拡張階層化状態遷移表（EHSTM）設計手法のススメ

CATS社
渡辺　政彦

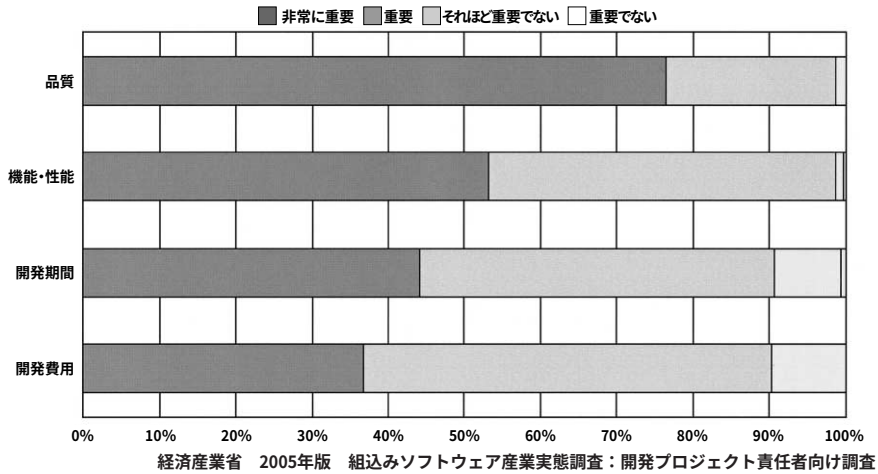
■拡張階層化状態遷移表（EHSTM）設計手法の歴史

EHSTM設計手法を、1987年頃に電力監視制御ソフトウェア開発向け社内設計手法として誕生してから、18年が経過した。EHSTM設計手法は、1992年に手法書として公開され、1998年に現在のEHSTM設計手法Ver.2へと進化した。1980年代後半に起きた、電力システム自動化の波は、急激にソフトウェアを大規模かつ複雑化しながらも、社会的インフラとして高信頼性を求め続けた。この変化を乗り切るべく開発された設計手法がEHSTMだった。今日、ケータイやクルマに組込まれるソフトウェアは500万行を超え、近い将来1000万行に達するとの予測がある。QCD（品質、コスト、納期）が臨界点に達している組込みソフトウェア開発への解決策として、EHSTM設計手法をお薦めする。

■組込みシステムの品質向上に必要なこと

経済産業省2005年度組込みソフトウェア産業実態調査の開発プロジェクト責任者向け調査[1]で実施された「プロジェクトで開発する製品・システムの重要度についてはどのようにお考えですか。」の問いに、「品質」が非常に重要であるとの回答が一番高い（図１）。それでは製品・システムの品質を高めるには何が必要であろうか。MADE IN JAPANが高品質の代名詞であった時代の製品と現在との違いは、ソフトウェア設計の比率である。15年前に設計全体の１割にすぎなかったソフトウェア設計が、今や６割を占める（図２）。当然ながらソフトウェア比率の高さに合った設計手法の導入がなされていなければ、QCDに綻びが出よう。かつて、製品の信頼性は、ハードウェアの信頼性に比重が重く置かれていた。例えば信頼性を表す尺度の１つに、平均故障間隔時間（MTBF：Mean Time Between Failure）[2]、つまり、障害を発生せずに装置が稼動した時間の平均で定義するものがある。ハードウェアの場合は物理的変化や劣化等（例えば熱や消耗等）の問題があるので、このようなモノサシは重要であった。しかしソフトウェアは、物理的変化による信頼性への影響を受けることはほとんどない。

図１　プロジェクトで開発する製品・システムの重要度



経済産業省　2005年版　組込みソフトウェア産業実態調査：開発プロジェクト責任者向け調査

■JIS X 0129：ISO/IEC 9126のソフトウェア品質特性

JIS X 0129：ISO/IEC 9126（図３）のソフトウェア品質特性は以下のように定義される[3]。

機能性：ソフトウェアが使用される目的に合っていること、処理内容が正確であることなどの度合い

信頼性：ソフトウェアが安心して使える度合い

使用性：ソフトウェアの使いやすさ

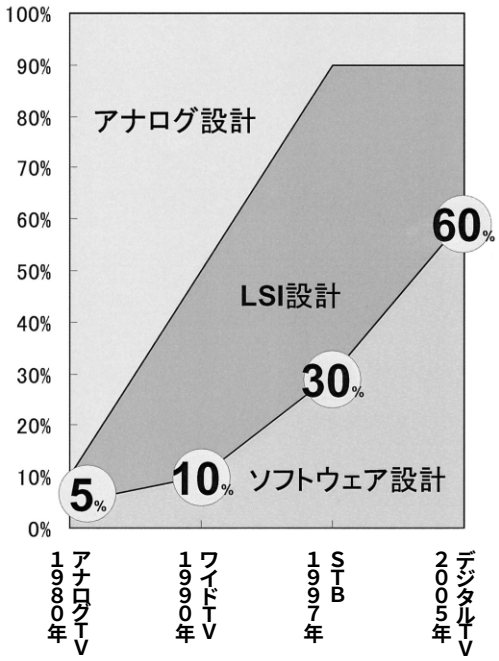
効率性：スループット、レスポンスタイム、ターンアラウンドタイムなどで測定される処理性能

