情報処理学会論文誌 Vol.53 No.2 523–534 (Feb. 2012)

プロブレムフレームに基づく組込みシステムの状態遷移分析支援システム

紫合 治[1,a)] 横山 薫[2]

受付日 2011年6月14日, 採録日 2011年11月7日

概要：本論文では，ソフトウェア開発の初期段階において有効であるプロブレムフレームに基づく要求分析支援システムについて述べる．システムは要求分析の工程に沿って，プロブレム図，ドメイン状態遷移図，要求状態遷移図の作成を支援し，それらの間の妥当性をチェックした後，マシンの仕様となる状態遷移記述を自動生成する．このとき，プロブレム図で定義したイベントや状態をそのままドメイン状態遷移図で活用し，ドメイン状態遷移図で定義した状態をそのまま要求状態遷移図で利用するなどにより，誤りのない要求分析を支援する．さらに，これらの状態遷移図間の整合性を調べて，要求の漏れや誤り等を自動的にチェックする機能を持つ．

キーワード：プロブレムフレーム，状態遷移図，要求分析，仕様生成

State Transition Analysis System for Embedded Systems by Problem Frames

Osamu Shigo[1,a)] Kaoru Yokoyama[2]

Received: June 14, 2011, Accepted: November 7, 2011

Abstract: This paper describes a total support system for state transition diagrams in the problem frames. The system provides graphical editors for problem diagrams, domain state transition diagrams as domain properties and requirement state transition diagram, all of which are related together. By analyzing these diagrams, the system automatically generates the machine state transition description.In the system, event names defined in the problem diagram are used in the related domain state transition diagrams and domain state names defined in the domain state transition diagrams are used in the requirement state transition diagram to describe the required domain behavior. This significantly reduces the effort to draw new diagrams. Also, by checking the consistency among these diagrams, the system reports the requirement errors, including the leakage of requirements.

Keywords: problem frames, state transition diagram, requirement, analysis, specification generation

1. はじめに

ソフトウェア開発の分析・設計では，UML [1] やその支援ツール [2], [3] が広く使われている．上流工程である要求分析においても，ユースケース図，シーケンス図，アクティビティ図，状態遷移図，クラス図などが活用されている．しかしながら，UML は元来ソフトウェアの構成や振舞いを設計するためのものであり，いわば問題の解決策を設計するためのツールといえる．

問題の解決を考える前に問題そのものに集中し，問題が存在する現実世界を分析・構造化し，さらにパターン化していく手法として，Jackson のプロブレムフレーム [4] の考え方が注目されている．システムの開発の初期段階においては，システムそのものを考える前にシステムを取り巻く現実世界を明確に把握することは重要であり，プロブレム

[1] 東京電機大学情報環境学部
School of Information Environment, Tokyo Denki University, Chiba 270–1382, Japan
[2] NEC ソフト株式会社
NEC Soft, Ltd.
[a)] shigo@sie.dendai.ac.jp

フレームの活用の効果が期待できる．

UML を活用する場合，図形描画だけでなく，図のエラーのチェックや関連図との関連のチェックなども含めて，ツールの活用が有効である．プロブレムフレームに対しても，プロブレム図の描画だけでなく，そこに現れるドメインのプロパティ定義や要求記述のための状態遷移図等，各種の図形の描画，図形間の関連のチェックや関連情報の生成等の機能を持つツールによって，分析作業の効率化が期待できる．

プロブレムフレームに沿った分析を支援するツールとして，UML のように広く利用されているものはまだない．研究者向ツールとして，イベント計算（述語論理式を使った手法）を使った OpenPF [5] が公開されているが，現場の技術者にとってはイベント計算の記述は簡単ではないと思われる．実務者向けのツールとしては，UML のクラス図，ユースケース，状態マシン図を使ってプロブレムフレームを適用する手法 [6], [7] があるが，この場合，もとのプロブレムフレームの主要な概念である問題（環境のドメイン）と解（マシン）の分離や，仕様現象と要求現象の区別などが不明確になってしまう．また，現象をメソッドやフィールド変数で表すため，問題分析の段階で実装の概念が出てきてしまう．これらを解決するためには，プロブレムフレームの概念を直接的に表現し，かつ実務者にも使いやすいツールが望まれる．

