Vol.2012-SE-175 No.9
2012/3/15

実行順序依存性のある状態遷移表に対する動作理解支援手法

古家直樹† 山岸正知†† 鈴木康文†
川上真澄† 小川秀人†

状態遷移表は，遷移動作を網羅的に記述可能であり，仕様の抜け漏れを防止することが出来るため，今日のソフトウェア開発で広く適用されている．その一方で，実行時の動作シーケンスを把握し難いという課題がある．特に，状態遷移動作の結果が実行順序によって変わるという，実行順序依存性を持つ状態遷移表の場合，状態遷移動作を直感的に理解することが困難である．そこで，状態遷移表から特定の動作シーケンスを生成することによる，動作仕様理解支援手法を提案する．

A Support Method for Understanding Actions from State Transition Matrix Having Execution Order Dependency

Naoki Furuya† Masatomo Yamagishi††
Yasufumi Suzuki† Masumi Kawakami†
and Hideto Ogawa†

A State transition matrix is widely used in software development since it can exhaustively describe the specification of software and avoid misspecification. However, it is difficult to understand intuitively each action sequence from a state transition matrix, especially if it has an execution order dependency. We suggest a support method for understanding actions, which is based on creating each action sequence from the state transition matrix.

1. はじめに

近年，ソフトウェアの規模が増大している一方で，短期開発が求められており，ソフトウェアの品質確保が困難になりつつある．その中で，ソフトウェア開発の上流の設計工程での品質確保が重要であると言われている[1]．設計工程におけるソフトウェア品質向上策として，ソフトウェアの搭載されたシステムやソフトウェアモジュールの振る舞いを状態遷移表や状態遷移図で記述する手法が広く使われている．

状態遷移表は網羅的に状態遷移動作を記述出来るため，仕様の抜け漏れを防止することが可能であるが，表を参照する開発者が具体的な状態遷移動作を追うことは困難であると言われている[1]．特に，複数回の状態遷移動作の順序次第で結果が変わるという，実行順序依存性を有する状態遷移動作を状態遷移表で記述した場合，開発者が頭の中で動作シーケンスを組み立てて，動作を直感的に理解することは困難である．

そこで，遷移表から特定の動作シーケンスを生成することによる動作理解支援手法を提案する．本報告では，カーナビゲーションシステム（以降，カーナビと称す）の画面遷移動作を例に説明する．

2. 実行順序依存性のある状態遷移動作の課題

2.1 実行順序依存性のある状態遷移動作

図 1 に，実行順序依存性のある動作例として，カーナビの割込み画面遷移動作を示す．これは，地図画面表示中にオーディオを ON をした直後に，VICS[a]を受信した場合の動作例である．カーナビは，一割込みイベント終了後に再び同一画面を表示することに備えて，表示画面を退避するスタックを保持している．図 1 において，VICS 画面を一定時間表示した後再び表示される画面は，スタックで最も上に格納されているオーディオ画面となる．なお，時間の経過や，カーナビのユーザが UI に表示された画面上の「戻る」ボタンを押すことで，スタック内の画面を表示する動作を戻り動作と称す．このスタックへの画面の貯まり方によって，割込みイベント終了後の UI の表示画面が変わるので，実行順序依存性のある動作シーケンスが起こり得る．複数回の状態遷移後の戻り動作結果の直感的な理解は，ＵＩの画面遷移と画面遷移毎のスタックの変化を正確に把握することが必要なため困難である．

†株式会社日立製作所 横浜研究所
††クラリオン株式会社 製品開発統括部
a VICS（Vehicle Information and Communication System:道路交通情報通信システム）は，財団法人道路交通情報通信システムセンターの登録商標です．