保守性：環境変化に対応するための変更とテストが容易に行えること

移植性：各種プラットフォームで容易に開発できること

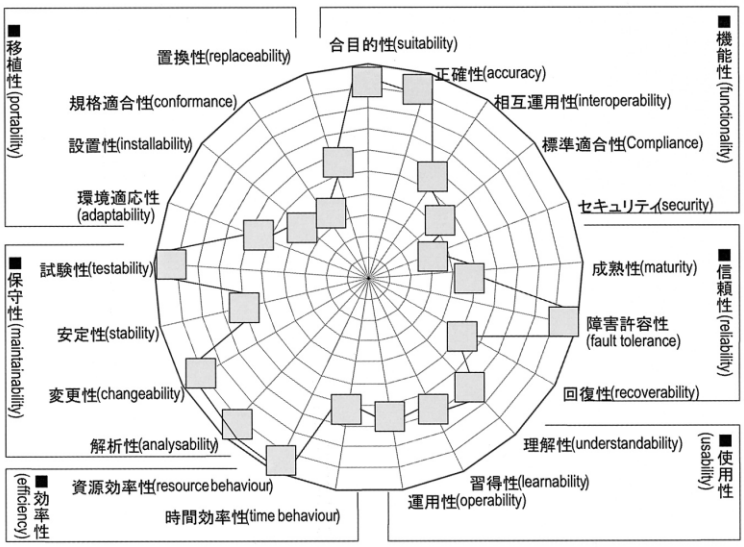
2005年度組込みソフトウェア産業実態調査では、上記６つのソフトウェア品質特性のどれがレビューの重要観点かを調査した。結果は、「要求分析・仕様設計」、「システム設計」、「ソフトウェア設計」の工程では「機能性の観点」が１番目に重要としている（図４、図５、図６）。「ソフトウェア実装」の工程では「信頼性の観点」が１番目に重要としている（図７）。調査結果から上流工程では機能の明確化がレビューポイントで、その後、実装では不具合の作りこみがないようにレビューを行っていると分析できる。JIS X 0129：ISO/IEC 9126では、機能の副特性を合目的性、正確性、相互運用性、標準適合性、セキュリティの５つに分類し、信頼性の副特性には、成熟性、障害許容性、回復性の３つに分類している（図３）。高品質ソフトウェア設計とは、各工程でどの品質特性に主眼を置いてモデリングするかが重要である。全ての品質特性を満足するモデリングは現実には有り得ない。工程間で品質特性観点の違いは、『分析モデル―設計モデル』間の対応がとれないことの要因の1つである。