本論文では，プロブレムフレームの中で組込みシステムによく出てくる「振舞いフレーム」に限定した支援システムについて述べる．システムはプロブレム図の作画をもとに，共有現象に基づくドメイン記述，要求記述をそれぞれ特有の状態マシン図で規定し，ドメイン記述と要求記述の矛盾の自動チェック機能や，マシンの仕様の自動生成機能を持つ．ここでは，適用フレームを制限し，かつプロブレム図に一部制約（マシンと直接関連しないドメインは省略する等）を加えることにより，より高度な支援機能を持たせるようにしている．プロブレムフレームによる組込みシステムの状態遷移設計手法として，著者らは JSP（ジャクソン法）による状態遷移設計手法 [8] を提案したが，その手法を効率良く適用するには支援ツールが必須であり，ここで述べるシステムは，この手法のガイドと自動化ツールとして使うこともできる．

以下に，2 章で基本となるプロブレムフレームと状態マシンの概念について説明し，3 章で支援システムの機能を，4 章でシステムの適用評価について述べる．5 章で関連研究について議論し，6 章でまとめと今後の課題について述べる．

2. プロブレムフレームと状態マシン

プロブレムフレームは，システムの開発において，開発すべきシステムについて考察するまえに，システムによって制御される部品や装置等の外部環境について十分に調べ，それらの外部環境がどのように振る舞っていくべきかを規定するという考え方である．これらの外部環境やその振舞い方は，元の問題そのものに属し，問題の解である開発すべきシステムの外にあると考える．つまり，解を考える前に問題を十分分析するという方法である．

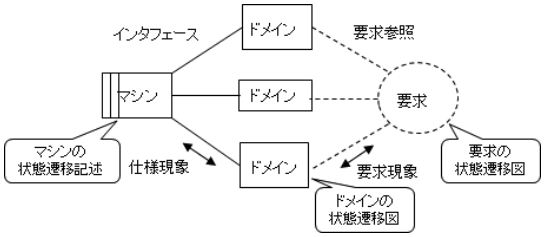

図 1 プロブレム図
Fig. 1 The problem diagram.

Jackson はさまざまな問題を 5 つの型（Problem Frames）に分類し，それぞれについて考察している [4]．ここでは，このうち，組込みシステムによく現れる 2 つの振舞い関連の問題（必要な振舞いと命令された振舞い）に限定して考察する．

2.1 プロブレム図

プロブレムフレームで基本となるものがプロブレム図である．**図 1** にプロブレム図の構成を示す．図で，2 本の縦線を持つ要素を**マシン**と呼び，問題の解決策を担当する．マシンと関連する四角の要素を**ドメイン**と呼び，外部環境となる部品や装置を表す．また，破線の楕円で，これらのドメインに対する**要求**を表す．基本的なプロブレム図では，マシンと要求がそれぞれ 1 つ存在し，ドメインが複数個存在する．ここでは簡単のため，基本的なプロブレム図（マシンと要求はそれぞれ 1 つ）に限るものとする．

ドメインには，マシンと直接関係するもの（モータや電磁弁等）と間接にしか関係しないもの（水槽の水，信号制御システムでの車，運転手等）がある．間接的なドメインは，**問題世界の深い**ところにあり，マシンに近くなるほど**浅い**ドメインといわれる [4]．ここでは，簡単なため，間接的なドメインは記述しないという制限を加える．この制限により，要求はマシンが直接制御するドメインに対して記述する必要がある．この制限についての考察は，5.1 節の関連研究との比較で述べる．

振舞いフレームのなかで，命令された振舞いでは，オペレータ（人間）を表す**従順なドメイン**[4] が出てくるが，ここでは，その代わりに操作ボタンや操作パネル等をドメインとし，人間をドメインとは見ないことにする．操作ボタンのドメインを従順なドメインととらえることができる．実際には，操作ボタンを操作するオペレータドメインが間接ドメインとなるが，ここでは間接ドメインは省略する．

マシンとドメインをつなぐ実線を**インタフェース**と呼び，

表 1 仕様現象と要求現象
Table 1 Example of the specification phenomena and the requirement phenomena.