Vol.2012-SE-175 No.9
2012/3/15

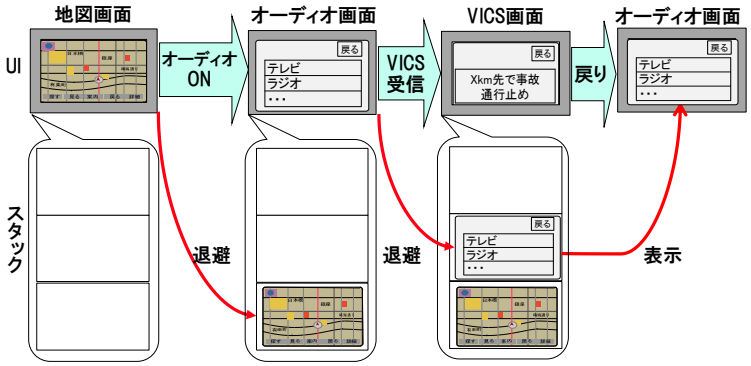

図 1　カーナビの割込み画面遷移動作の例

2.2　状態遷移動作の理解困難性が引き起こす問題

実行順序依存性のある状態遷移動作の理解困難性は，製品開発の現場において次の課題を招く．例えば，状態遷移動作を行うソフトウェアの仕様を，ソフト設計工程において，状態遷移表で設計する場合を考える（図 2）．ソフト設計者は，機能仕様書を基に状態遷移表を作成する．続いて，実装工程において，実装者が状態遷移表を基にソフト実装を行う．その後，単体・結合テスト工程において，ソフトウェアの動作検証が行われる．テスト設計者は状態遷移表を基にテストケースを設計する．

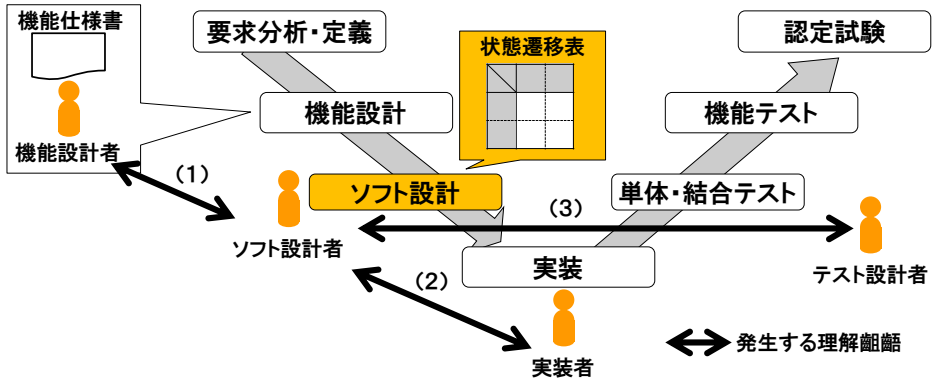

図 2　状態遷移ソフトウェアの各開発工程で発生する理解齟齬

状態遷移表を作成したソフト設計者以外に，同表を参照する機能設計者，実装者，テスト設計者にとって，状態遷移表に実行順序依存性がある場合は状態遷移動作を正しく理解することが困難である．そのため，開発者間の状態遷移仕様の理解齟齬が発生し，開発の後工程（機能テスト以降の工程）への状態遷移ソフトウェアの不具合流出を生じる．不具合流出の原因となる問題を，表 1 に示す．

表 1　状態遷移ソフトウェアの不具合流出の原因となる問題

	関係者（開発者）	理解齟齬	問題
(1)	ソフト設計者と機能設計者	ソフト設計レビューで，機能設計者が状態遷移表の仕様誤りを見逃す	誤った状態遷移表を実装者に渡し，それを基にソフトが作成される
(2)	ソフト設計者と実装者	実装者が状態遷移表を読み間違える	誤ったソフトが作成される
(3)	ソフト設計者とテスト設計者	テスト設計者が状態遷移表を読み間違える	誤ったテストケースを設計して，不良を見逃す

そのため，ソフト設計工程において，状態遷移表の仕様誤りを摘出することが重要である．

3.　提案手法

3.1　状態遷移表の分析

カーナビソフト開発では，ソフト設計者は割込みイベントによる画面遷移動作を網羅的に把握することを目的に，状態遷移表を用いてソフト設計を行っている．

本報告では，実際のカーナビ開発で用いられている状態遷移表，及びカーナビのスタックを簡易に表したモデルを用いて，提案手法の説明及び検証を行う．表 2 に，簡易化したカーナビの状態遷移表を示す．この表の各セルには，ある UI 上の表示画面(以降，現画面と称す)中に，イベント（割込みイベント）が発生した時の，UI の次の表示画面（以降，次画面と称す）と，変数 x に対する操作が記述されている．ここで，x はスタックに貯まっている画面数に依存する変数である．また，各割込みイベント発生時に割込む画面は一意に定まっており，表 3 に示す通りである．

Vol.2012-SE-175 No.9
2012/3/15

表 2　カーナビの状態遷移表

現画面（現状態）＼イベント	a	b	c
A	x=0のとき A/x保持 x>0のとき 戻り画面/x=x-1	B/x保持	A/x=x+1
B	B/x=x+1	x=0のとき B/x保持 x>0のとき 戻り画面/x=x-1	C/x保持
C	A/x保持	C/x=x+1	x=0のとき C/x保持 x>0のとき 戻り画面/x=x-1

