ソフトウェアエンジニアリングシンポジウム 2015
IPSJ/SIGSE Software Engineering Symposium (SES2015)

プロブレムフレームに基づく状態マシン仕様の設計支援システム

市川杏子† 紫合治†

組み込みシステムの開発において，開発者は，システムの仕様を満たすのみならず，それらの持つ外部環境についてももれなく把握しておく必要がある．本稿では，プロブレムフレームの考え方を用いて，ドメインプロパティと要求から，システムの仕様の状態マシンを自動的に生成する手法を提案する．また，提案手法に基づいた，状態マシン仕様の設計支援システムを開発した．

Design Support System for State Machine Specification of Problem Frames

KYOKO ICHIKAWA† OSAMU SHIGO†

In development of embedded system, developers have to not only prevent the specifications of the system, but also comprehend things which are affected by the system. This paper proposes the method of creating the state machine design automatically from domain property and requirement. In addition, it develops state machine design support system based on the method.

1. はじめに

近年の技術の著しい発展にともない，電子レンジや冷蔵庫などの電化製品，銀行の ATM など，組み込みシステムを必要とする電子機器，装置が大幅に増加している．しかしながら，組み込みシステムの開発において，制御対象機器や装置の仕様を満たしながら，適切に動作をするシステムを設計することは手間や時間を必要とする．さらに，機器や装置は想定外の振る舞いやエラーを起こす可能性があり，開発者はシステムを分析，設計する際には，機器や装置を含めた，システムによって影響を受けるすべての事象について，もれなく把握することが重要となる．

本論文では，この問題を解決するために，システムを設計するより先に，システムを取り巻く外界の環境そのものを分析する手法である，プロブレムフレーム[1]に着目した．また，プロブレムフレームの考え方に基づき，外界の環境と，それに対する振舞いを規定することで，システムの仕様を自動的に生成する手法を提案する．さらに，提案手法を元にした，システムの状態マシンの設計支援システムを開発した．

我々の先行研究[2]として，プロブレムフレームに基づきシステムの仕様をユーザが手動で状態マシンを設計し，作成した状態マシンを分析するシステムがあるが，本論文では，システムの仕様を自動的に生成することで，記述間違いや見逃し等の誤りを防いだり，ユーザが気付かなかった可能な遷移を発見することができる．

また，もう一つ先行研究[3]として，プロブレムフレームに基づきシステムの仕様を自動的に生成し，テキストで示すシステムがあるが，本論文では，テキストだけではなく状態遷移図も同時に作成することにより，視覚的に直観的にシステムの仕様を把握することができる．

以下に，2 で関連研究について述べ，3 でプロブレムフレームについて説明し，4 で提案手法のアイディアを述べ，5 で提案手法によるシステムの概要を説明し，6 で実例を用いた本システムの評価を行い，最後に 7 でまとめと今後の課題について述べる．

2. 関連研究

プロブレムフレームは，Michel Jackson によって提唱され，本[1]が出版されて以降，プロブレムフレームの拡張や進化についての研究がすすめられた．Cox[4]等は，これらの研究の全貌をまとめ，プロブレムフレーム研究のロードマップを発表している．

ツール化について，Ourusoff[5]は，教育の立場も踏まえて，現場の技術者がプロブレムフレームを適用する場合には，ツール支援が不可欠であると述べている．プロブレムフレームの研究者向けのツールとして，OpenPF[6]がある．これはテキスト表現からプロブレム図を生成したり，要求や仕様の規定をイベント計算(Event Calculus)で表現し，それから複数の副問題の合成の妥当性等をチェックしたりなどができるツールである．イベント計算は述語論理式をベースとした記述で，汎用性がありどのようなフレームに対しても適用できるが，反面，現場の技術者がそのまま使うのは難しいと思われる．Colombo 等[7]や Hatebur 等[8]は UML のクラス図によってプロブレムフレームのプロブレム図やコンテキスト図のメタモデルフレーム向けのツールを生成している．これは，プロブレム図に関するツールであり，個々のドメインプロパティや要求の記述に対しての支援まではない．

† 東京電機大学大学院 情報環境学研究科
Graduate School of Information Environment, Tokyo Denki University

また，実務者向けのプロブレムフレーム用のツールとして，UML を活用する方式が提案されている[9][10]．これらは，既存の UML ツールを使って，問題を UML のクラス図や状態マシン図で記述し解析する方法で，既存 UML ツールの機能がそのまま活用できる．しかし，UML は解の方式を規定するのが主な目的であり，それを問題そのものの記述に適用するのは難しい．たとえば，外部のドメインと解となるマシンがともにクラスとなってしまい，その区別が明確でなくなる．また，プロブレム図における要求要素は，クラス図では表せない．クラス図に出てくる要素は，あくまでも機器やプログラム要素などの実体を伴うものであり，要求のような実体を伴わない要素は表せない．さらに，プロブレムフレームでの共有現象が，クラスのメソッドやフィールドになり，実装を意識した表現となってしまう．