問題	ドメインの例	仕様現象	要求現象
片側交互通行用信号	信号機	赤と緑のランプへのパルス	進め状態，止れ状態
走行距離計表示	走行車	車の回転毎に出るパルス	スピードと走行距離
周期と範囲の入力	周期＆範囲	編集機能	データ値
炉の操作	炉設備	炉の操作（点火，送風等）	燃焼中，停止等の状態
水門制御	門扉＆モータ	逆（順）回転，On/Off，位置	閉鎖状態，開門状態等

マシンがドメインと関連していることを表す．また，要求とドメインをつなぐ破線を**要求参照**と呼び，要求がドメインに対して要請することを表す．ここでは，要求とマシンは直接的な関連を持たないという制限を設ける．つまり，要求は解決策に対しては何も要請しないものとする．さらに，簡単のため，ドメイン間の関連は付けられないものとする．なお，間接ドメインを持たないことやドメイン間の関係は持たないという制限は，支援システムの実現での簡単化のためであり，将来的にはシステムの機能強化として，この制限を取り除くことも検討している（6 章参照）．

インタフェースや要求参照にそって，それらの線でつながれた要素間で共有する**現象**が規定される．インタフェースには，マシンが直接感知したり発生させたりできる**仕様現象**を，また要求参照には，顧客が関心を寄せる**要求現象**を規定する．仕様現象は，マシンが決めてドメインに渡す**出力現象**（マシンにとっての出力）と，ドメインが決めてマシンに渡す**入力現象**（マシンにとっての入力）がある．出力は，装置への指令，入力はマシンへの指令やセンサの結果報告等からなる．一方，要求現象は，装置がどうなっているかの状態を示すことが多い．**表 1** に，Jackson の本 [4] に現れるいくつかのドメインの例について，仕様現象と要求現象を書き出す．この例では，すべて仕様現象はイベント，要求現象は状態となっている（正確には，要求が強制する現象が状態となる）．ここでは，仕様現象はイベント，要求現象は状態に限定するものとする．

2.2 状態マシン

振舞いフレームの場合，ドメインは，仕様現象（イベント）と要求現象（状態）の関連を状態マシンとして規定する．また，マシンの仕様として，関連するドメインのイベント制御を状態マシンで規定する．一方，要求は必ずしも状態マシンで記述するとは限らないが，Jackson の片側交互通行用信号や水門制御などの例 [4] でも見られるとおり，ドメインの状態変化の要求を状態マシンで表すことも多い．また，Bray [9] はフレームごとに適切な手法を述べた表を示しているが，そこでは制御フレーム（振舞いフレームのこと）には有限状態マシン（FSM）を推奨している．著者らは，ドメインの状態マシンの対応関係の規定からマシンの仕様となる状態マシンを生成する手法を提案したが [8]，そこでの対応関係の規定は要求を状態マシンで表した形式になっている．ここでは要求も状態マシンで記述する．ただし，不完全な要求への対応として，断片的な状態マシンも扱えるようにした．この場合，ドメインの状態マシンの規定に沿って，支援システムが遷移の漏れ等につきエラーを表示するので，それらを修正していくことで完全な要求の状態マシンを得ることができる（4.3 節参照）．

ドメインの状態マシンは，そのドメインがどうなっているかを規定したもので，「所与」の記述であり，問題の分析において，考え出すものではなく，事実として与えられるものである．これは，設計対象マシンに対するポートのプロトコルを規定した Interface Automata [10] と同様の記述とする．

要求の状態マシンは，ドメインがどのようになるべきか，どのように振る舞うべきかを，ドメインの状態の組合せによって規定したもので，「願望」の記述となる．これは，状態の中に世界の状況規定をドメインの状態の組として含む Kripke 構造 [11] や，状態指向の状態マシン [12] に対応すると考えることができる．

マシンの状態マシンは，ドメインからの入力イベントに反応して，どのように出力イベントを出していくかを規定したもので，UML の振舞い状態マシンに対応する．我々の支援システムでは，マシンの状態マシン（仕様）はドメインと要求の記述から自動生成する．

これらを使って，支援システムによる問題分析の流れは以下のようになる．

① プロブレム図を作成し，共有現象（イベントと状態）を規定．
② ドメインごとに，①のイベントと状態の関係をドメイン状態マシンで規定．
③ ドメイン達の状態の振舞いの条件を要求の状態マシンで規定．
④ ①〜③から，マシンの状態マシン（仕様）を自動生成．

