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～

2016年11月16日

名古屋大学大学院 情報科学研究科　　若林　丈紘

一般社団法人 組込みシステム技術協会 Japan Embedded Systems Technology Association
© Japan Embedded Systems Technology Association 2016

状態遷移設計研究会とは

http://www.jasa.or.jp/top/activity/state_transition.html

© Japan Embedded Systems Technology Association 2016

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！

© Japan Embedded Systems Technology Association 2016

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！

© Japan Embedded Systems Technology Association 2016

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

②構文ツリーの作成

・フローチャートの生成
・構文ツリー図の作成

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

条件			処理
無条件実行			T=in1,S=in2
T==1			S++
	S<X1	State==S1	(Out=S)
		State==S2	(Out=0),State=S3
		else	S++,State=S1
	else	State==S1	(Out=X2+X2)
		else	(Out=X1),State=S2
T>4	State==S1		(Out=S)
else			(Out=0)

条件			処理
無条件実行			T=in1,S=in2
T==1			S++
	S<X1	State==S1	(Out=S)
		State==S2	(Out=0),State=S3
		State==S3	S++,State=S1
	else	State==S1	(Out=X2+X1)
		State==S2	(Out=X1),State=S2
		State==S3	(Out=X1),State=S2
T>4	State==S1		(Out=S)
	State==S2		/
	State==S3		/
else			(Out=0)

④状態変数の抽出

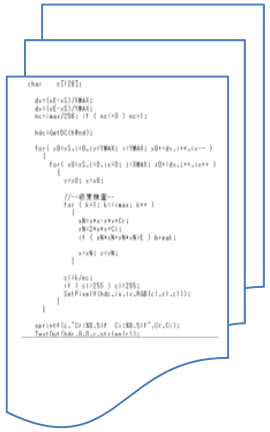
・状態変数の抽出
・状態遷移表の編集

		State		
		S1	S2	S3
無条件実行		T=in1,S=in2	T=in1,S=in2	T=in1,S=in2
T==1		S++	S++	S++
	S<X1	(Out=S)	(Out=0),State=S3	S++,State=S1
	else	(Out=X2+X1)	(Out=X1),State=S2	(Out=X1),State=S2
T>4		(Out=S)	/	/
else		(Out=0)	(Out=0)	(Out=0)

状態遷移表の作成 完了！

リバースモデリングの作業手順

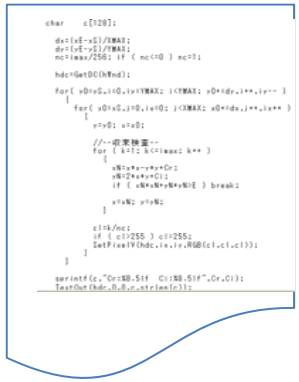
ソースコード前処理



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

コメントはいらない！



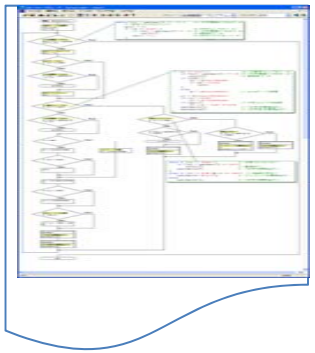
②構文ツリーの作成

・フローチャートの生成
・構文ツリー図の作成

© Japan Embedded Systems Technology Association 2016

リバースモデリングの作業手順

条件処理表の作成



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

条件			処理
無条件実行			T=in1,S=in2
T==1			S++
	S<X1	State==S1	(Out=S)
		State==S2	(Out=0),State=S3
		else	S++,State=S1
	else	State==S1	(Out=X2+X2)
		else	(Out=X1),State=S2
T>4	Sate==S1		(Out=S)
else			(Out=0)