要求からの仕様生成の研究としては，Seater 等[11]の problem transformations，Rapanotti 等[12]の problem reduction，Li[13]の problem progression などがある．これらはともに，問題世界の深いドメイン(マシンとは，いくつかのドメインを経由した間接的な関係しかもたないドメイン)に対する要求の表現に現れる要求現象を，対応する仕様現象に置き換えることにより，要求記述を徐々にマシンに近い(浅い)ドメインの現象で表現するように変換していく方式である．ここで，要求現象から仕様現象を取り出すのは，ドメインプロパティの記述を使う．要求やドメインプロパティの記述として，Seater 等は Alloy[14]の記述を，Rapanotti 等は Gentzen 風の定理証明記述を使っている．このため，変換の手法は汎用的で，どのフレームにも適用できる．ただ問題によっては，いつも深いドメインから始めるのがよいわけではない．

3. プロブレムフレーム

プロブレムフレームとは，システムの開発において，システムを設計する前に，システムによって制御される部品や機器・装置などの外部環境を分析し，それらがどのように振る舞うべきかを規定するという考え方である．

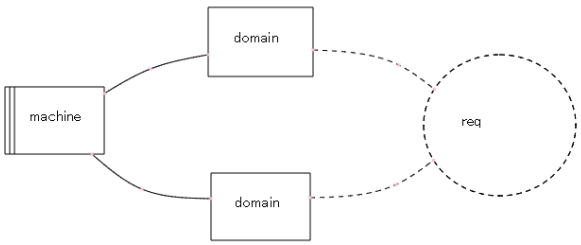

図 1　プロブレム図
Figure 1　The problem diagram.

プロブレムフレームにおいて基本となるものが，プロブレム図である(図 1)．プロブレム図ではシステムを，開発の対象である「マシン」，外部環境となる部品や装置などの制御対象の「ドメイン」，ドメインの振る舞いを規定する「要求」の３つに分け，それぞれ縦線２本を持つ四角，四角，破線の楕円で表現する．基本的なプロブレム図では，マシンと要求がそれぞれ一つずつ，そしてドメインが複数個存在する．本論文では，簡単なため，基本的なプロブレム図を用いるものとする．マシンとドメインを結ぶ実線は「インタフェース」と呼ばれ，マシンとドメイン間のイベントを規定する．ドメインと要求を結ぶ破線は「要求参照」と呼ばれ，ドメインの状態を規定する．ドメインは，自身を中心とするインタフェースと要求参照の関連を規定する，「ドメインプロパティ」を持つ．組み込みシステムの場合においては，ドメインプロパティは状態マシンで表現することが可能であることが多いため，本論文ではドメインプロパティを状態マシンとして規定する．

4. 提案手法

漏れなく最終的なシステムの状態マシンを生成するためには，各ドメインの入出力遷移で遷移可能なすべての状態の組み合わせを，総当たりで考えればよい．しかし，すべての組み合わせの中には，システムの仕様を満たさない状態の組み合わせが現れたり，また，コストも非常に大きくなる．

この問題を解決するために，本研究では”安定状態”という概念を用いることにした．安定状態とは，開発するシステムの仕様が許す各ドメインの状態の組み合わせで構成される．安定状態は，いずれかのドメインの状態に入力イベントが発生しない限り，次の安定状態には遷移しない特徴を持つ．安定状態とはつまり，システムの持つ機能の概要を示すものである．

以上のことを踏まえ，開発するシステムの仕様を自動生成するための流れは，以下の通りとなる．

① システムのプロブレム図の作成
② 各ドメインのドメインプロパティを状態マシンとして定義
③ 要求を安定状態として与える
④ 安定状態と入力遷移のみの状態マシンを自動生成する
⑤ ④で得られた状態マシンから，不要な遷移を排除する制限の付与
⑥ ⑤を元に，④で得られた状態マシンに出力遷移を加え，開発したいシステムの最終的な仕様の状態マシンを再生成する

このうち、④，⑥はツールにより自動化ができる．④で自動生成された状態マシンに，遷移先のない安定状態が作成されるなどのエラーが発生した場合は，前のステップに戻り、②のドメインプロパティの修正や③の安定状態の追加を促す．これらの手順を繰り返すことにより，最終的なシステムの仕様を得る．

4.1 システムのプロブレム図

開発したいシステムをプロブレム図で表現することにより，システムの概要を把握する．

4.2 各ドメインのドメインプロパティの定義

4.1 で作成したプロブレム図より，ドメインプロパティを状態マシンとして作成する(図 2).