3. 分析支援システム

プロブレム図と各種状態マシンを作成し分析するためのシステムを開発した．なお，ここで説明するドメインの状

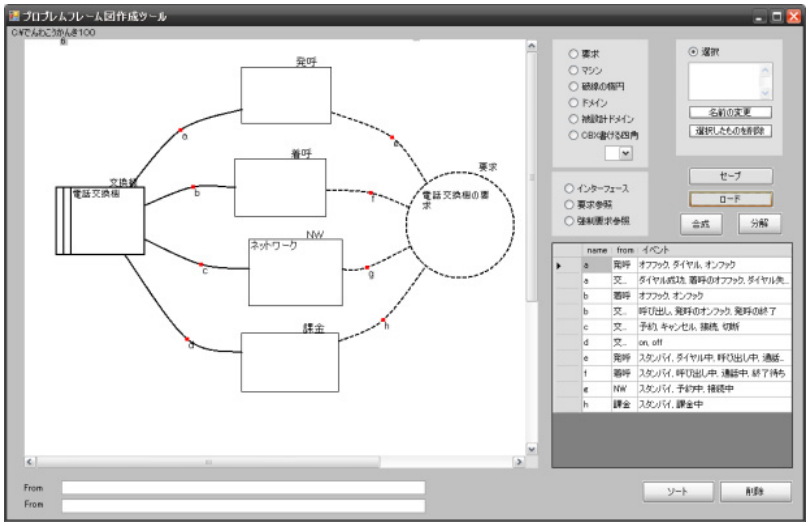

図 2 プロブレム図の作成画面
Fig. 2 The problem diagram editor.

態遷移図，要求の状態遷移図，マシンの状態遷移記述についての形式的な規定については，文献 [8] を参照されたい．

3.1 支援システムの狙い

プロブレムフレームによる問題分析では，通常 4 種類の記述（プロブレム図，ドメイン仕様，要求記述，マシンの仕様）を書く必要があるが，プロブレム図によって問題全体の枠組みを示し，その共有現象の記述によって，他の問題記述で用いる共通語彙を規定することになる．手書き（たとえばパワーポイントによる）の場合は，プロブレム図を見ながら，ドメインの状態遷移図や要求の記述を書く必要がある．いろいろな状態遷移図や仕様の記述で同じ語彙（イベント名や状態名）を何度も書くことになり，手間がかかるだけでなくエラーの発生を促すことにもなる．要求の状態遷移記述では，ドメインの状態遷移図を見ながら，ある状態でどんなイベントが発生するか調べて，遷移規則を決めていく必要がある．さらに，要求やドメインの状態遷移をたどりながら，マシンの仕様を書いていく場合，要求と何枚かのドメインの状態図を同時に見ながら作業を進める必要があり，ドメインの数が多くなると人手による作業はますます困難になってくる．

支援システムは，これらの問題を解消することを狙いとしたものである．最初にプロブレム図を書くと，仕様現象や要求現象の記述をガイドする共有現象テーブルが現れ，そこにイベントや状態名を記述することによって，以降の分析に必要な語彙をほとんど定義することができる．ドメインや要求の記述では，これらの語彙を適切に選択しながら図を書いていくので，作成が楽になり，記述誤りを防ぐ効果がある．また，ドメイン状態図と要求状態図の間の無矛盾性チェックによって，要求のエラーを見つけるだけでなく，不完全な要求から始めて，チェック結果のエラーを修正していく形で，要求を完成させていくこともできる．最後に，ドメインと要求の記述から，その要求を満たすマシンの仕様（厳密には仕様の例）を自動生成する機能を持つ．これによって，4 種類の記述の中でも手間のかかるマシンの仕様記述を行う作業が省け，かつ誤りのない仕様を得ることができる．

3.2 プロブレム図作成ツール

図 2 はプロブレム図作成ツールによる電話交換システムのプロブレム図の作成例を示す．図の画面中，右上部分に要素や接続線の種類を示すラジオボタンが，右下部分に，インタフェースや要求参照の共有現象を示す，共有現象（イベント・状態）管理表がある．

プロブレム図の入力は，最初にノードとなるマシン，ドメイン，要求を作描画し，次に，2 つのノードを指定してアークとなるインタフェースや要求参照を描画する．ツールは基本的なプロブレム図のルールに外れた図の描画に対してはエラーを出すようにしている．たとえば，1 つのプロブレム図にはマシンと要求はそれぞれ 1 つしか作成できないよう，2 つ目の描画ではエラーメッセージを出すようにしている．また，インタフェース接続線はマシンとドメイン間，要求参照接続線は要求とドメイン間のみ引けるようにしている．

インタフェースに対しては，入出力の共有現象（仕様現象）として，イベントを規定する．インタフェースを 1 つ引くと，共有現象管理表にそのインタフェースに関連する入力イベントと出力イベントを記述するための空の行が 2 行生成される．ここで，管理表の from がドメイン名の行にはマシンの入力イベントを記述し，from がマシン名の行にはマシンの出力イベントを記述する．要求とドメイン間の要求参照の場合は，共有現象管理表の from がドメイン