・条件分岐をif－else構造でモデル化
・switch-caseもif－else構造
・条件処理表の作成

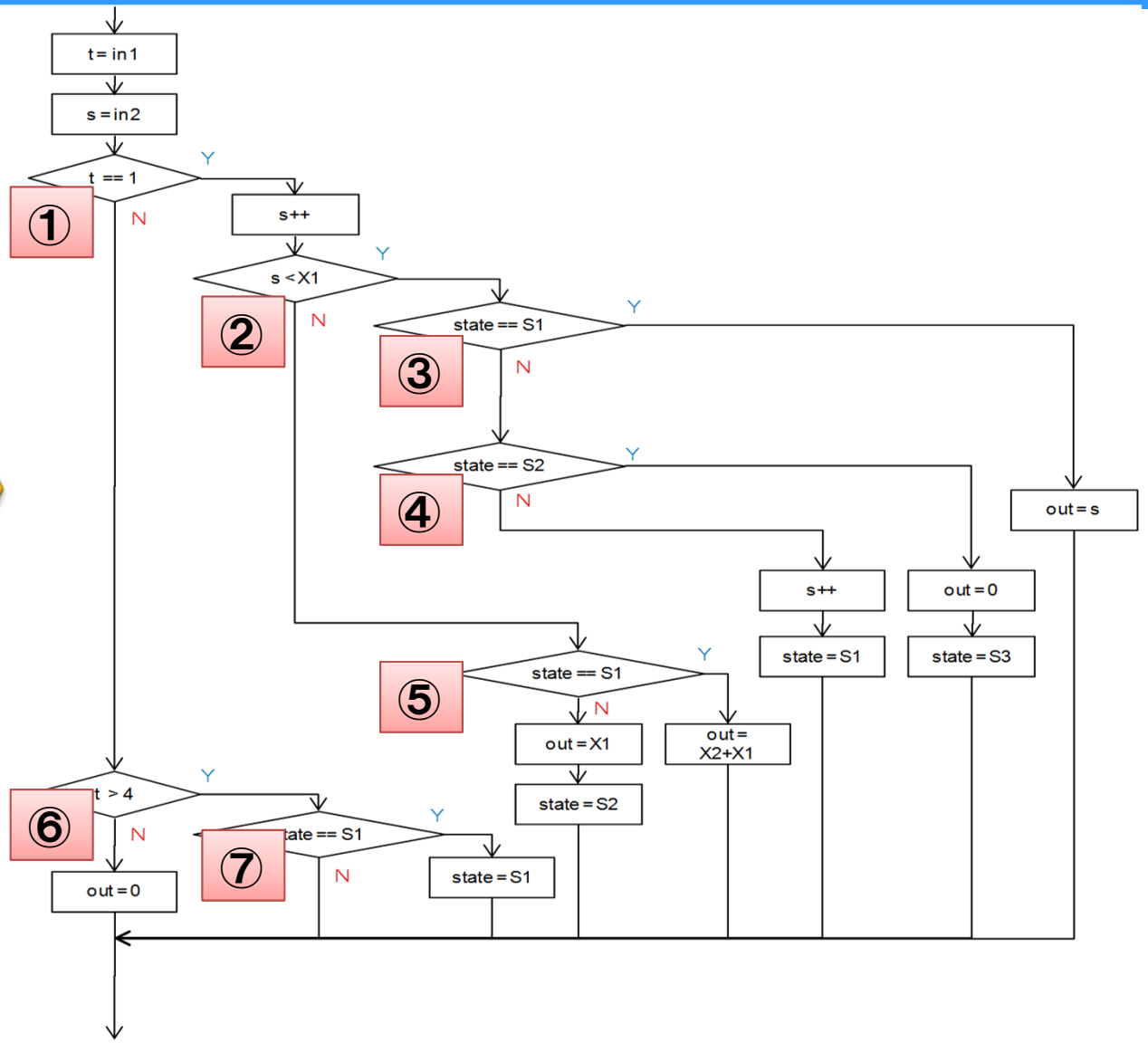
© Japan Embedded Systems Technology Association 2016

リバースモデリングの作業手順

```
extern int in1,in2
int state;
#define S1  1
#define S2  2
#define S2  3
#define X1 5
#define X2 T1*2
int out

void task(void)
{
   int t,x;

   t = in1;
   s = in2;
   if ( t == 1) {
      s++;
      if (s < X1) {
         if( state == S1 ){
            out = s;
         else if( state == S2 ){
            out = 0;
            state = S3
         else{
            s++;
            state = S1;
         }
      } else {
         if( state == S1 ){
            out = X2+X1;
         else{
            out = X1;
            state = S2
      }
   }
   else if( t > 4 ){
      if( state == S1 ){
            out = s;
      }
   }
   else{
      out = 0;
   }
}
```



© Japan Embedded Systems Technology Association 2016

リバースモデリングの作業手順

条件処理表の作成

条件			処理
無条件実行			T=in1,S=in2
T==1 ①			S++
	S<X1 ②	flg==S1 ③	(Out=S)
		flg==S2 ④	(Out=0), flg=S3
		else	S++flg,=S1
	else	flg==S1 ⑤	(Out=X2+X1)
		else	(Out=X1), flg=S2
T>4 ⑥	⑦	flg==S1	(Out=S)
else			(Out=0)

ソースの分岐条件から条件を左、処理を右、に置いた仮の状態遷移表を作成

フローチャートの①～⑦の構成と、条件処理表の①～⑦の構成は同じになる

© Japan Embedded Systems Technology Association 2016

リバースモデリングの作業手順

状態変数の抽出　　状態遷移表の作成

条件			処理
無条件実行			T=in1,S=in2
T==1			S++
	S<X1	State==S1	(Out=S)
		State==S2	(Out=0),State=S3
		State==S3	S++,State=S1
	else	State==S1	(Out=X2+X1)
		State==S2	(Out=X1),State=S2
		State==S3	(Out=X1),State=S2
T>4	State==S1		(Out=S)
	State==S2		/
	State==S3		/
else			(Out=0)

④状態変数の抽出

- 状態変数の抽出
- 状態遷移表の編集

		State		
		S1	S2	S3
無条件実行		T=in1,S=in2	T=in1,S=in2	T=in1,S=in2
T==1		S++	S++	S++
	S<X1	(Out=S)	(Out=0),State=S3	S++,State=S1
	else	(Out=X2+X1)	(Out=X1),State=S2	(Out=X1),State=S2
T>4		(Out=S)	/	/
else		(Out=0)	(Out=0)	(Out=0)