図２　ソフトが開発の中心になった



日経ビジネス　2005.3.28
特集ソフト大国　出所：松下電器産業

図３　ソフトウェア品質特性パターン

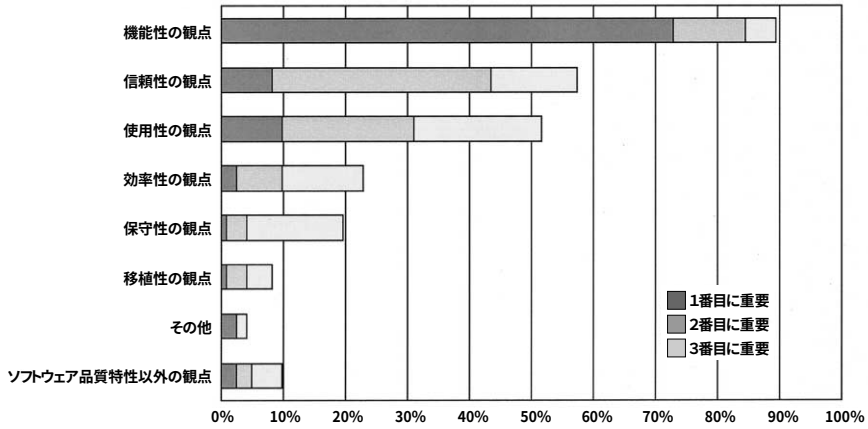


1 URL: http://www.ipa.go.jp/software/sec/download/200506es.php

2 MTBF の公式：MTBF = 稼動時間の合計 / 障害発生回数

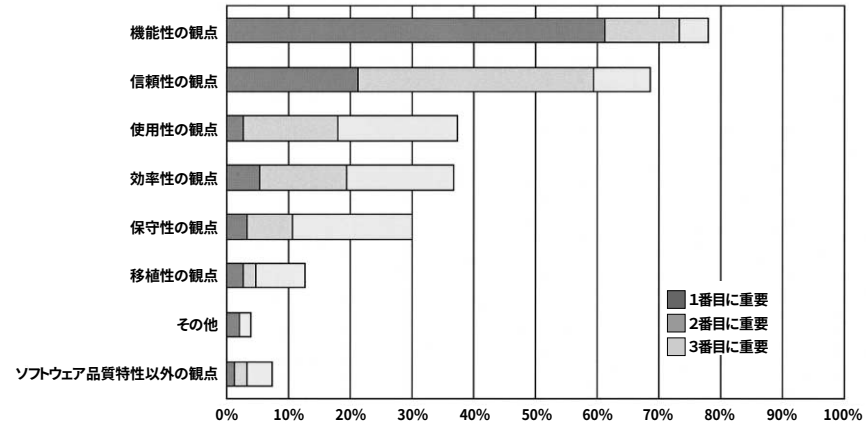
3 2005年度組込みソフトウェア産業実態調査でのJIS X 0129：ISO/IEC 9126に対する説明を抜粋

図４　レビューまたはインスペクションを実施する観点：要求分析・仕様設計



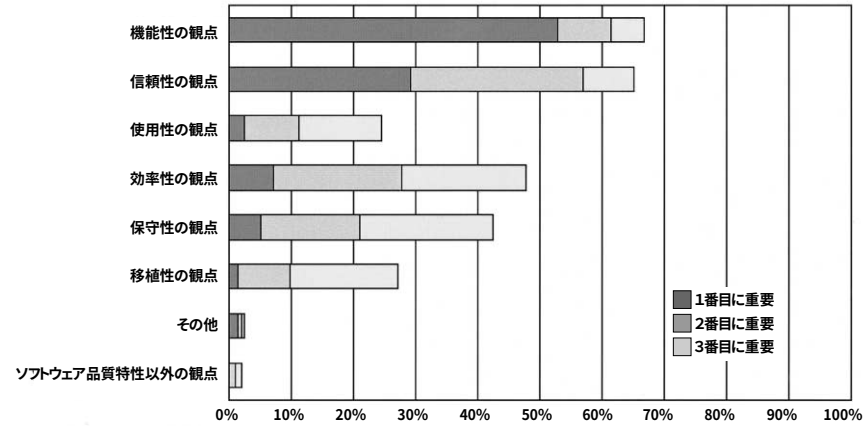
経済産業省　2005年版　組込みソフトウェア産業実態調査：開発プロジェクト責任者向け調査

図５　レビューまたはインスペクションを実施する観点：システム設計



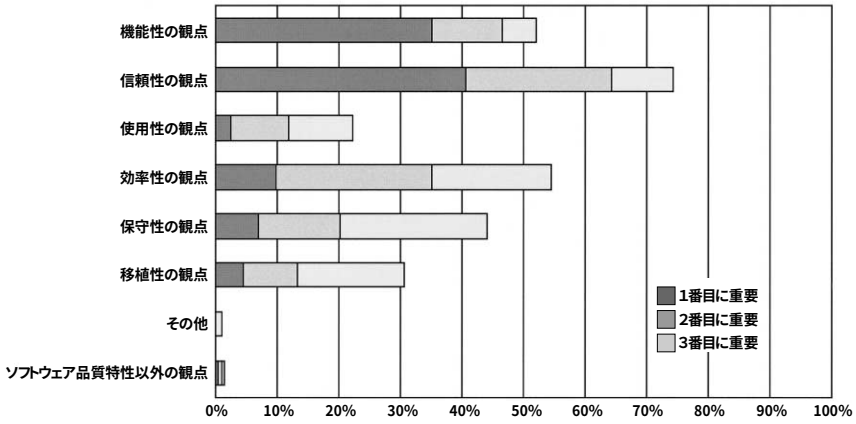
経済産業省　2005年版　組込みソフトウェア産業実態調査：開発プロジェクト責任者向け調査