状態は，要求参照で定義された状態名を各 1 回使用し，出力遷移あるいは入力遷移のいずれも出入りしない状態が存在してはならない．ただし，遷移先が自分自身であっても良い．初期状態は二重丸で表し，その他の状態は一重丸で表す．

入出力遷移は，インタフェースで定義された入出力イベント名を各 1 回以上使用し，入力遷移を実線の矢印，出力遷移を破線の矢印で表す．入力遷移とは，ドメインがマシンに対してイベントを渡す遷移で，出力遷移は，マシンがドメインに対してイベントを渡す遷移である．

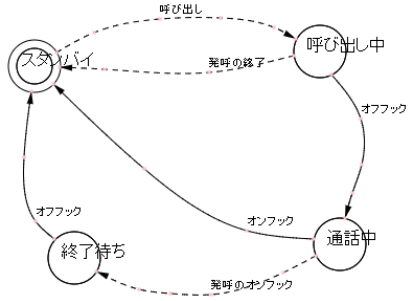

図 2 ドメインプロパティの状態マシン
Figure 2 State machine of domain property

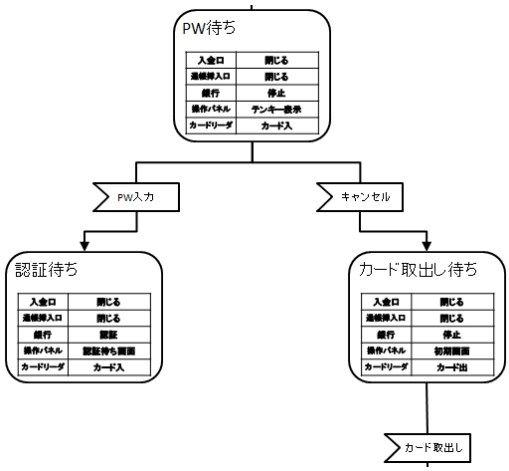

図 3 安定状態
Figure 3 Stable states

4.3 安定状態の定義

安定状態[15]とは，システムの仕様が許す各ドメインの状態の組み合わせからなる．システム開発者は，判明しているシステムのもつ機能を安定状態として定義し、それぞれの安定状態の状態名を定義する．

4.4 安定状態と入力遷移からなる状態マシンの生成

4.2 と 4.3 より，開発したいシステムの状態マシンを生成する．生成の手順は，以下の通りとなる．

(1) ある安定状態を構成する，いずれかのドメインの状態に入力イベントが発生すると，そのドメインの状態のみが次の状態へと遷移した，”中間状態”に遷移する．
(2) 中間状態を構成する各ドメインの状態が，0 個以上の出力遷移のみで到達できる，すべての状態の組み合わせを考える．
(3) (2)のドメインの状態の組み合わせの中から，4.3 で定義された安定状態のドメインの組み合わせと一致する組み合わせを探す．1 個以上の安定状態に遷移が可能で，遷移先のない安定状態が存在してはならない．2 個以上の安定状態に遷移する場合，条件分岐を付与する．

全ての安定状態に上記の手順を行うことで，システムの状態マシンを生成する．

具体例で表すと，以降のような流れになる．図 4，図 5 のようなドメインプロパティを持つ二つのドメイン domain1 と domain2 がある．domain1 の状態が a，domain2 の状態が x である安定状態 SS1 と，domain1 の状態が c，domain2 の状態が z である安定状態 SS2 の二つが定義されている．初期の安定状態は SS1 である．

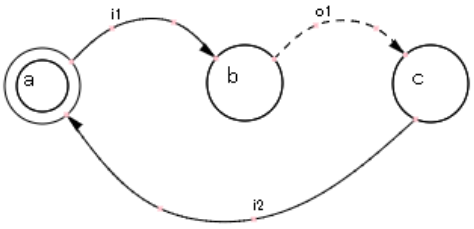

図 4 domain1 のドメインプロパティ
Figure 4 Domain property of domain1

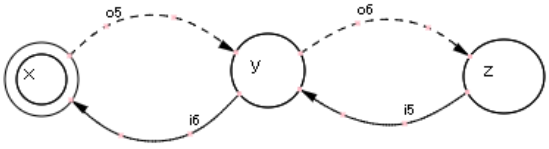

図 5 domain2 のドメインプロパティ
Figure 5 Domain property of domain2

(1) domain1 の状態 a に対し，i1 という入力イベントが発生すると，domain1 の状態 a は状態 b に遷移し，domain2 の状態 x は維持される．よって，中間状態は b と x となる．

(2) domain1 の状態 b が 0 回以上の出力遷移のみで到達できる状態は，b と c の２つである．domain2 の状態 x が 0 回以上の出力遷移のみで到達できる状態は，x と y と z の３つである．２つのドメインによるすべての状態の組み合わせは，domain1 と domain2 の状態が，b と x，b と y，b と z，c と x，c と y，c と z の６つである．

(3) (2)で得られた６つの状態の組み合わせのうち，事前に定義されている安定状態の状態の組み合わせと一致する状態の組み合わせを探す．domain1 の状態が c，domain2 の状態が z の組み合わせは，安定状態 SS2 の状態の組み合わせと一致するので，安定状態 SS1 に入力イベント i1 が発生すると，安定状態 SS2 に遷移可能であると分かる(図 6).