■ ポイントは、状態変数を見つけること

■ 分岐条件の変数は状態変数か？

- 状態変数の定義
  - 状態変数は有限個である
  - 状態変数は、内部で更新される

© Japan Embedded Systems Technology Association 2016

リバースモデリングの作業手順

条件			処理
無条件実行			T=in1,S=in2
T==1			S++
	S<X1	flg==S1	(Out=S)
		flg==S2	(Out=0),flg=S3
		flg==S3	flg＝S1
	else	flg==S1	(Out=X2+X1)
		flg==S2	(Out=X1),flg=S2
		flg==S3	(Out=X1),flg=S2
T>4	flg==S1		(Out=S)
	flg==S2		/
	flg==S3		/
else			(Out=0)

flgはS1,S2,S3の有限値かつflgは内部で更新されている

flgは、状態変数だ

elseを有限値に置換

© Japan Embedded Systems Technology Association 2016

リバースモデリングの作業手順

状態遷移表の作成

条件			flg		
			S1	S2	S3
無条件実行			T=in1,S=in2	T=in1,S=in2	T=in1,S=in2
T==1			S++	S++	S++
	S<X1		(Out=S)	/	/
			/	(Out=0),flg=S3	/
			/	/	flg＝S1
	else		(Out=X2+X1)	/	/
			/	(Out=X1),flg=S2	/
			/	/	(Out=X1),flg=S2
T>4			(Out=S)	/	/
			/	/	/
			/	/	/
else			(Out=0)	(Out=0)	(Out=0)

flgを
状態変数
として、状態に
移行する

© Japan Embedded Systems Technology Association 2016

リバースモデリングの作業手順

条件		flg		
		S1	S2	S3
無条件実行		T=in1,S=in2	T=in1,S=in2	T=in1,S=in2
T==1		S++	S++	S++
	S<X1	(Out=S)	(Out=0),flg=S3	flg＝S1
	else	(Out=X2+X1)	(Out=X1),flg=S2	(Out=X1),flg=S2
T>4		(Out=S)	/	/
else		(Out=0)	(Out=0)	(Out=0)

うん、
状態遷移表に
なった

これで、やっとレビューができる！！

© Japan Embedded Systems Technology Association 2016

サンプル検証

■ 新人研修での実習（スロットマシンの作成）で、新人の作成したプログラムを検証

- スロットは3つ、ボタンは1つ
- 起動直後は3つのスロットがすべて回転
- すべて回転しているときに、掛け金設定
- 1回目のボタン押下で、掛け金を設定し、一番左のスロットを止める
- 2回目のボタン押下で、真ん中のスロットを止める
- 3回目のボタン押下で、右のスロットを止めると同時に、止まった3つの絵柄を比較し、その結果でもち金を増減
  - 3つとも同じ→掛け金5倍
  - 2つ同じ　→掛け金そのまま
  - すべて異なる→掛け金没収
- 4回目のボタン押下で、3つのスロットをすべて回転している状態に戻す

© Japan Embedded Systems Technology Association 2016

サンプル検証1（意味不明→仕様明確化）

状態変数名が意味不明

状態を0～3の直値で指定

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

条件は 状態2 のときのみ有効

© Japan Embedded Systems Technology Association 2016

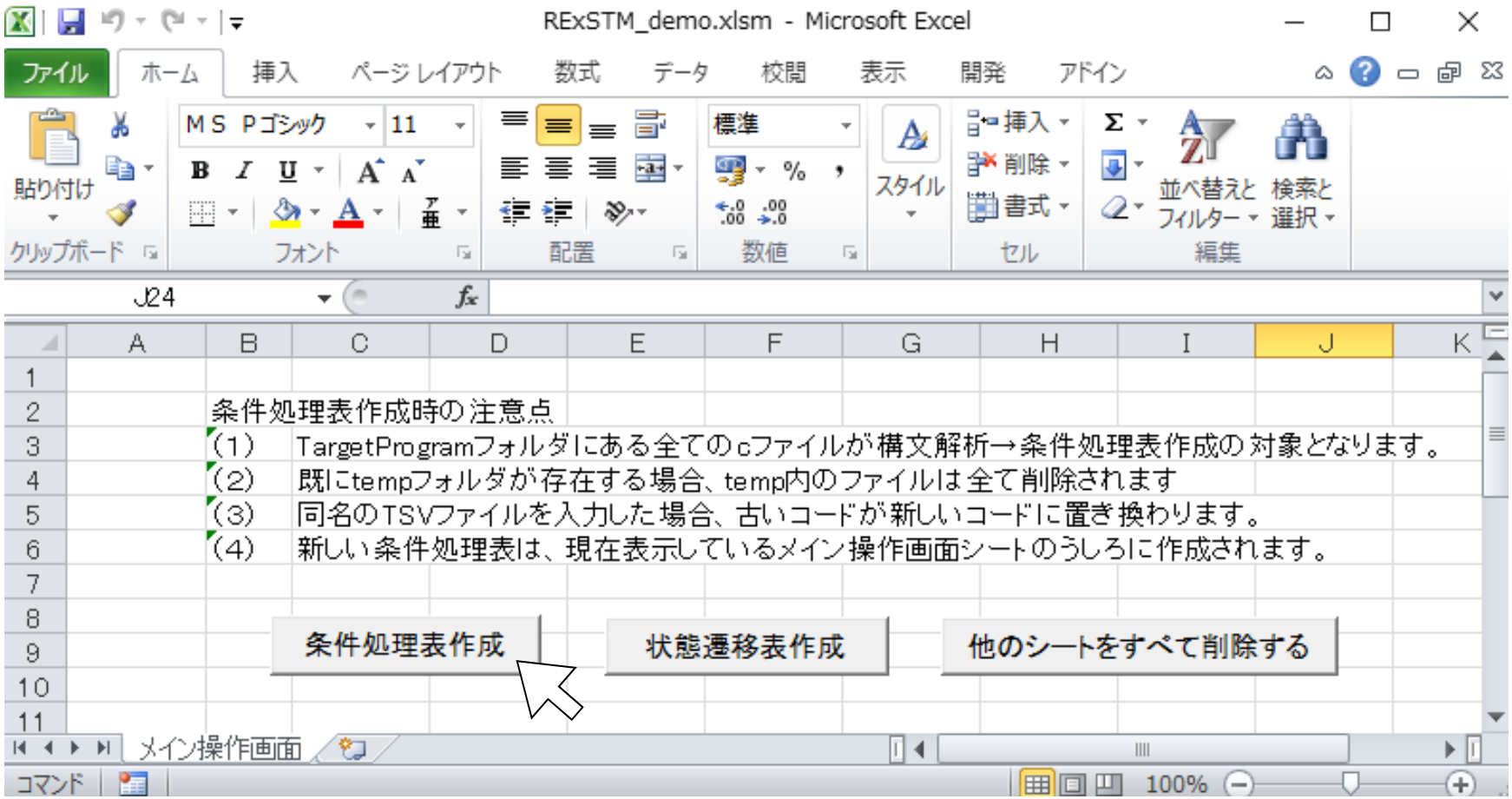
サンプル検証2(実装のブラッシュアップ)