次画面（次状態）/変数xに対する操作

表 3　イベントと割込む画面の対応

イベント	割込む画面
a	A
b	B
c	C

表 2 の状態遷移表を分析する．破線で囲んだ各セルは，戻り動作が発生していることを意味する．すなわち，x=0（スタックに画面が存在しない場合）は，戻り動作でも現画面と同一の画面が次画面となる．x>1 の場合は，戻り動作により，スタックの一番上に格納されている画面が次画面となる．このように変数 x（スタック内の画面数）によって，画面遷移動作結果が変わるので，本状態遷移表は実行順序依存性を持つ．なお，その他のセルにおける記述と，実際の画面遷移動作との対応は，表 4 に示す通りである．カーナビの画面動作では，現画面と割込みイベントとの関係によって画面動作が異なり，必ずしも割込み画面が次画面にならないことや，現画面又は割込む画面がＵＩに表示されない場合にはスタックに退避される時と破棄される時とがあることが特徴である．

表 4　状態遷移表の記述と，意味する画面動作との対応

次画面	変数 x に対する操作	意味する画面動作	
		現画面	割込む画面
現画面と異なる	x=x+1	スタックに退避	ＵＩに表示
	x 保持	破棄	ＵＩに表示
現画面と同じ	x=x+1	ＵＩに表示継続	スタックに退避
	x 保持	ＵＩに表示継続	破棄

3.2　複数回遷移時の課題

ここで，複数回の画面遷移動作を行った後の，画面と変数 x，及びスタックの結果を考える．図 3，及び図 4 に，共に初期画面は A，変数 x の初期値 0（すなわち，スタックに何も画面が入っていないことを意味する）で，3 回の状態遷移を含む動作シーケンス 1，2 を示す．なお，ある初期状態からの複数回の状態遷移動作の結果を，動作シーケンスと称す．動作シーケンス 1 と 2 とでは，次のようにイベント a とイベント c の発生順序が異なっており，3 回の状態遷移後の結果も異なる．

- 動作シーケンス 1 のイベント:a→c→b
- 動作シーケンス 2 のイベント:c→a→b

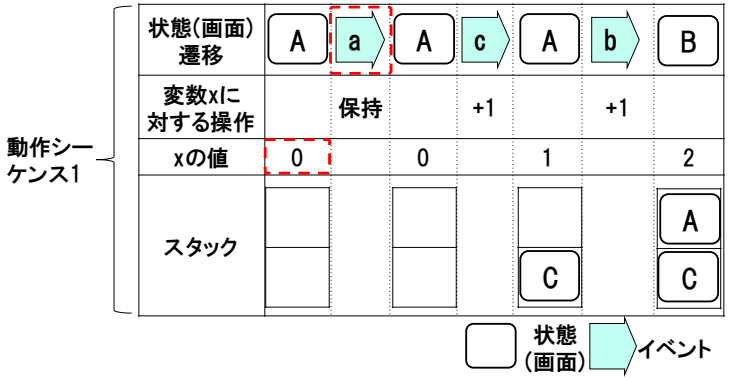

図 3　動作シーケンス 1

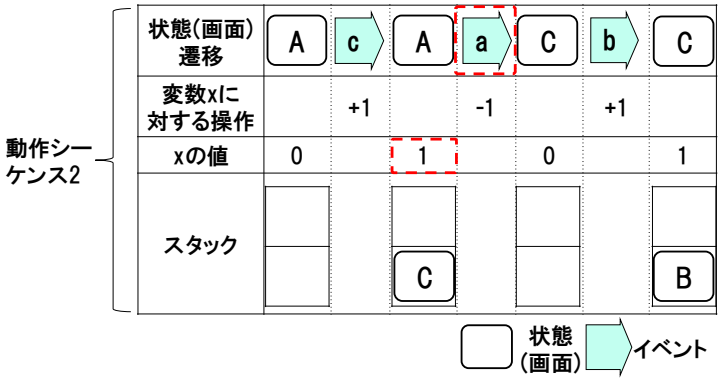

図 4　動作シーケンス 2

これは，状態（画面）A にて，イベント a が発生した時に，x の値によって戻り動作を行う/行わないの分岐が存在するためである．x の値の変化はイベントの発生順序（実行順序）に依存するので，本状態遷移は実行順序依存性が存在する．本例の状態遷移表から状態遷移を理解するには，スタックの状態を表す変数 x の値の変遷も合わせて理解することが必要となり，実行順序依存性のある動作を直感的に理解し難い．

