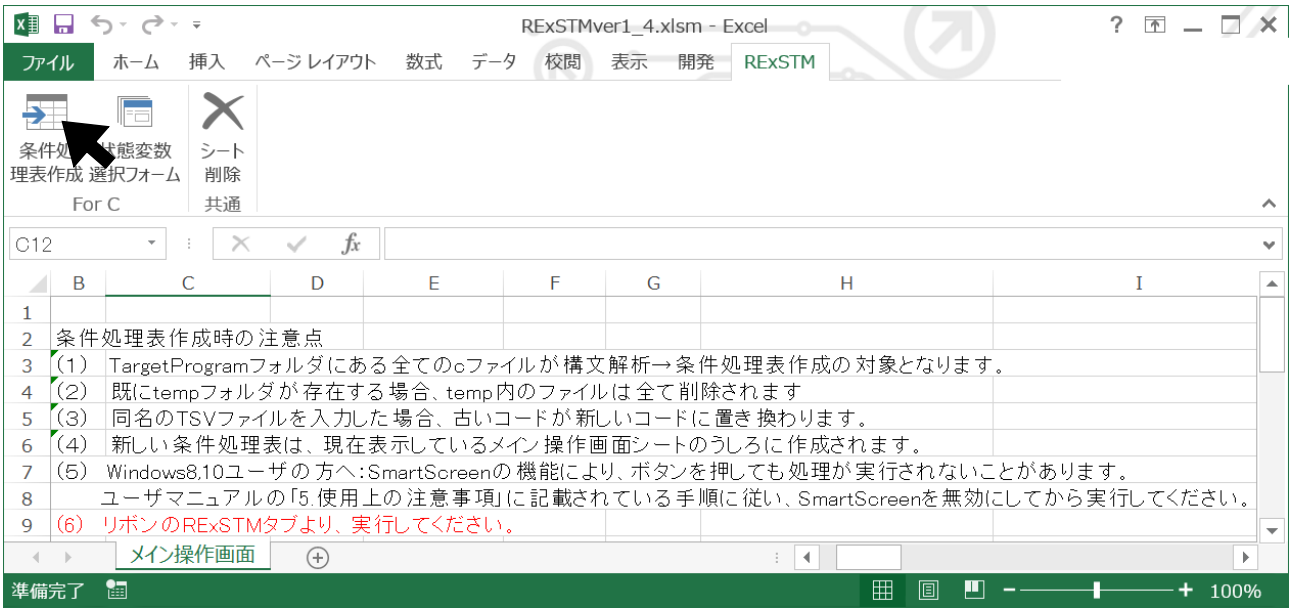
- 分岐条件を階層化
- 条件・処理の対応を記載した条件処理表の作成

④状態変数の抽出

- 状態変数の抽出
- 状態遷移表の編集

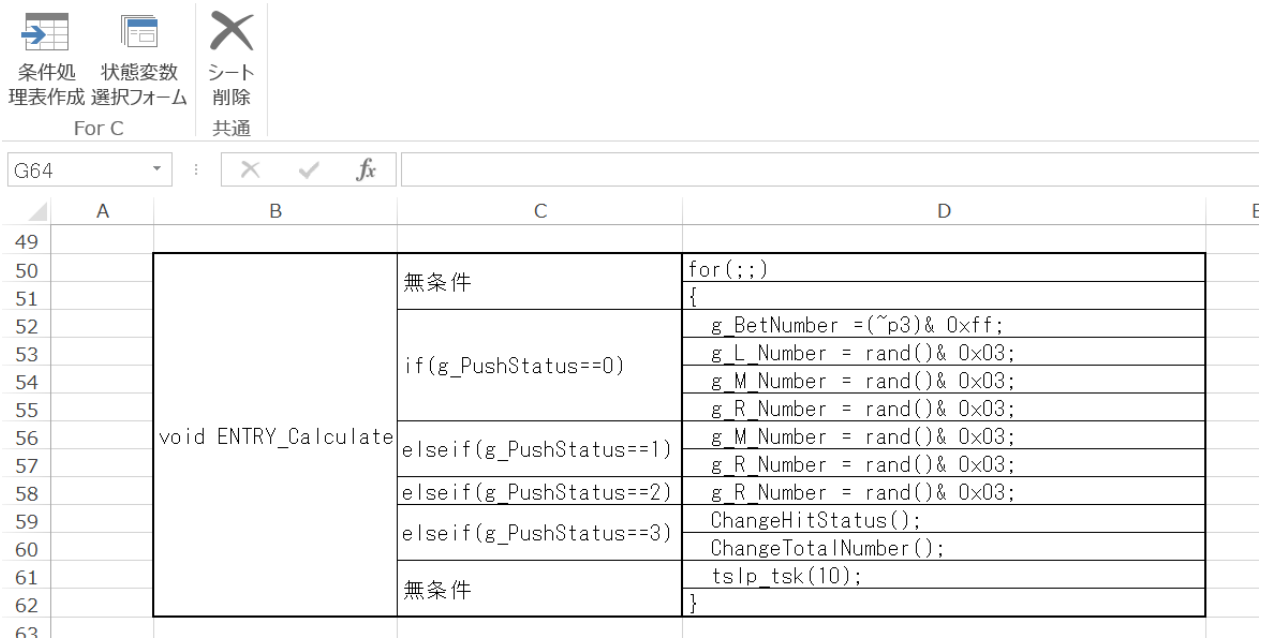
状態遷移表の作成 完了！

RExSTM for C　条件処理表・状態遷移表作成ツール



ソースコードの整形・解析を行い、
状態遷移表生成のための情報整理を実施

RExSTM for C　条件処理表の出力



	A	B	C	D
49				
50		void ENTRY_Calculate	無条件	for(;;)
51				{
52			if(g_PushStatus==0)	g_BetNumber =(~p3)& 0x03;
53				g_L_Number = rand()& 0x03;
54				g_M_Number = rand()& 0x03;
55				g_R_Number = rand()& 0x03;
56			elseif(g_PushStatus==1)	g_M_Number = rand()& 0x03;
57				g_R_Number = rand()& 0x03;
58			elseif(g_PushStatus==2)	g_R_Number = rand()& 0x03;
59			elseif(g_PushStatus==3)	ChangeHitStatus();
60				ChangeTotalNumber();
61			無条件	tslp_tsk(10);
62				}
63				