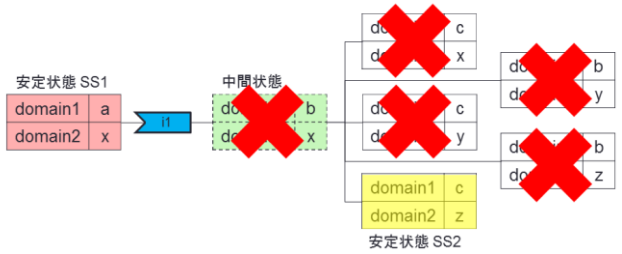

図 6 安定状態の遷移
Figure 6 Transition of stable state

4.5 不要な状態遷移の排除

4.4 で生成される状態マシンには一般に，理論上遷移可能であっても，システムの仕様にはそぐわない不要な遷移が含まれる．開発対象になり得るシステムは多種多様であり，そのすべてに対して，不要な遷移を自動的にかつ確実に排除することは難しく，システム開発者が目視で取捨選択を行うことが好ましい．しかし大規模なシステムになると，生成される全ての理論上可能な遷移の中から目視で取捨選択をすることは，開発者に対し非常に多くの手間や時間を求めることになるため，本方式では，不要な遷移の大部分を自動的に排除するものとした．残った不要な遷移は，システム開発者が目視でチェックし排除する．

不要な遷移の大部分を自動的に排除するための条件は，6 章で述べる．

4.6 出力遷移を加えた最終的なシステムの状態マシンの生成

4.4 と 4.5 によって生成された状態マシンに，4.4(2)で遷移に使用した出力遷移を付加し，システムの最終的な状態マシンを生成する．

5. 支援システム

4 の提案手法を元に，支援システムを開発した． 本支援システムは３つの画面から構成される．

図 7 はプロブレム図の作成・編集，プロジェクト全体としてのセーブ・ロード，プロジェクトの削除を行う画面である．本支援システムにおいて，この画面がメイン画面となる．ユーザが行うことができる操作は，プロブレム図を構成する図形の作成・編集，およびそれらを結ぶ線の作成・編集と，それぞれの名前の記述・変更，作成したプロブレム図の画像の保存，プロジェクトのセーブ・ロード・削除である．インタフェースと要求参照は，本システムが自動的に認識する．

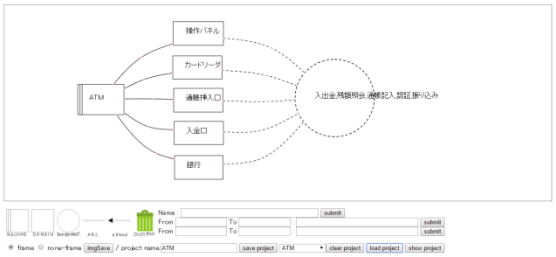

図 7 プロブレム図編集画面(メイン画面)
Figure 7 Edit screen of problem diagram(main screen)

図 8 はドメインプロパティを作成・編集・削除する画面である．図 7 で作成したプロブレム図の中から，編集したいドメイン上をダブルクリックすることで表示される．ユーザが行うことができる操作は，状態の図形の作成・編集，および入出力遷移の作成・編集と，それぞれの名前の選択・変更，作成した状態マシンの画像保存，作成した状態マシンの保存・削除である．ドメインプロパティの作成・編集で使用する状態名と入出力イベント名は，プロブレム図で作成されたインタフェースと要求参照の情報を継承しているため，ユーザが改めて記述する必要がなく，つづりの間違いや未定義の状態名・入出力イベント名の記述を防ぐことができる．