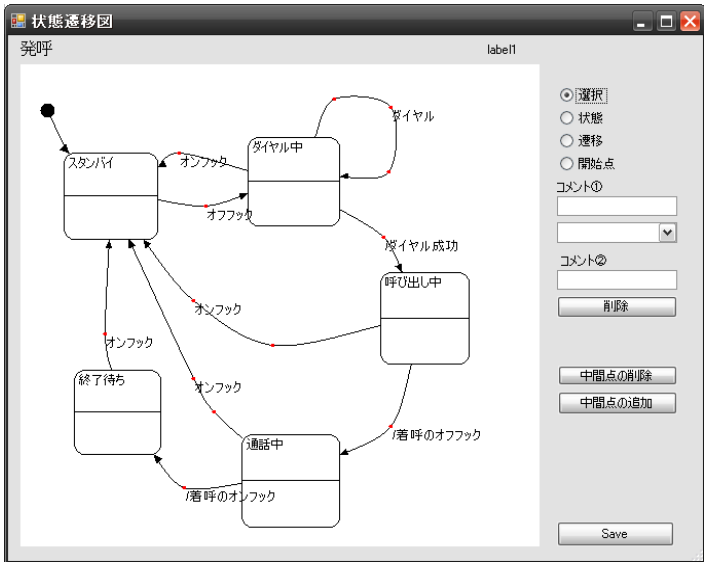

図 4 発呼ドメインの状態遷移図
Fig. 4 The caller domain state diagram.

	name	from	イベント
▶	a	発呼	オフフック, ダイヤル, オンフック
	a	交...	ダイヤル成功, 着呼のオフフック, ダイヤル失...
	b	着呼	オフフック, オンフック
	b	交...	呼び出し, 発呼のオンフック, 発呼の終了
	c	交...	予約, キャンセル, 接続, 切断
	d	交...	on, off
	e	発呼	スタンバイ, ダイヤル中, 呼び出し中, 通話...
	f	着呼	スタンバイ, 呼び出し中, 通話中, 終了待ち
	g	NW	スタンバイ, 予約中, 接続中
	h	課金	スタンバイ, 課金中

図 3 共有現象（イベント・状態）管理表（交... は交換機を示す）
Fig. 3 The shared phenomena table.

名の行にはそのドメインの状態名を記述し，from が要求名の行には何も記述しない．図 3 に図 2 のプロブレム図に対する共有現象管理表の例を示す．図で，共有現象が空白の行は削除されている．

3.3 ドメインの状態遷移図作成ツール

プロブレム図におけるドメインは，それぞれいくつかの状態を持つ．状態の情報は共有現象管理表でそのドメインと要求間に引かれる要求参照に対する項目として記述されている．ドメインの状態遷移図とは，各ドメインがどのようなイベントによってその状態がどのように変化するかを表した図である．ドメインの状態遷移図は，プロブレム図と互換性を持ちながら作成することができる．図 4 はドメインの状態遷移図作成ツールを用いて作成した，電話交換機問題における発呼ドメインの状態遷移図である．ドメインの状態遷移図での遷移は，1 つの入力または出力イベントをともなう．

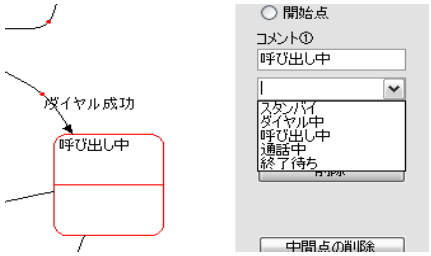

図 5 ドメインの状態名の選択
Fig. 5 Domain state selection.

ドメインの状態遷移図作成ツール画面を開くには，プロブレム図において作成したいドメインをダブルクリックすればよい．黒丸が開始点，角の丸い長方形は状態を表し，矢印と入出力イベントの記述によって遷移を表す．入力イベントと出力イベントは “／” を用いた表記による区別を行い，出力イベントの前に “／” を記述する．ドメインの状態遷移図はプロブレム図の共有現象管理表と互換性を持ち，状態を登録するときにはそのドメインの状態名のリストが表示されるので，そこから選択することによって状態名を指定できる（図 5）．同様に，遷移の登録では，そのドメインの入出力イベントのリストが表示され，そこから選択することでイベントを指定できる．指定したい名前が選択リストになかった場合は，状態名やイベントをその場で記述することができ，この場合は逆にここで記述した情報がプロブレム図の共有現象管理表に反映される．