図６　レビューまたはインスペクションを実施する観点：ソフトウェア設計



経済産業省　2005年版　組込みソフトウェア産業実態調査：開発プロジェクト責任者向け調査

図７　レビューまたはインスペクションを実施する観点：ソフトウェア実装



経済産業省　2005年版　組込みソフトウェア産業実態調査：開発プロジェクト責任者向け調査

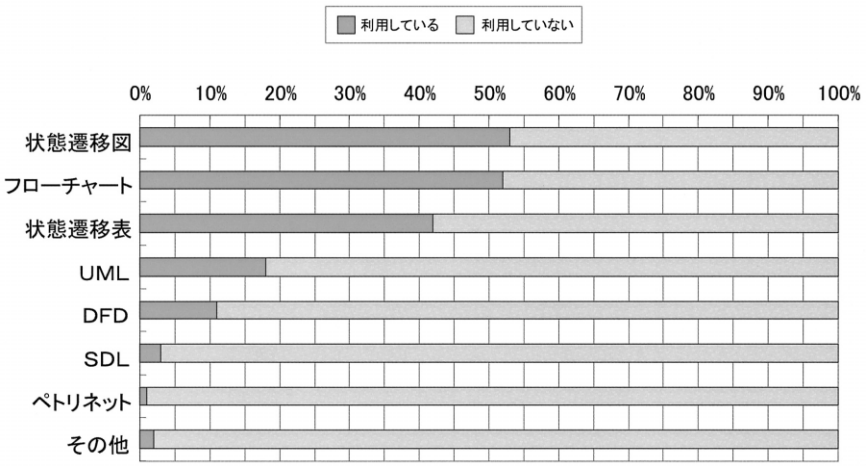
■EHSTMとは

企画要求獲得・要求分析仕様設計・システム設計で使用されるセミフォーマル記述言語に状態遷移図･表が上位を占めている（図８）。状態遷移図が状態遷移表よりも多く使われているが、本稿ではEHSTMを導入することでソフトウェア品質特性（図３）の４つの特性「機能性」、「信頼性」、「保守性」、「効率性」に効果があること示す（図９）。

従来の状態遷移表は、記述対象であるシステムやソフトウェアが巨大化、複雑化すると、状態遷移表自体が巨大化、複雑化する欠点があった。そこで、状態遷移表に簡約化技法を取り入れた表記法がEHSTMである。EHSTMで採用した状態遷移表向け簡約化の代表的な技法は、次の５つである（図10）[4]。①階層化（事象階層化・状態階層化）　②クローン　③遷移型　④アクティビティ　⑤アクションセルNSチャート記述

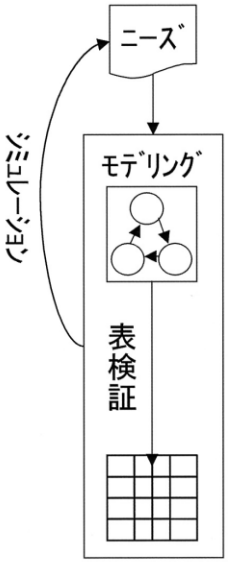
この技法を用いたEHSTMにより、巨大化、複雑化する状態遷移表を簡約化することができた。コニカでは複写機の組込みソフトウェア開発にEHSTMを適用し、従来と比べ83%のセル数を削減できたと報告している。EHSTMは産業界で広く使われるようになり[5]、例えば、電波産業界（ARIB）のIMT2000（3GPP/FORMA）仕様書に採用実績がある。

図８　企画要求獲得・要求分析仕様設計・システム設計の工程で使用したフォーマル記述言語



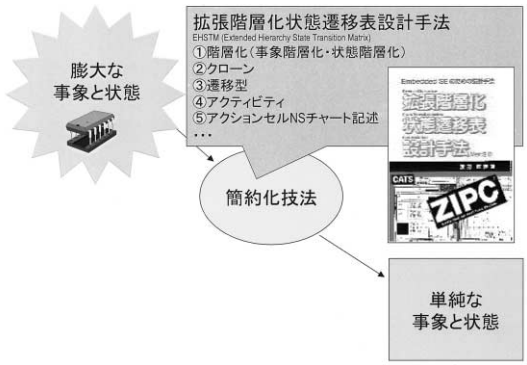
経済産業省　2005年版　組込みソフトウェア産業実態調査：開発プロジェクト責任者向け調査