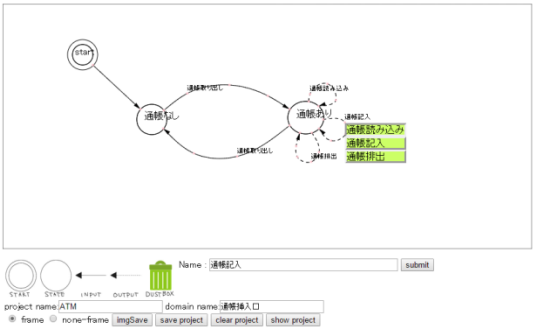

図 8 ドメインプロパティ編集画面
Figure 8 Edit screen of domain property

図 9 は，安定状態の編集と，システムの最終的な状態マシンを自動生成する画面である．図 7 で作成したプロブレム図の要求をダブルクリックすることで表示される．左側の画面にシステムの状態マシンが自動生成され，右側の画面の上部にある表で安定状態を定義する．安定状態表の下のテキストボックスは，生成された状態マシンをテキスト化したものが表示される．ユーザが行うことができる操作は，左画面では状態マシンを自動生成するボタンのクリック，右画面の安定状態表では，各ドメインの状態の選択・変更，作成した状態マシンのテキスト化を行うボタンのクリックである．安定状態を定義するうえで使用する各ドメインの状態名は，プロブレム図で作成された要求参照の情報を継承しているため，ユーザが改めて記述する必要がなく，つづりの間違いや未定義の状態名の記述を防ぐことができる．

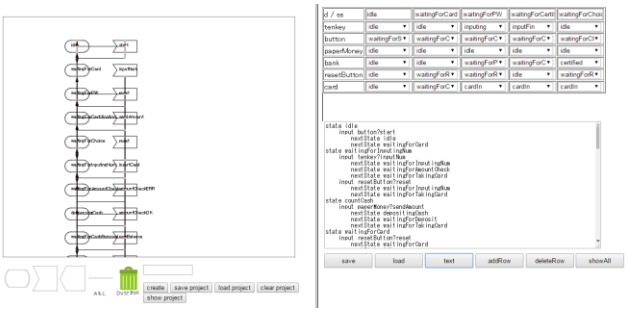

図 9 安定状態の編集およびシステムの状態マシンの自動生成画面
Figure 9 Screen of editting stable states and creating state transition diagram of system

提案手法の 4.5 と 4.6 の機能については，現段階では未実装である．

4.5 については，6 章で記述する不要な遷移の排除条件の適用をモードとして採用し，様々な排除条件を適用した結果の状態マシンをボタン一つで切り替えることができるようにする．また，ユーザが要不要をユーザ自身でも決定できるような機能を付ける．

4.6 については，4.4 と 4.5 より完成した状態マシンに対し，使用した出力遷移を加えた最終的なシステムの状態マシンを，ボタン一つで生成させることができるように実装する．

6. 評価

5 で作成したシステムの有効性を評価するために，実例として，簡易的な銀行の ATM システムを取り上げる．

6.1 問題

ATM では，利用者がお金の預け入れ，引出し，残高照会を行う．利用者はまず，磁気カードをカード挿入口に挿入する．ATM はカードの挿入を確認すると，銀行のネットワークにアクセスし，パスワードを要求する．利用者はテンキーを用いてパスワードを入力し，ATM はそのパスワードが正しいかどうか，銀行に認証確認を行う．認証が出来なければ，利用者にもう一度パスワードの入力を促し，認証が完了すれば，操作の選択を促す．預け入れが選択された場合，ATM は入金口を開き，利用者はその中に金を入れる．ATM は金が入ったことを確認すると，入金口を閉じ，金を数え，残高に追加し，カードを排出する．引出しが選択された場合，利用者はテンキーで引き出したい金額を入力する．ATM は入金口に金額分を用意し，残高を減らし，入金口を開く．利用者が金を取り出したのを確認すると，カード挿入口からカードを排出する．残高照会が選択された場合，銀行のネットワークにアクセスし，残高を表示し，カードを排出する．キャンセルボタンが押された場合，いずれの画面の場合であっても操作を中止し，カードを排出する．また，処理中にエラーが発生した場合も同様に，操作を中止し，カードを排出する．

以上の問題を元に，プロブレム図を作成する(図 10)．ドメインは，カード(Card)，紙幣(PaperMoney)，テンキー(Tenkey)，ボタン(Button)，リセットボタン(ResetButton)，銀行(Bank)の 6 つとした．

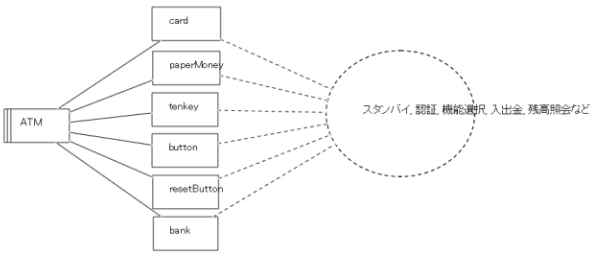

図 10 ATM のプロブレム図
Figure 10 Problem diagram of ATM

6.2 ドメインプロパティの定義

各ドメインについて，ドメインプロパティを定義する．

6.2.1 カードドメイン

カードドメインは，ATM 起動時のカードの挿入の催促と，処理終了時のカードの排出を行う．ドメインプロパティを図 11 に示す．

6.2.2 紙幣ドメイン

紙幣ドメインは，預け入れの場合，入金口を開け入金を促す．入金口の金が入ったら入金口を閉じ，入金された金額を数え，システムに結果を報告する．引出しの場合，出金金額分を入金口に用意し，入金口を開ける．入金口から金がとり出されると，処理を終了し入金口を閉じる．ドメインプロパティを図 12 に示す．

6.2.3 テンキードメイン

テンキードメインは，テンキーが必要となったとき，入

力を促す. 入力が完了すると, 結果をシステムに報告する. ドメインプロパティを図 13 に示す.