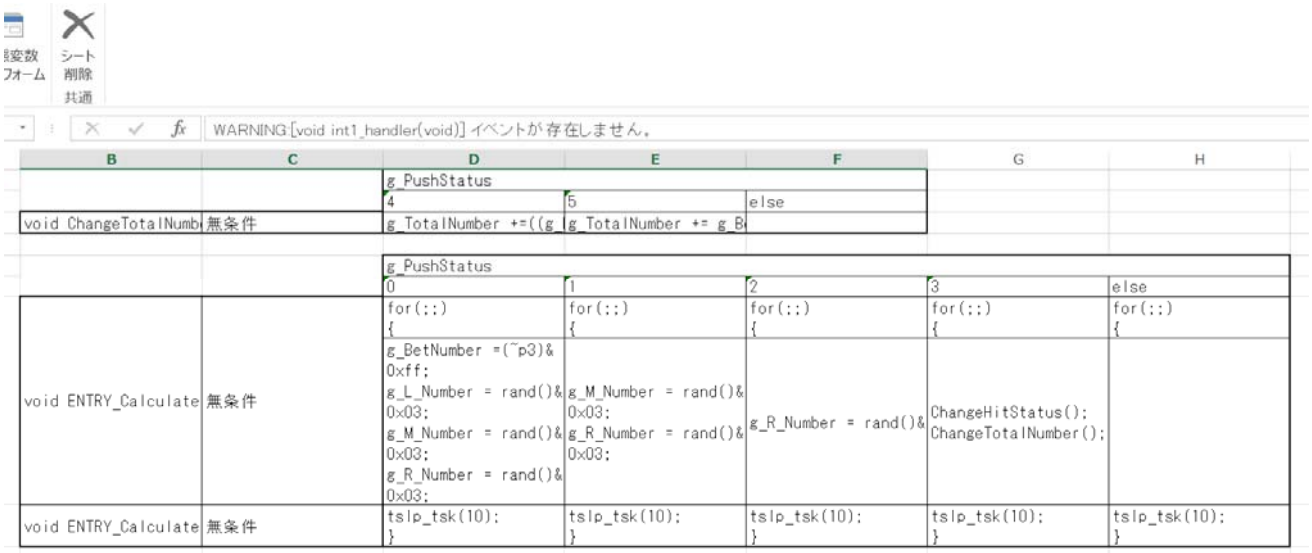
RExSTM for C　状態変数の選択

状態変数の
候補を抽出して表示

```
void ChangeHitStatus()
{
        /*3つの数字が一致する場合 — HIT3 */
        if( ( g_L_Number == g_M_Number) && ( g_M_Number == g_R_Number ) ){
                g_PushStatus = STATUS_Hit3;
        }
        /*どれか 2 つの数字だけが一致する場合 — HIT2 */
        else if( ( g_L_Number == g_M_Number) ||
                ( g_L_Number == g_R_Number ) || ( g_M_Number == g_R_Number ) ){
                g_PushStatus = STATUS_Hit2;
        }
        /*どれも一致しない場合 — MISS */
        else{
                g_PushStatus =
        }
}
```

ソース上、どこで変数が
使われているかをハイライト表示

RExSTM for C 状態遷移表の出力



WARNING:[void int1_handler(void)] イベントが存在しません。

		g_PushStatus				
		4	5	else		
void ChangeTotalNumb	無条件	g_TotalNumber +=((g	g_TotalNumber += g_B			

		g_PushStatus				
		0	1	2	3	else

void ENTRY_Calculate	無条件	for(;;)
		{
		g_BetNumber =(~p3)&
		0xff;
		g_L_Number = rand()&
		0x03;
		g_M_Number = rand()&
		0x03;
		g_R_Number = rand()&
		0x03;
			for(;;)
			{
			g_M_Number = rand()&
			0x03;
			g_R_Number = rand()&
			0x03;
				for(;;)
				{
				g_R_Number = rand()&
					for(;;)
					{
					ChangeHitStatus();
					ChangeTotalNumber();
						for(;;)
						{

void ENTRY_Calculate	無条件	tslp_tsk(10); }	tslp_tsk(10); }	tslp_tsk(10); }	tslp_tsk(10); }	tslp_tsk(10); }

RExSTM for C ツールデモ

■ 画面を御覧ください。

2016年度　活動報告

① ツール化活動

- enPiT/Emb（分野・地域を超えた実践的情報教育協働ネットワーク　組込みシステム分野）を活用した、OJLによるツールの改良を実施

② 普及活動

- 広報活動の実施
- 学会での発表

③ ガイドライン

- キャプチャー動画 or スライドショー形式による操作手順（リバース手順）のガイドの整備

①ツール化活動

■ リバースモデリングツールの検討・作成

- 2014年度
  - ー ツール要件固めを進めた.
  - ー 自動でできる作業はツールで実行
    - ✓ **ソースから条件・処理対応表（条件処理表）を生成**
    - ✓ **状態変数候補を抽出、選択→状態遷移表に展開**
  - ー 状態変数の特定など人力が必要な作業に集中
    - ✓ 現場で導入がしやすくなる、検証してもらえる
- 2015年度
  - ー enPiT を利用して実際にツール化
  - ー 現状、あるソースコードに対しては適用可能であることを確認した