関数	条件	g_pushStatus						
		Rolling	Stop1	Stop2	Stop3	Hit2	Hit3	Miss
Entry_Calculate	L,M,Rがすべて一致	／	／	／	g_pushStatus = Hit3 Total += Bet*5	／	／	
	L,M,Rの2つが一致	／	／	／	g_pushStatus = Hit2 Total += Bet	／	／	／
	else	／	／	／	g_pushStatus = Miss	／	／	／
ENTRY_ChargeStatus		g_pushStatus = Stop1	g_pushStatus = Stop2	g_pushStatus = Stop3	／	g_pushStatus = Rolling	g_pushStatus = Rolling	g_pushStatus = Rolling

状態の粒度が違う
stop3状態での
内部条件とするべき

© Japan Embedded Systems Technology Association 2016

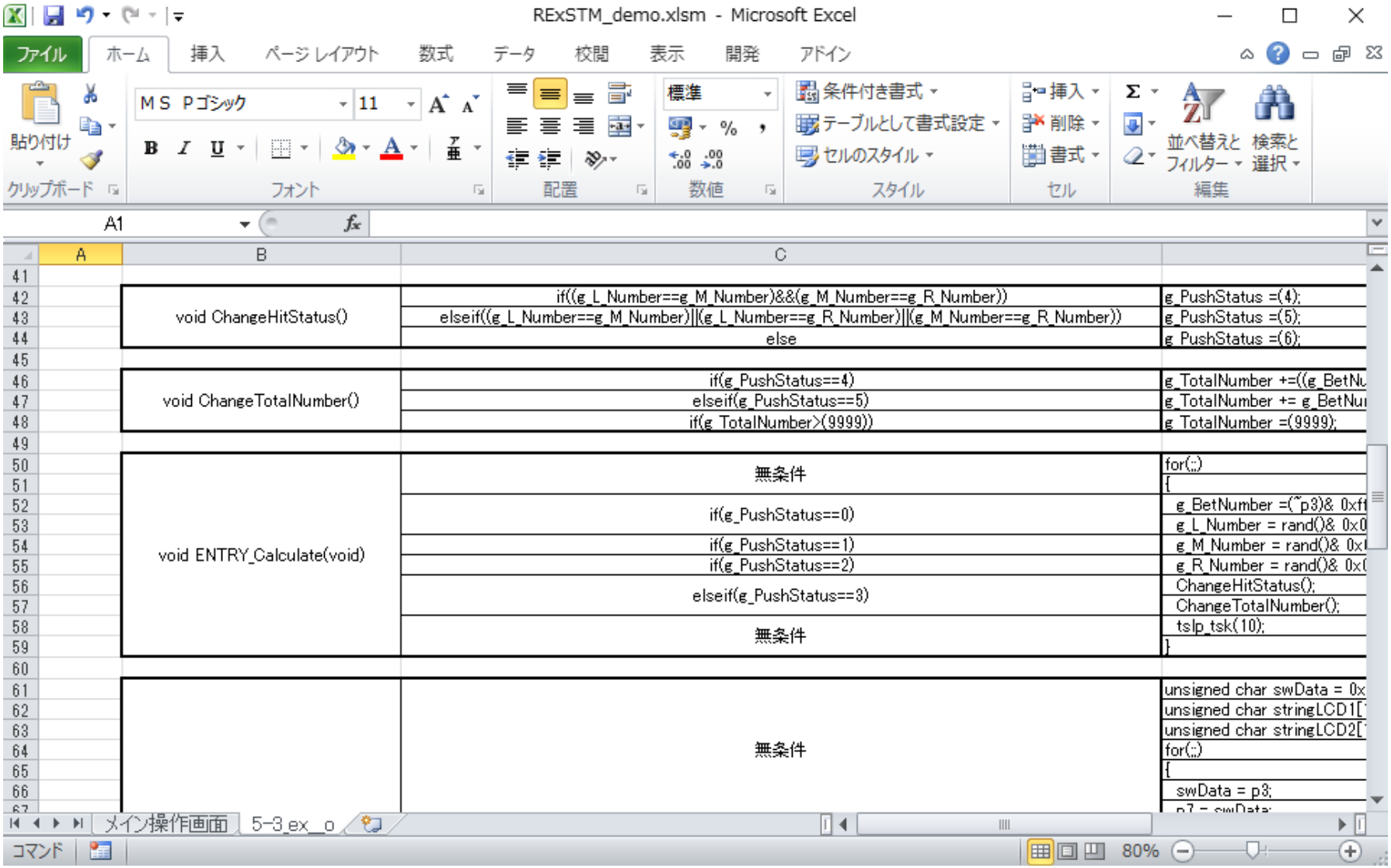
RExSTM for C　条件処理表・状態遷移表作成ツール



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

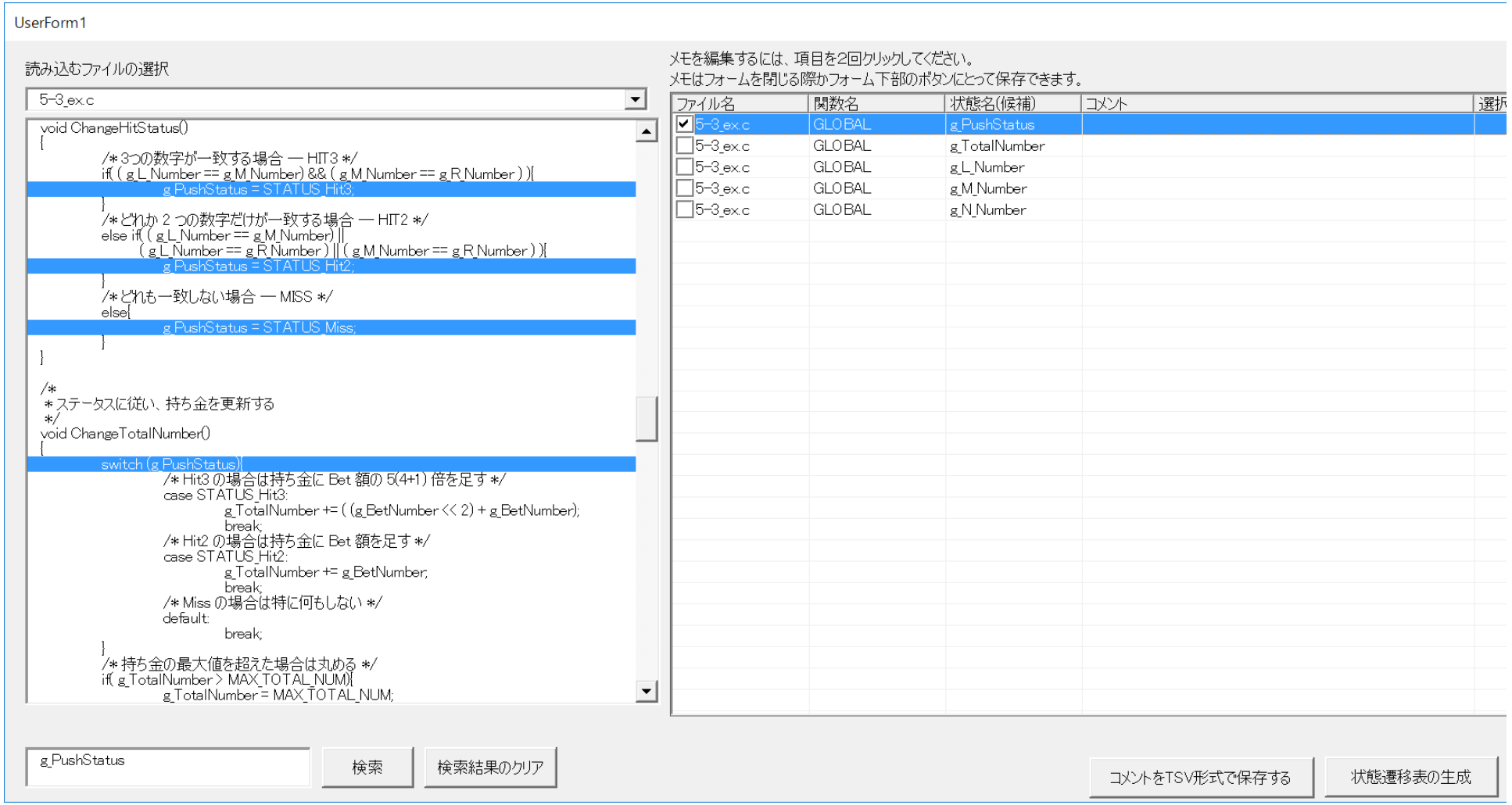
© Japan Embedded Systems Technology Association 2016