図9　ソフトウェア品質特性－機能性向上の技法



品質特性	副特性	定義	技法	測定方法
機能性	合目的性	明示された仕事に対する機能の集合が存在し、適切であることをもたらすソフトウェアの属性 仕様通りに動くか？	ﾆｰｽﾞをﾓﾃﾞﾙ化し、このﾓﾃﾞﾙをｼﾐｭﾚｰｼｮﾝすることで、ﾆｰｽﾞと合致しているかをﾃｽﾄする	ﾃｽﾄｹｰｽ
	正確性	正しい結果もしくは正しい効果、または同意できる結果もしくは同意できる効果をもたらすソフトウェアの属性 仕様の抜け・漏れが防止できるか？	状態図から状態表に変換し、空白セルを定義する	空白定義
効率性	資源効率性	ソフトウェアの機能を実行する際の、使用した資源の量及びその使用時間をもたらすソフトウェアの属性 ROM / RAM サイズは？	ﾓﾃﾞﾙからﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞを自動生成する	ROM / RAMｻｲｽﾞ
信頼性	障害許容性	ソフトウェアの障害部分を実行した場合、又は仕様化されたインタフェース条件に違反が発生した場合に、仕様化された達成のレベルを維持する能力をもたらすソフトウェアの属性 想定外のことに対応できているか？	想定外事象をelseイベントとする	ﾓﾃﾞﾙ内のelseｲﾍﾞﾝﾄ定義数
保守性	解析性	欠陥もしくは故障の原因の診断又は改訂すべき部分の認識に必要な労力に影響するソフトウェアの属性 不具合発生時に解析しやすいか？	事象No、状態Noのリングバッファログ	解析時間

図10　拡張階層化状態遷移表（EHSTM）設計手法



■機能性－合目的性

「機能性－合目的性」とは、仕様通りにソフトウェアが動くかということである。このためには、状態遷移モデルの全遷移パスを実行して検証することが望ましい。しかしながら、全ての遷移パスを抽出するのは、人手では経済的に不可能であり、ツールが必要となる。IPA（情報処理振興事業協会）重点領域情報技術開発事業により開発されたPerfectPassは、EHSTMを読み込み、全ての遷移パスを自動的に抽出する。抽出したパスはテストシナリオとしてZIPCのATV機能により自動的にテストを実行し、検証する。ツールによる自動化をしても、現在のパソコンの処理能力では相当な時間を必要とする。地球シミュレータ並みの処理能力か、グリットコンピューティングの本格適用が望まれるところだ。

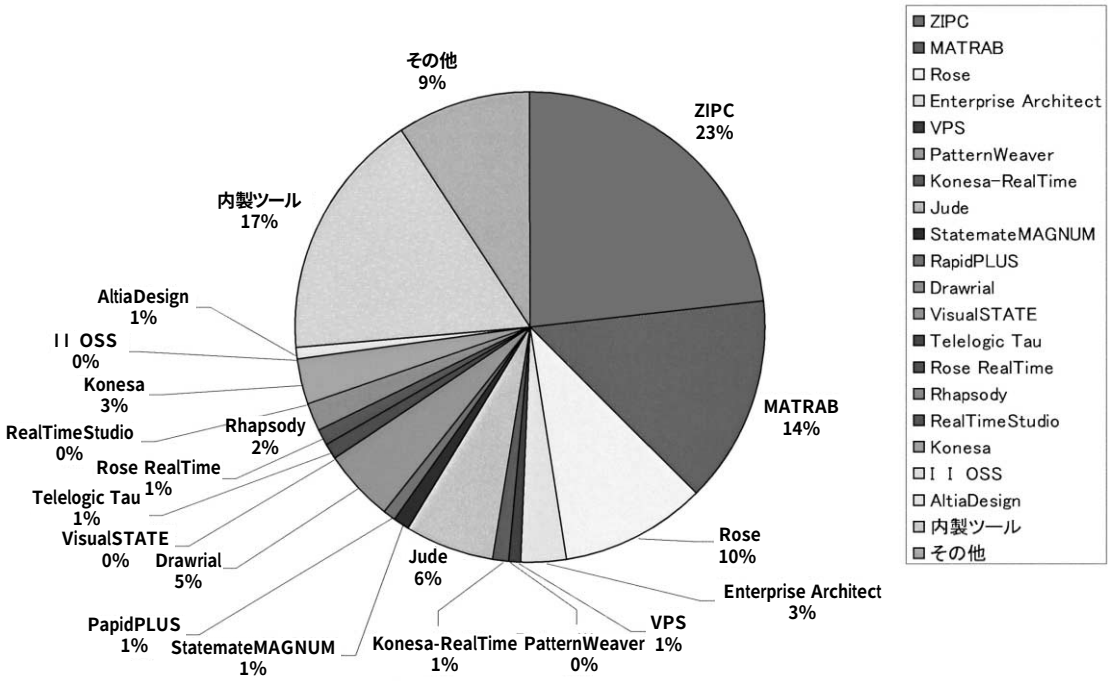
モデルを動的に実行し、テストを網羅的に行うことは非常にコストがかかる。そこで、モデルを静的に解析し、モデルが特定の性質を満たしているかを検査する検証技法がある。これがモデル検査である。キャッツ社－九州大学－福岡知的クラスター研究所では共同でEHSTMのモデル検査を研究し、その成果であるZIPC－GARAKABU（製品コード名）連携を第８回組込みシステム開発技術展で発表した。モデル検査はモデルテストと組み合わせることで、膨大なテスト時間を品質低下させることなく削減できる解の１つである。