②普及活動　③ガイドライン

- 2016年度
  - ツールのブラッシュアップをした。
  - 解析可能な対象の増やした。
  - 公開に向けての準備作業をした。
    - ✓ マニュアルやドキュメント類を作成
  - 電子情報通信学会 ソフトウェアサイエンス研究会
    - ✓ 2016年7月
    - ✓ 山本，吉田，竹田，舘，高田，“組込みソフトウェアを対象とした状態遷移表抽出手法”
  - ZIPC WATCHERS Vol. 19
    - ✓ 吉田，青木，“コードからSTMへリバースRExSTM for C ツールについて”

②普及活動　③ガイドライン

- 2017年度
  - ツールを公開し普及活動をする。
  - ツールのプロモーション動画を作成しツールを利用したい企業へ配布する。
  - 論文や記事などを執筆する。
  - ツール公開後も不具合や使い勝手などを改修、サポートを実施。

WGの議論中や開発中に気が付いたこと

- C言語の表現の多様さに改めて気が付いた。
  - 他の言語と比較しても、C言語は改めて自由度が高いことに気が付いた
- 要求仕様に時間がかかった
  - あいまいな部分を形にするのが難しかった
- 状態遷移表に落とし込むときに非常に苦労した
- 状態遷移設計に対するバリエーションの多さが難しかった
- 理論的に理解しても、それをロジックに落とし込むときに想定していないコードがあった
- 状態遷移表の構造との差分がコードに見受けられた
- 状態モデルで作成されたソースコードでも後日パッチ等を当てた場合に、当初の状態モデルのコードがくずれてしまい、そのようなソースコードのリバースには苦労した。
- 状態変数を見つけることじたいが非常に困難

状態遷移設計をされているコードをみつけることだけでも意味がある

RExSTM for C　今回の成果

■ ソースコード→状態遷移表生成の多くを自動化

- 単純作業が減り、人が考える部分に集中できる
  - ー ソースコード整形、状態遷移表作成などは自動で実施
  - ー 手動と比べて作成時間を大きく減らすことができた
  - ー ただし、単純なパターンのみ

■ 状態変数の候補を自動抽出

- 状態変数の条件に合致する変数を自動抽出
- 候補として表示されたものから選ぶだけでよい
  - ー 手作業の場合、状態変数の特定が一番難しい

公開についての予定

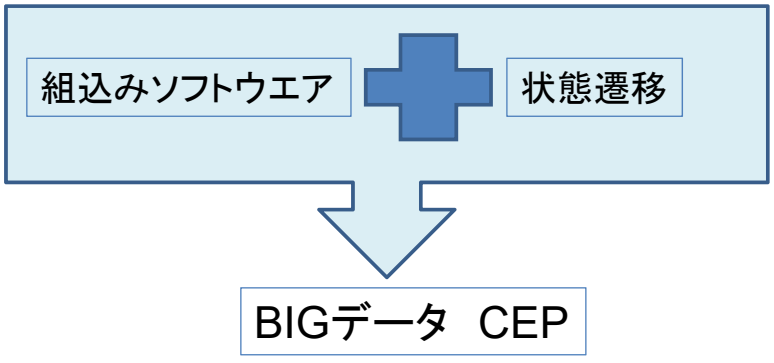
■ 公開の日程
- 当初5月末予定を目標にしていたが、品質向上のため8月予定と延期した

■ パブリックコメント
- 8月末から3ヶ月程度JASAのメンバー専用ページで公開する予定
- ツールの使い方や不具合等の質問を受け付けます
- ダウンロードには、社名や連絡先を記入していただく予定です。

■ ツールの修正
- パブリックコメントをベースにツールを改修する

WGとしての今後

■ 今後は、BIGデータ関連で、まだ日本に浸透していないCEP（Complex Event Processing）について勉強会を実施する



ご清聴ありがとうございました

サンプル検証コード募集中！
状態遷移設計研究WG　会員募集中！

【講演タイトル】

2017/7/12 発行

発行者　一般社団法人 組込みシステム技術協会
東京都中央区日本橋大伝馬町6-7
TEL: 03(5643)0211 FAX: 03(5643)0212
URL: http://www。jasa。or。jp/