RExSTM for C　条件処理表の出力



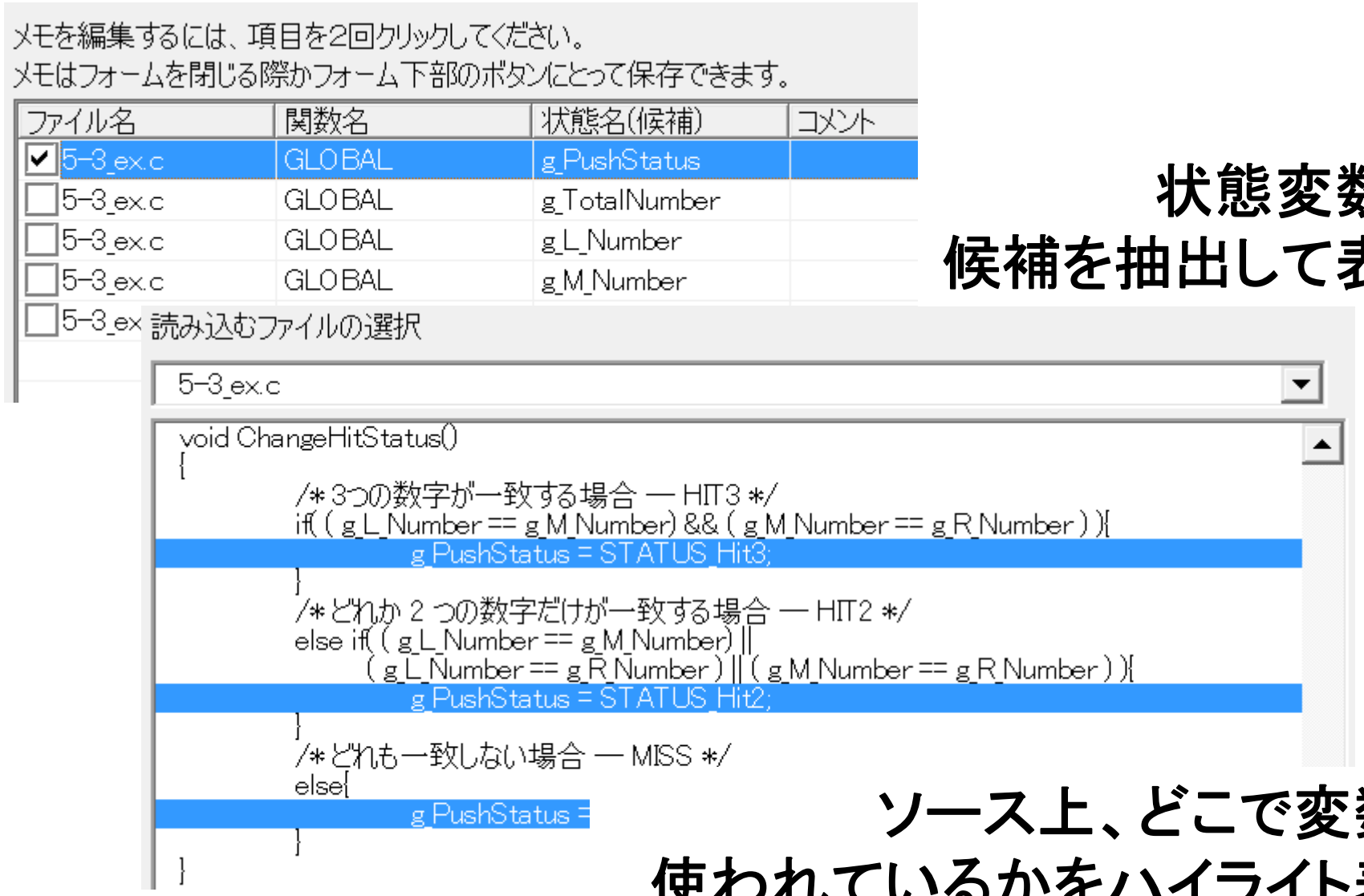
© Japan Embedded Systems Technology Association 2016