■機能性－正確性

状態遷移図モデルでは、どのように振舞いをするかといった視点に重きがおかれる。振舞いが正しい結果をもたらすためには、仕様上の漏

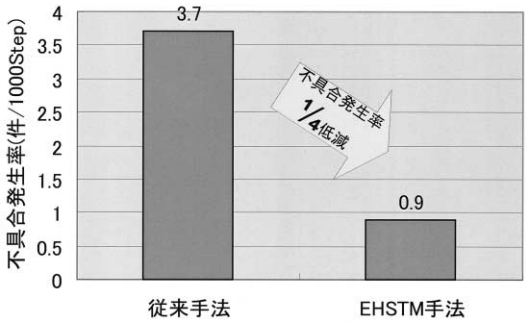
図19　組込みシステム向けCASEツール　最新導入率

2004年　JEITA（社団法人　電子情報技術産業協会）調べ



れ、抜けを発見しなくてはならない。状態遷移表の特性として、事象と状態の組合せから振舞いの漏れ、抜けを発見しやすい。しかし、ISO/IEC11411:JISX0131で定義される状態遷移表は、HarelやUMLなどの状態遷移図を変換できない。なぜならばISO/IEC11411:JISX0131状態遷移表は、状態階層化や並列状態などを取り扱えないからである。EHSTMでは状態階層化や並列状態などに対応しているので、ZIPCのコンバータによって各種[6]の状態遷移図モデルをEHSTMに変換できる。「機能性－正確性」に有効だった事例としてコニカの複写機にEHSTMを適用し、不具合発生率が1/4に低減された事例がある（図11）[7]。

図11　拡張階層化状態遷移表（EHSTM）設計手法の事例



渡辺・滝・黒畑：状態遷移表設計によるソフトウェア開発プロセス改善
コニカ㈱オフィスドキュメントカンパニー
機器開発統括部　第3開発センター：ZIPC WATCHERS Vol.4

■効率性－資源効率性

システム設計、ソフトウェア設計、ソフトウェア実装と工程が下流になるにつれて、レビューまたはインスペクションの「効率性視点」の重要度が増してくる（図4、図5、図6、図7）。状態遷移モデルを分析モデルといった上流工程の文書化だけの目的ではなく、設計モデルにも適用できるようにするには、「効率性－資源

4　詳細は東銀座出版：拡張階層化状態遷移表設計手法:ISBN4-89469-004-7を参照のこと

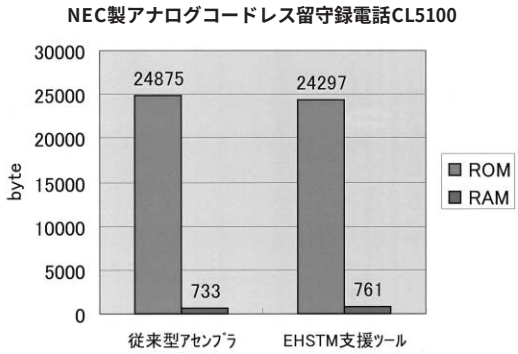
5　1998年から2004年までEHSTM支援ツールZIPCは、組込みCASEツールのトップシェアを獲得している（JEITA調べ）図19参照

6　IBM社ROSE, I-Logx社StatemateMAGNUM, TMW社MATLAB/Stateflowなど

7　ZIPC Watcher Vol.4

効率性」が重要である。そこで状態遷移モデルを支援するCASEツールの自動生成するコード効率が、設計モデル適用可否を左右する。EDAツールが効率の良いゲートサイズにしのぎを削るように、CASEツールにも効率の良いコード生成が望まれ続ける。ZIPCはアセンブラと比較しても遜色のないCコードを自動生成した事例がある（図12）。

図12　拡張階層化状態遷移表（EHSTM）設計手法支援ツールの事例



ZIPC Ver.4.0 プロトタイプ事例:インターフェース1996年8月号

■信頼性－障害許容性

ソフトウェア信頼性の副特性である障害許容性を、「ソフトウェアの障害部分を実行した場合、又は仕様化されたインタフェース条件に違反が発生した場合に、仕様化された達成のレベルを維持する能力をもたらすソフトウェアの属性」と定義している。つまり、ソフトウェアが想定外のことに対応できているかということになる。

EHSTMではelse事象を用いることで、想定外のことに対応しやすい（図13）。想定していない事象が飛び込んできた場合にどうするかをあらかじめ設計できるのである。テストファーストなどで活用されているアサーションを、状態モデルの事象レベルで実現したものといえる。アサーションは、C、C++、JAVAといったプログラミング言語仕様で定義された例外を発生させる構文である。アサーションが検出されると、アサーションの式と発生したファイル名、行番号をエラー表示してコアダンプする。マイクロソフトの製品を使っていればたびたび見たことがあるだろう。Junitなどの自動ユニットテストフレームワークは、アサーションの入った