3.3　動作シーケンス生成による理解性向上の提案

図 2 に示した開発者間の理解齟齬を防止するには，開発者間で状態遷移動作の共通理解を持つことが必要である．しかし，実行順序依存性のある状態遷移表は，可読性が低い．そこで，状態遷移表から動作シーケンスを生成して，状態遷移表の理解性向上の一助とすることを提案する．ソフト設計レビュー時に状態遷移表と合わせて動作シーケンスを参照することで，機能設計者による状態遷移表の仕様誤りの見逃しを削減する．また，実装者，及びテスト設計者も，それぞれ状態遷移表からの実装，テストケース設計時に動作シーケンスを参照し，状態遷移動作の理解性向上を図る．

また，ユーザが指定した初期状態名と，複数回のイベント名とから動作シーケンスを生成するツール（動作シーケンス生成ツール）を開発する．図 5 に動作シーケンスと，動作シーケンス生成ツールの概要を示す．同ツールは，状態遷移表から状態遷移仕様（状態遷移先，及び操作）を基に動作シーケンスを生成する．そのため，各開発者が頭の中で動作シーケンスをイメージする場合に比べ，短い時間で正確な動作シーケンス生成が可能である．

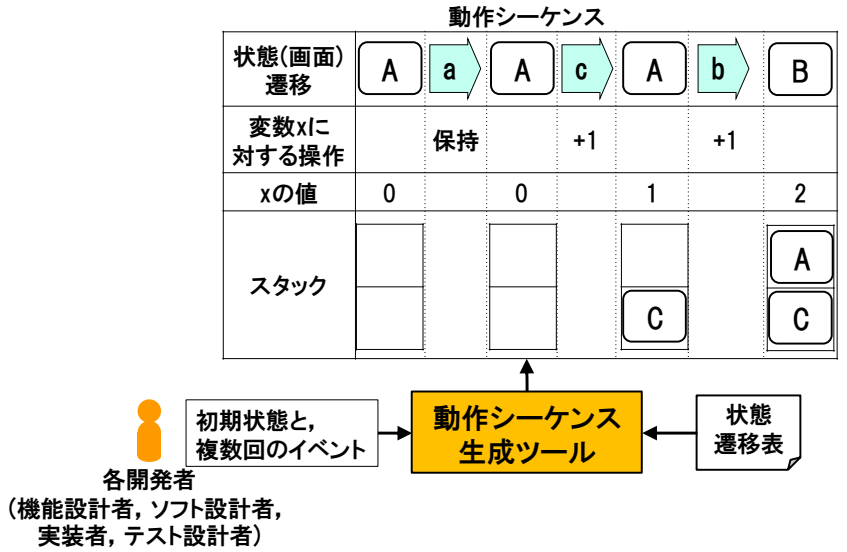

図 5　動作シーケンスと，動作シーケンス生成ツールの概要

4.　評価

本節では，提案手法の評価を行う．4.1 節で動作シーケンス生成に指定するイベント回数の評価，4.2 節で動作シーケンス生成に要する時間の評価，4.3 節で不具合摘出効果の見積もりについてそれぞれ説明する．

4.1　動作シーケンス生成に指定するイベント回数の評価

動作シーケンス生成時に指定する適切なイベント回数を評価した．表 2 の状態遷移表で，戻り動作を行う(現画面,イベント)=(A, a), (B, b), (C, c)の 3 つのセルに変数 x の条件分岐が存在する．そこで，条件分岐を全網羅するのに必要なイベント回数を割り出した．表 5 に表 2 の状態遷移表から生成される，イベント 2 回の全組み合わせを示す．なお，初期状態ではスタックに画面がなく，x の値は 0 と仮定した．