6.2.4 ボタンドメイン

ボタンドメインは，操作の選択を行う．可能な操作は，START，GETCASH，BALANCE，DEPOSIT，CONTINUE の５つである．GETCASH，BALANCE，DEPOSIT はそれぞれの処理が完了すると処理終了となる. CONTINUE は再び操作の選択を促す．ドメインプロパティを図 14 に示す.

6.2.5 リセットボタンドメイン

リセットボタンドメインは，リセットが押されたら，どの画面時であっても，マシンに対しアイドル状態に戻るよう要求する．ドメインプロパティを図 15 に示す.

6.2.6 銀行ドメイン

銀行ドメインは, 入力されたパスワードの真偽を判断し, 認証完了かエラーの合図を出す．引出しの場合，入力された金額と残高を比較し，金額が残高を超えていればエラーを送り, 超えていなければ出金し, 残高から金額分を引く. 預け入れの場合, 入金された金額を数え, 残高に追加する. 残高照会の場合，残高を送る．ドメインプロパティを図 16 に示す.

6.3 安定状態の規定

ATM におけるすべての安定状態を表 1 に示す. 安定状態は３０個となった.

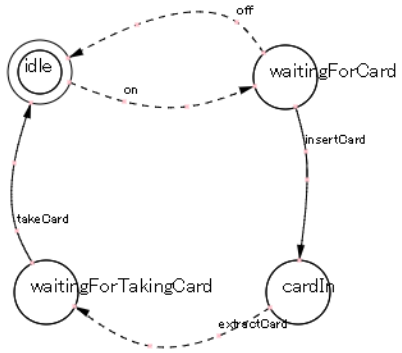

図 11 カードドメインのドメインプロパティ
Figure 11 Domain property of Card domain

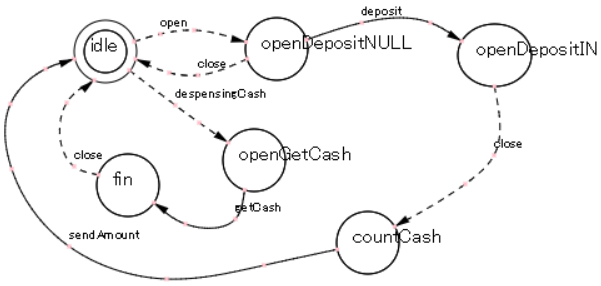

図 122 紙幣ドメインのドメインプロパティ
Figure 122 Domain property of PaperMoney domain

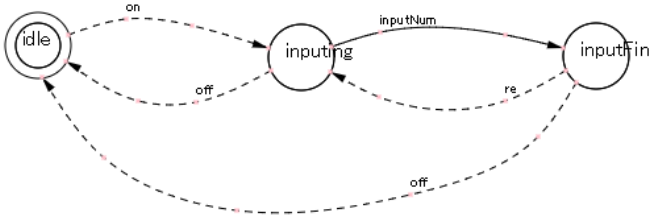

図 133 テンキードメインのドメインプロパティ
Figure 133 Domain property of Tenkey domain

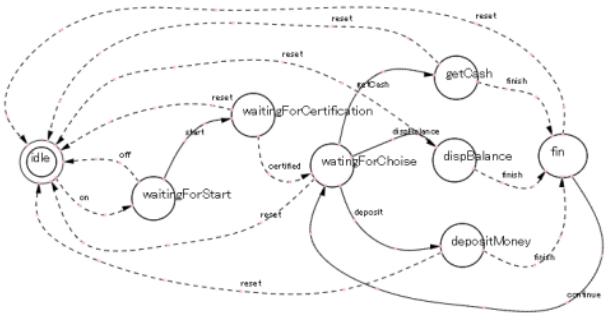

図 14 ボタンドメインのドメインプロパティ
Figure 14 Domain property of Button domain

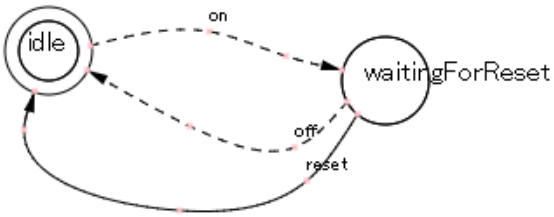

図 15 リセットボタンドメインのドメインプロパティ
Figure 15 Domain property of ResetButton domain

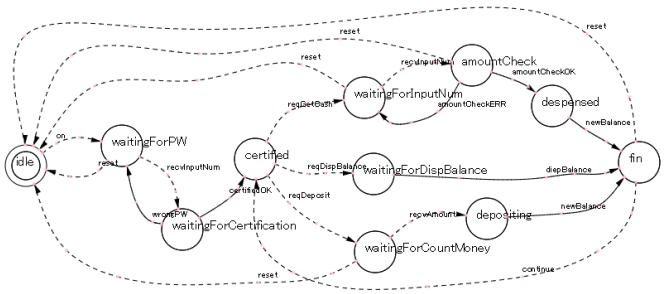

図 16 銀行ドメインのドメインプロパティ
Figure 16 Domain property of Bank domain

表 1 ATM の安定状態

Table 1 Stable states of ATM

Domain / Stable State	idle	waitingForCard	waitingForPW	waitingForCertification	waitingForChoice
Card	idle	waitingForCard	cardIn	cardIn	cardIn
PaperMoney	idle	idle	idle	idle	idle
Tenkey	idle	idle	inputing	inputFin	idle
Button	waitingForStart	waitingForCertification	waitingForCertification	waitingForCertification	waitingForChoice
ResetButton	idle	waitingForReset	waitingForReset	idle	waitingForReset
Bank	idle	idle	waitingForPW	waitingForCertification	certified