RExSTM for C 状態変数の選択



© Japan Embedded Systems Technology Association 2016

RExSTM for C　状態変数の選択

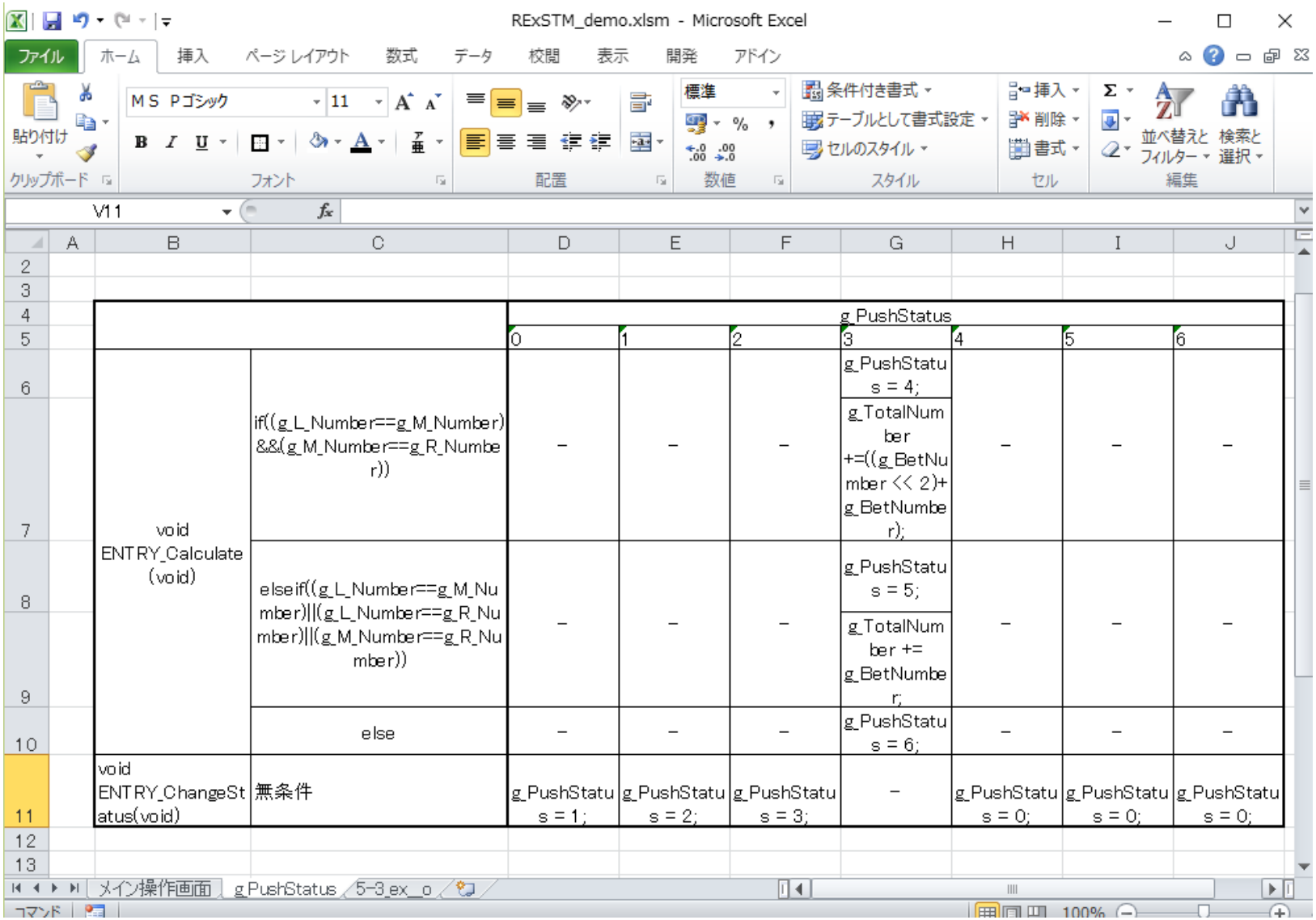


状態変数の
候補を抽出して表示

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

© Japan Embedded Systems Technology Association 2016

RExSTM for C　状態遷移表の出力



			g_PushStatus						
			0	1	2	3	4	5	6
void ENTRY_Calculate (void)	if((g_L_Number==g_M_Number) &&(g_M_Number==g_R_Number))		–	–	–	g_PushStatus = 4; g_TotalNumber +=((g_BetNumber << 2)+ g_BetNumber);	–	–	–
	elseif((g_L_Number==g_M_Number)||(g_L_Number==g_R_Number)||(g_M_Number==g_R_Number))		–	–	–	g_PushStatus = 5; g_TotalNumber += g_BetNumber;	–	–	–
	else		–	–	–	g_PushStatus = 6;	–	–	–
void ENTRY_ChangeStatus(void)	無条件		g_PushStatus = 1;	g_PushStatus = 2;	g_PushStatus = 3;	–	g_PushStatus = 0;	g_PushStatus = 0;	g_PushStatus = 0;

© Japan Embedded Systems Technology Association 2016

RExSTM for C　状態遷移表の作成（手動色付）

		g_pushStatus						
		0	1	2	3	4	5	6
void ENTRY_Calculate	(g_L_Number == g_M_Number) && (g_M_Number == g_R_Number)	–	–	–	g_pushStatus = 4; g_TotalNumber += ((g_BetNumber << 2) + g_BetNumber);	–	–	–
	(g_L_Number == g_M_Number) || (g_L_Number == g_R_Number) || (g_M_Number == g_R_Number)	–	–	–	g_pushStatus = 5; g_TotalNumber += g_BetNumber;	–	–	–
	ELSE	–	–	–	g_pushStatus = 6;	–	–	–
void ENTRY_ChangeStatus	無条件	g_pushStatus = 1;	g_pushStatus = 2;	g_pushStatus = 3;	–	g_pushStatus = 0;	g_pushStatus = 0;	g_pushStatus = 0;

© Japan Embedded Systems Technology Association 2016

手動検証での結果　2015年度の資料より

関数	条件	g_pushStatus						
		Rolling	Stop1	Stop2	Stop3	Hit2	Hit3	Miss
Entry_Calculate	L,M,Rがすべて一致	／	／	／	g_pushStatus = Hit3 Total += Bet*5	／	／	／
	L,M,Rの2つが一致	／	／	／	g_pushStatus = Hit2 Total += Bet	／	／	／
	else	／	／	／	g_pushStatus = Miss	／	／	／
ENTRY_ChargeStatus		g_pushStatus = Stop1	g_pushStatus = Stop2	g_pushStatus = Stop3	／	g_pushStatus = Rolling	g_pushStatus = Rolling	g_pushStatus = Rolling

→同じ形での出力ができている

© Japan Embedded Systems Technology Association 2016

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できる
  - ー　手動と比べて作成時間を大幅に減らすことができる
  - ー　ただし、単純なパターンに限られる

■　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できる
- 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

© Japan Embedded Systems Technology Association 2016

2016年度　活動計画

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

② 普及活動

- 出張セミナー、出張コンサルサービスの展開リバース手順の無償レクチャー

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）のガイド公開

© Japan Embedded Systems Technology Association 2016

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年度はツール要件固めを進めた
  - 自動でできる作業はツールで実行
    - ✓ ソースから条件処理表を生成する
    - ✓ 状態変数候補を抽出、選択→状態遷移表に展開する
  - 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年度はenPiT を利用して実際にツール化した
  - 現状，あるソースコードに対しては適用可能
- 2016年度はツールのブラッシュアップや，解析可能な対象を増やす