表 5　状態遷移表（表 2）から生成される，イベント 2 回の全組み合わせ

状態遷移1　状態遷移2

No.	初期画面（状態）	イベント1	画面2	x	イベント2	画面3
1	A	a	A	0	a	A
2	A	a	A	0	b	B
3	A	a	A	0	c	A
4	A	b	B	0	a	B
5	A	b	B	0	b	A
6	A	b	B	0	c	C
7	A	c	A	1	a	C
8	A	c	A	1	b	B
9	A	c	A	1	c	A
10	B	a	B	1	a	B
11	B	a	B	1	b	A
12	B	a	B	1	c	C
13	B	b	B	0	a	A
14	B	b	B	0	b	B
15	B	b	B	0	c	B
16	B	c	C	0	a	A
17	B	c	C	0	b	C
18	B	c	C	0	c	C
19	C	a	A	0	a	A
20	C	a	A	0	b	A
21	C	a	A	0	c	A
22	C	b	C	1	a	A
23	C	b	C	1	b	C
24	C	b	C	1	c	B
25	C	c	C	0	a	A
26	C	c	C	0	b	C
27	C	c	C	0	c	C

表 5 において太枠で示した項目は，2 度目の状態遷移で戻り動作を行うセルを通る動作シーケンスである．画面 2 からの状態遷移において，画面 2={画面 A，画面 B，画面 C}のそれぞれのケースで，x=1 と x=0 の双方の分岐を網羅している．

これは，スタック動作では，一度の状態遷移での変数 x の変化量は最大で 1，戻り動作を行うセルにおける条件分岐の境界値は x=0 のため，一度目の状態遷移で x=1，0 の双方の状況を作成しておき，二度目の状態遷移でその双方の状況から戻り動作を行えば，分岐を網羅出来ることによる．

従って，初期状態とイベント 2 回の合計 3 因子の全組み合わせを網羅すれば，表 2 の状態遷移表の条件分岐を全て網羅出来る．すなわち，イベント回数 3 回以上の動作シーケンスも生成は可能であるが，分岐を網羅するためのイベント回数は 2 回で十分である．

なお，画面数（状態数），イベント数が増えた場合でも，一度の状態遷移での変数 x の最大変化量が 1，境界値は x=0 である限りは，イベント 2 回で全ての分岐が網羅可能である

4.2　動作シーケンス生成時間の評価

カーナビの状態遷移表からの，二度の画面遷移を含む動作シーケンス生成時間を，人手の場合とツール生成の場合とで比較する．表 6 に，二度の画面遷移を有する動作シーケンスの 1 シーケンスの生成所要時間を示す．人手の場合，二度目の画面遷移結果の導出には，一度目の画面遷移よりも時間が掛かる．これは，スタックの状況を踏まえて状態遷移結果を考えなくてはならないためである．一方ツールでは，人手の約 100 倍高速に動作シーケンスを生成出来る．なお，人手の所要時間は，報告者自らがカーナビの状態遷移表からの画面遷移動作試行を行った結果であり，ツールの所要時間は標準的な性能を持つ PC にて導出する場合の見積もり値である．

表 6　2 度の画面遷移を有する 1 動作シーケンスの，生成所要時間

	人手	ツール
一度目の画面遷移	5秒	0.1秒
二度目の画面遷移	15秒	0.1秒
動作シーケンス生成所要時間	20秒	0.2秒

なお，カーナビの状態遷移表は，画面数は 30，割込みイベント数は 45 である．そのため，二度の画面遷移を含む全動作シーケンス数は，

30×45×45=60,750(件)

となる．仮にこの全件を生成する場合，所要時間(動作シーケンス件数×動作シーケンス 1 件の生成所要時間)は，表 7 に示す通りとなる．

表 7　2 度の画面遷移を有する全動作シーケンスの，生成所要時間

	人手	ツール
1動作シーケンス生成所要時間	20秒	0.2秒
全動作シーケンス(60,750件)の生成所要時間	337時間	3.4時間

全件の動作シーケンス生成は，ツールの場合でも 3 時間以上掛かることになる．加えて，生成された動作シーケンス 60,750 件を全件確認するのは困難であり，生成する動作シーケンスを絞り込むなどの工夫が必要になる．これについては，5.3 節で見解を述べる．

4.3　不具合摘出効果の見積もり

最後に，不具合摘出効果の見積もりとして，本報告の手法をカーナビ開発のソフト設計レビューに適用した場合に摘出可能な不具合の割合を算出した．先ず，カーナビソフト設計部の不具合管理システムに登録されている，ある製品の不具合リストの中で，機能テスト，及び認定試験工程で発見された割込み画面遷移に関する不具合を抽

出した．次に，個々の不具合の件名と内容を調査した．その結果，機能テスト，及び認定試験工程で発見されていた割込み画面遷移の不具合の約半数を，ソフト設計レビューにて摘出可能である見込みを得た．なお，摘出不可能な不具合が存在するのは，状態遷移表に記述されないため，動作シーケンスの生成が不可能であった画面遷移動作が存在するためである．