図13　信頼性：障害許容性

■0 CD		電源ＯＮ ■電源ＯＮ	電源ＯＦＦ
	E＼S	0	1
電源	0	=>電源ＯＦＦ	=>電源ＯＮ
else	想定外の動作		

■0.1 電源ＯＮ		トレー格納 ■トレー格納	トレー排出
	E＼S	0	1
イジェクト	0	=>トレー排出	=>トレー格納
else	想定外の動作		

■0.1.1 トレー格納		停止中	再生中	一時停止中	巻き戻し中	早送り中
	E＼S	0	1	2	3	4
再生	0	=>再生中				
停止	1		=>停止中	=>停止中		
一時停止	2		=>一時停止中	=>再生中		
早送りボタン押下	3		=>早送り中			
早送りボタンリリース	4					=>再生中
巻き戻しボタン押下	5		=>巻き戻し中			
巻き戻しボタンリリース	6				=>再生中	
電源	7		=>停止中	=>停止中	=>停止中	=>停止中
イジェクト	8		=>停止中	=>停止中	=>停止中	=>停止中
else	想定外の動作					

テストケースを自動的に実行するものだ。アサーション技術がテストファース開発プロセスを支えている。ハードウェア開発で用いられるVHDLやVerilog-HDLはアサーションを言語仕様として持っていないため、ソフトウェアのようなアサーションベース検証ができなかった。しかし、PSL（Property Specification Language）/SugarやSystemVerilogがアサーションに対応し、アサーションベース検証はEDAでも広がりを見せている。

■保守性－解析性・変更性・試験性

状態遷移表は２次元テーブルで情報を管理しているので、リングバッファ形式なりのログバッファに現在の状態と事象Noを記録するだけで、不具合発生時に解析しやすい（図14）。上記のアサーションと組み合わせて使用することで、例外処理が発生に到るまでの過去の事象、状態の遷移を解析できるので、不具合要因発見に役立つ。

状態遷移表は表であるがゆえに変更がしやすい（図15）。状態遷移図では状態の追加、削除といった変更がしづらく、ケアレスミス的な不具合が混入しやすい。

状態遷移表の設計書をそのままチェックリストに使えることから、試験性が高いモデルを実現できる（図16）。

図14　保守性：解析性

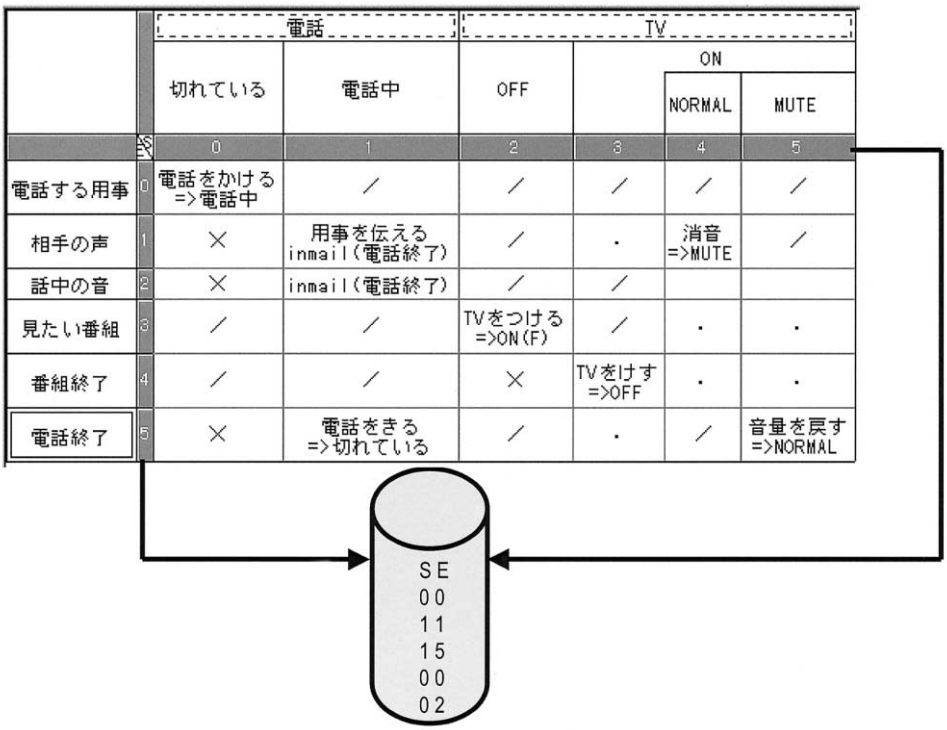
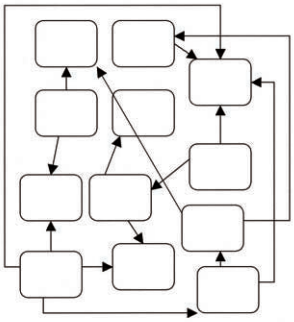