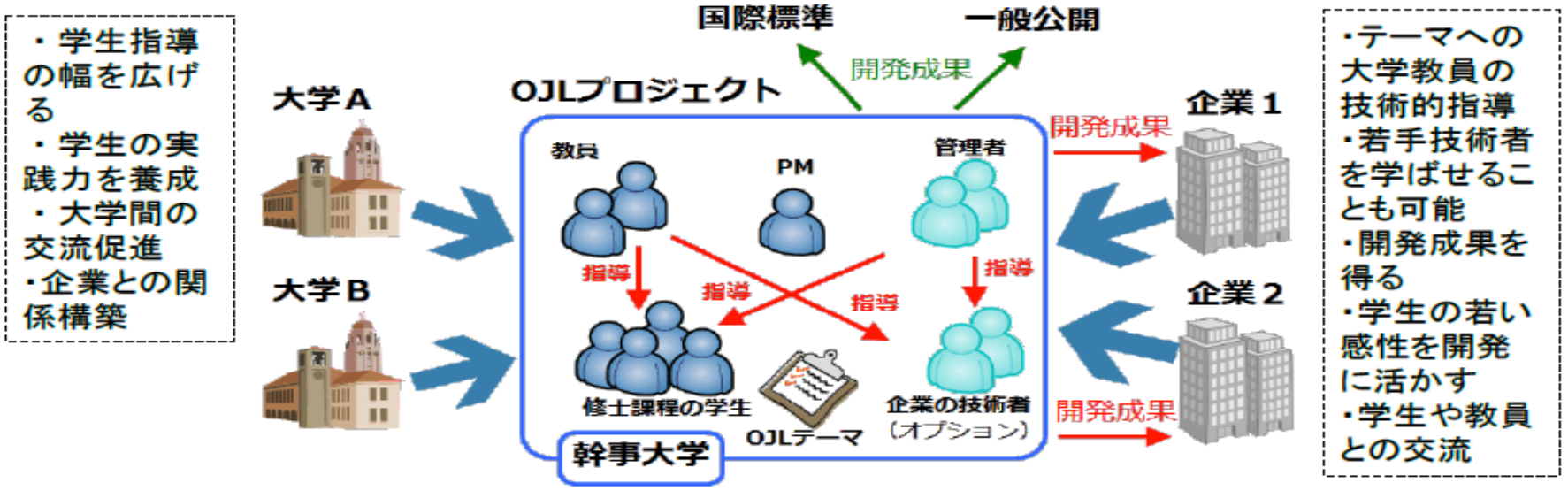
© Japan Embedded Systems Technology Association 2016

①ツール化活動

enPiT Emb 分野・地域を越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
enPiT Emb

On the Job Learning

大学内に企業の開発テーマを持ち込み，学生に参加させ
企業の開発スタイルに準じることを学生に求め，
実践力を養成する育成手法．PBLの発展形態．



© Japan Embedded Systems Technology Association 2016

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　名古屋大学事業

事業紹介 | 発展型OJL | ニュース・イベント | 公開資料 | FAQ | お問い合わせ

home > 名古屋大学事業 > 学生募集中のOJLテーマ

- 発展型OJL
- 特長
- スケジュール
- 合宿スケジュール
- ▸ 学生募集中のOJLテーマ
- 参加申込み方法

学生募集中のOJLテーマ

現在、参加学生を募集している2015年度OJLテーマは、以下の通りです。提案書をご覧の上、参加を希望する方は申込書を提出してください。参加申込みについては、こちらをご覧ください。

- OJL-1. RoboCar 1/10による車載ソフトウェアプラットホームの実証実験
- OJL-2. 自動車向けリモートサービスと攻撃手法の検討
- OJL-3. C言語リバースモデリングツールの開発

OJLテーマ3：C言語リバースモデリングツールの開発

enPiT Emb　提案者：名古屋大学

C言語リバースモデリングツールの開発

- C言語のソースから分岐条件に使っている変数を抽出
- さらにその変数を更新していたら，状態変数候補と見なす
- 抽出した状態変数候補を使って，状態遷移表を作るサポートツール（Excelベース）を生成する

前提知識
- C言語

獲得知識・技能
- リバースモデリング
- 状態遷移
- C言語のトークン解析
- Excelを使うツール

連携企業
一般社団法人 組込みシステム技術協会

NCES 名古屋大学大学院情報科学研究科 組込みシステム研究センター　© Japan Embedded Systems Technology Association 2016　ESOJL　H27

提案書
更新日：2015年5月11日

OJLテーマリストに戻る

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！
- 状態遷移設計研究会　会員募集中！
- 出張セミナー、出張コンサル、各種相談受付中！

© Japan Embedded Systems Technology Association 2016

「レガシーコードの蘇生術」

2016/11/16 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 住長第二ビル３F
TEL: 03(5643)0211 FAX: 03(5643)0212
URL:http://www.jasa.or.jp/TOP/

本書の著作権は一般社団法人組込みシステム技術協会(以下、JASA) が有します。
JASAの許可無く、本書の複製、再配布、譲渡、展示はできません。
また本書の改変、翻案、翻訳の権利はJASAが占有します。
その他、JASAが定めた著作権規程に準じます。

© Japan Embedded Systems Technology Association 2016