Domain / Stable State	waitingForInputingNum	waitingForAmountCheck	despendingCash	saitingForCashRemoval	waitingForDeposit
Card	cardIn	cardIn	cardIn	cardIn	cardIn
PaperMoney	Idle	idle	idle	openGetCash	openDepositNULL
Tenkey	inputing	inputFin	idle	idle	idle
Button	getCash	getCash	getCash	getCash	depositMoney
ResetButton	waitingForReset	idle	idle	idle	waitingForReset
Bank	waitingForInputNum	amountCheck	despensed	fin	waitingForCountMoney

Domain / Stable State	countCash	depositingCash	waitingForBalance	processFin	watingForTakingCard
Card	cardIn	cardIn	cardIn	cardIn	waitingForTakingCard
PaperMoney	countCash	idle	idle	idle	idle
Tenkey	idle	idle	idle	idle	idle
Button	depositMoney	depositMoney	dispBalance	fin	idle
ResetButton	idle	idle	idle	waitingForReset	idle
Bank	waitingForCountMoney	Depositing	waitingForDispBalance	fin	idle

図 17 状態マシンのテキスト化

Figure 17 Text-izing of state machine

6.4 結果・検討

6.4.1 結果

本システムによって生成された結果の一部を抜粋したものを図 17 に示す.

生成された遷移は６１個あった．このうち自己遷移(図 17 黄枠)は１５個，不要な遷移(図 17 赤枠)が６個，システムの仕様上不要であるが存在しても問題のない遷移(以下あり得る遷移と記述する，図 17 青枠)が１６個であった．生成された遷移には，以下の３つの特徴があった.

① すべての自己遷移は不要な遷移であった.
② 不要な遷移とあり得る遷移に分類された遷移の約半分が，waitingForTakingCard への遷移であった.
③ ほとんどの安定状態において，正しい遷移は１つのみであった.

正しい遷移が０個の安定状態は，遷移先のない安定状態を許さないという条件(4.4(3)参照)より，エラーメッセージを表示する．ドメインプロパティもしくは安定状態の定義に誤りがある可能性があるため，ドメインプロパティと安定状態の再検討行う必要がある.

6.4.2 検討

6.4.1 より，不要な遷移の排除の条件として，以下の４つを検討した.

① 自己遷移は全て不要な遷移とする．必要な自己遷移は，開発者が手動で訂正する．
② 安定状態に属性を付与する．ATM の例において，waitingForTakingCard はエラーが起こった場合やリセットボタンが押されたときにのみ遷移が許される安定状態であるため，waitingForTakingCard に対し，エラーが発生したときもしくはリセットボタンが押されたときにのみ遷移するといった属性を加えることにより，それ以外の遷移を作成しない．
③ ①②によって遷移の排除を行い，なお複数個の安定状態への遷移が可能である場合，遷移先の安定状態が１つになるように遷移の削除を促す．２つ以上の遷移が正しい場合は，条件分岐を付加することを促す．
④ 4.4 の(2)において，中間状態の各状態が到達可能な状態を検索するときに使用する，出力遷移の回数に制限を付ける，もしくは条件を付ける．