5. まとめ

5.1 結果と効果

状態遷移表は動作を網羅的に把握出来るが，個々の状態遷移を追うことが困難という課題があった．特に，実行順序依存性を伴う場合，直感的に遷移表から動作を理解することは困難であり，それによる開発者間の動作仕様の理解齟齬が問題となる．そこで，状態遷移表から，動作シーケンスを可視化することによる仕様理解支援法を考案した．本手法をカーナビソフト開発のソフト設計レビューに適用した場合，従来後工程（機能テスト，及び認定試験工程）で発見されていた割込み画面遷移に関する不具合の約半分を摘出可能との見積もり結果を取得した．なお，本報告では，スタックを有するカーナビの画面遷移ソフトを例に説明したが，本手法は実行順序依存性を有するシステムに広く適用可能である．

5.2 関連研究

状態遷移設計支援ツールの例として ZIPC[2][3][4], 画面遷移設計ツールの例として，GENWARE[5]や Drawrial[4][6]がある．また，文献[7]にて，デジタル家電を対象としたモデルベース開発環境の CE-Cube を紹介している．これらのツールや開発環境は，いずれも状態遷移（画面遷移）表や図の上で遷移したセルを塗り潰すことで，状態遷移動作の確認を行うことが出来る．しかし，複数回の遷移動作の実行順序を追うという観点では, 本報告で示した実行順序で並べて表示する形式のほうが理解容易性は高い．

また，文献[8]にて，状態遷移表から動作シーケンスを導出して，それを基に自動テストを行う技術を紹介している．また，キャッツ社のツールの PerfectPass[9]では，状態遷移テスト設計補助のために，ユーザが設定した条件に従って，状態遷移表から状態遷移動作シーケンスを抽出が可能である．テスト実行の順番の検討を補助するものである. [8][9]のいずれも, 本報告と同等の動作シーケンスを生成する技術を有するが，テスト工程での活用を目的としており，本報告の技術とは目的が異なる．

5.3 今後の課題

本取り組みにより，実行順序依存性を有し，直感的な動作理解が困難である状態遷移表からの動作理解支援を可能とした. 本手法は, カーナビの画面遷移動作に限らず，実行順序依存性のある状態遷移表に対して広く適用可能である．

今後の課題として，状態遷移表の大きさ，動作シーケンスに指定するイベント回数が増大するにつれ，生成する動作シーケンス数が増大することに対する対策が必要である．これに対しては，

- 案 1.状態遷移表の中で, 仕様変更のあったセルを含む動作シーケンスを優先的に生成する
- 案 2.各動作シーケンスの複雑度を何らかの観点で計測し，複雑度順に表示する

等の対策が考えられる．案 2 に関しては，本報告例では，個々の動作シーケンスにおける変数 x の変化した距離（変化量の絶対値）を計測し，距離が大きい順に表示することが挙げられる．

また，生成した動作シーケンスを自動テストのテストスクリプトに変換し，自動テストを行うことでテスト効率化を図るなど，動作シーケンスのソフト設計工程以外の工程への応用が考えられる．

参考文献

1) 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター: 組込みソフトウェア開発における品質向上の勧め[設計モデリング編], pp.52-53, アイティメディア株式会社,（2006）
2) 渡辺正彦：拡張階層化状態遷移表設計手法，キャッツ株式会社，(1998)
3) キャッツ株式会社製品情報「ZIPC」
http://www.zipc.com/product/zipc/
4) 安部田章 : ZIPC+Drawrial による家電組込みソフトウェア開発の効率化，ZIPC WATCHERS，Vol.10，pp.32-37(2006)
5) 株式会社アイ・エル・シー製品情報「GENWARE」
http://www.ilc.co.jp/commodity/genware3/index.html
6) キャッツ株式会社製品情報「Drawrial」
http://www.zipc.com/product/drawrial/
7) 川上真澄，小川秀人：デジタル家電ドメインに特化したモデルベース開発環境，情報処理学会論文誌，Vol.52，No.12，pp.3184-3191(2011)
8) 保坂信幸: 状態遷移表を用いた状態遷移テストの自動化，Japan Symposium on Software Testing in Tokyo 2010
9) キャッツ株式会社製品情報「PerfectPass」
http://www.zipc.com/product/perfect_pass/