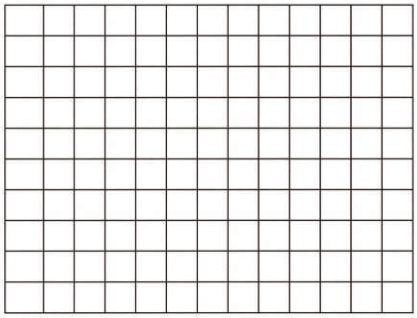
図15　保守性：変更性

図



○STDは概要を掴むのには適している（分析フェーズ）
×異常・例外ケースを入れると図が収拾つかない
×STDはモレ・ヌケやすい
×ドキュメントの変更が手間取る（配置空間や結線を意識する）

表

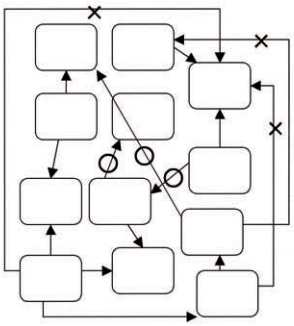


×概要が掴み難い

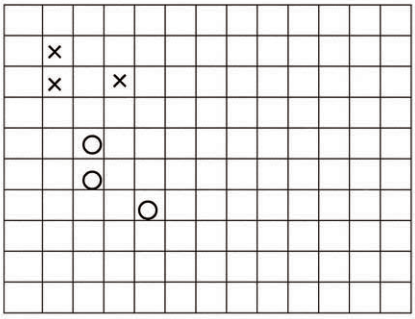
○異常・例外ケースを入れられる（設計フェーズ）
○STMはモレ・ヌケを防止する
○STMは行列の変更だけで済む

図16　保守性：試験性

図



表



■レガシィ問題と部品化再利用

モデルベース開発への移行を阻害するものは既存資産である。EHSTMを用いてモデルベースを行う際には、コールドスポットとホットスポットを良く見極めることが大切だ。良く変更を受ける部分をEHSTMの事象、状態として設計し、変更が少ないコールドスポットをEHSTMのアクションからブラックボックス的に呼び出す。EHSTMモデルは、ソフトウェアの部品化、再利用を促進する。EHSTMでは事象欄をインタフェース設計として、外部からアクセスされる部分を表記する。状態欄は外部からは隠蔽されるタスク、システム内部の状態である。アクション欄にはブラックボックス化されたモジュール部品を配置する。EHSTMを用いることで、外部事象、内部状態、再利用モジュールを明確にモデリングできる（図17）。

EHSTMは「いつ」「どこで」「何をする」かが表で組合せとして見ることができるので、レビューやインスペクションにも効果がある。そこで、中国なインドなど日本語でのコミュニケーションを密にとりづらい海外アウトソースの活用にEHSTMを適用するユーザが増えている。

モデルベースへ移行する投資対効果は、完全なソフトウェア再利用とCASEツールを用いた自動化プロセスが実現できれば、４年後には30倍+25倍の投資効果が期待できる（図18）。

是非、皆さん、EHSTMモデルベース開発で、一歩前進してください。

図17　表による部品化再利用促進

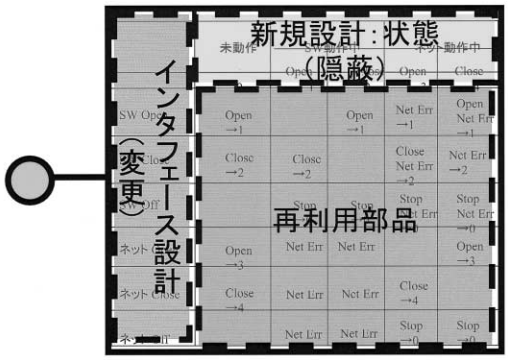


図18

技術	4年利用後の1ドルあたりの投資効果($)
完全なソフトウェア再利用	30
CASE	25
ソフトウェアの品質測定	17
ソフトウェアの見積もりツール	17
正規の設計インスペクション	15
正規のコードインスペクション	15
オブジェクト指向プログラミング	12
ソフトウェアの生産性測定	10
ソフトウェアプロセスアセスメント	10
機能的尺度	8

Capers Jones著：Assessment and Control of Software Risks:Prince-Hall, 1994,
邦訳：島崎・富野「ソフトウェア病理学」共立出版, 1995