6.4.3 検討手法の評価実験

ここでは，検討手法のうち④に関して評価実験を行った．出力遷移に対する制限として，以下の２つを検討した．

(1) 使用する出力遷移は１つまでとする
(2) 遷移先の状態が idle 状態である場合，idle 状態以降の遷移を認めない

(1)の条件を付加し状態マシンを再生成すると，6.4.1 の結果の中から１６個の遷移を排除された．排除された遷移は，正しい遷移が１個，あり得る遷移が８個，不要な遷移が２個，自己遷移が４個であった．正しい遷移も排除されたが，多くの誤りの遷移が排除され，(1)の制限は効果的であると考えられる．

(2)の条件を付加し状態マシンを再生成すると，6.4.1 の結果の中から５個の遷移を排除できた．排除された遷移は，正しい遷移が１個，あり得る遷移が０個，不要な遷移が１個，自己遷移が３個であった．(1)と比べると，排除される個数は大きく減った．(1)が過剰排除であったと判断された場合，(2)の制限は効果的であると考えられる．

7. おわりに

組み込みシステムの仕様設計を支援するために，プロブレムフレームの考え方に基づいた，システムの状態マシンの自動生成手法を提案した．また，提案手法を元に，システムの仕様を自動生成するシステムを作成した．作成したシステムに，実例として単純化した銀行の ATM を適用したところ，起こりうるすべての遷移からなる状態マシンを自動的に生成することができた．しかし，起こりうるすべての遷移には不要な遷移が含まれ，ATM の例より，不要な遷移の多くをシステムが自動的に排除する条件を検討し，検討手法の一部に対し評価実験を行った．

検討手法の評価実験により，多数の不要な遷移を排除することができたが，正しい遷移が排除されたり，遷移先の安定状態がすべて排除されるなどのエラーメッセージが表示された．現時点ではドメインプロパティか安定状態の誤りとみなしているが，適用した条件が適切であるかどうかの確かな検証が今後の課題と考える．さらに，検討手法の①②③についての有効性を検証，また，銀行の ATM だけではなく，様々な実例を用いてさらに分析・検討・評価を行い，本システムの有用性をより詳細に評価する所存である．

参考文献

1) Jackson, M.: Problem Frames: Analyzing & Structuring Software Development Problems, Addison-Wesley (2001).
2) 紫合治, 横山薫: プロブレムフレームに基づく組み込みシステムの状態遷移分析支援システム, 情報処理学会論文誌, Vol.53, No.2, pp523-534 (2006).
3) 和田遥，紫合治: 組み込みシステムにおける仕様の自動生成, ソフトウェアエンジニアリングシンポジウム(2011).
4) Cox, K., Hall, Jon, G. and Rapaotti, L.: Editorial; A roadmap of Problem Frames research, Information and Software Technology, Vol.47, Issue 14, pp.891-902 (2005).
5) Ourusoff, N.: Towards a CASE tool for Jackson's JSP, JSD and Problem Frames, Proc. 1st International Workshop on Applications and Advances of Problem Frames, pp.69-73 (2004).
6) Tun, T.T., Yu, Y., Haley, C. and Nuseibeh, B.: Modelbased Argument Analysis for Evolving Security Requirements, 4th International Conference on Secure Software Integration and Reliability Improvement, pp.88-97 (2010).
7) Colombo, P., et.al: Towards a Meta-model for Problem Frames: Conceptual Issue and Tool Building Support, Fourth International Conference on Software Engineering Advance, pp.339-345 (2009).
8) Hatebur, D., Heisel, M., and Schmidt, H.: A Formal Metamodel for Problem Frames, MoDELS 2008, LNCS 5301, pp.69-82 (2008).
9) Lavazza, L. and Bianco, V.D.: A UML-based approach for representing problem frames, Proc. 1st International Workshop on Applications and Advance of Problem Frames, pp.39-48 (2004).
10) Choppy, C. and Reggio, G.: A UML-based approach for problem frame oriented software development, Information and Software Technology, Vol.47, Issue 14, pp.929-954 (2005).
11) Seater, R. and Jackson, D.; Problem Frame Transformations: Deriving Specifications from Requirements, Proc. 2nd International Workshop on Applications and Advances of Problem Frames, pp.71-80 (2006).
12) Rapanotti, L., Hall, L.G. and Li, Z: Deriving Specifications from Requirements through Problem Reduction, IEE Proc. Software, Vol.153, No.5, pp.183-198 (2006).
13) Li, Z.: Progressing Problems from Requirements to Specifications in Problem Frames, Proc, 3rd International Workshop on Applications and Advances of Problem Frams, pp.53-50 (2008).
14) Jackson, D.: Alloy: A Lightweight Ocject Modeling notation, ACM Trans. Software Engineering and Methodology, Voll.11, No.2, pp.256-290 (2002).
15) 紫合治: 安定状態と優先イベント規定によるコントローラ生成, 情報処理学会論文誌, Vol.56 (2015)