図 4 に示す発呼ドメインの振舞いは次のとおりである．スタンバイ状態時にオフフックを行うことによりダイヤル中へと遷移し，ダイヤル中状態においてダイヤルを行うと同じ状態に遷移し，スタンバイ以外の状態においてオンフックを行うとスタンバイ状態へと遷移する．これらは発呼ドメイン自身が自働的に行う入力イベントによる遷移で

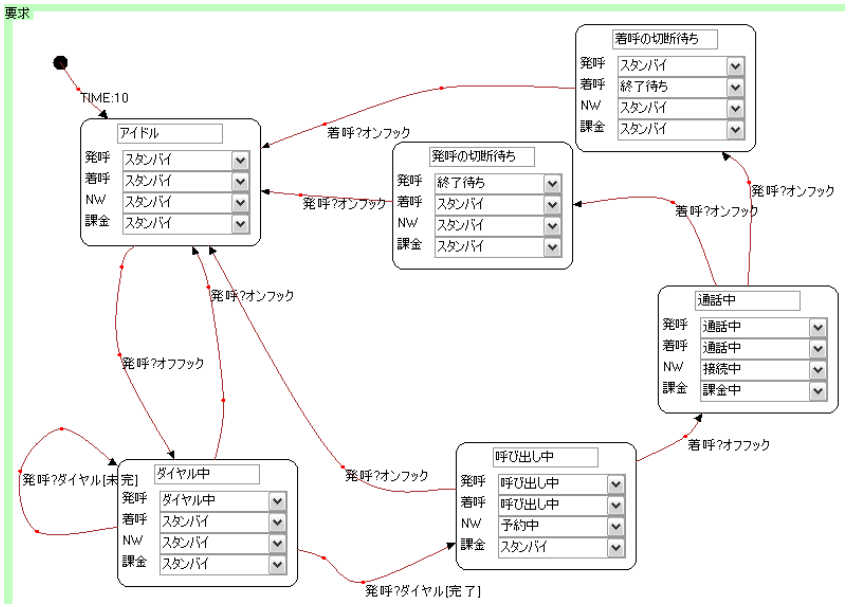

図 6 電話交換機の要求の状態遷移図
Fig. 6 The requirement state diagram for the switching system.

ある．発呼ドメインの他動的なイベントである出力イベントによる遷移としては，ダイヤル中状態からダイヤル成功イベントによる呼び出し中への遷移，呼び出し中から着呼のオフフックイベントによる通話中への遷移，また通話中から着呼のオンフックイベントによる終了待ちへの遷移がある．1つの状態から，入力イベントと出力イベントによる遷移がある場合は，マシンは入力イベントを受ける前に出力イベントを出すことができる．

3.4 要求の状態遷移図作成ツール

プロブレム図を作成し，そのすべてのドメインに対してドメインの状態遷移図を作成することにより，要求の状態遷移図の作成を開始することができる．要求の状態遷移図では，タイムアウトか，ドメインからの入力イベントに対して，すべてのドメインの状態がどのように変化していくかを規定する．このため要求の状態はすべてのドメインの状態を管理する．**図 6** に電話交換機問題における要求の状態遷移図の描画例を示す．

要求の状態遷移図作成ツール画面を開くには，プロブレム図において要求をダブルクリックする．黒丸が開始点，角の丸い長方形は状態を表し，1つの状態ですべてのドメインの状態の組合せを管理する．矢印による遷移は，ドメインの状態遷移図とは異なり，入力イベントによってのみ遷移し，さらに条件記述を付加することができる．条件は入力イベントの記述に[ ]を追加して表記する．入力イベントはどのドメインから発生したイベントかを明確にするため，“?”を用いて“ドメイン名？イベント名”のように表記する．

要求の状態遷移図はすべてのドメインの状態遷移図と関連性を持っている．要求の状態を描画すると，自動的にすべてのドメインに対して状態名のリストを保持するコンボボックスが生成される．要求の状態は，状態名の入力と各ドメインの状態の選択によって規定できる（**図 7**）．また，遷移のイベントに対しては，すべてのドメインの入力イベントのリストを持つコンボボックスが表示される（**図 8**）ので，そこから選択することによって遷移のイベントを規定できる．このように，要求の状態遷移図作成ツールはプロブレム図とドメインの状態遷移図と関連性を持つことから，状態名や入力イベントを選択することで簡単に記述ができるといった入力補助機能を備えている．

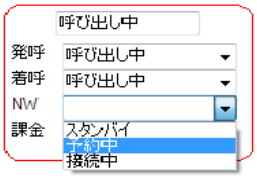

図 7 要求でのドメイン状態の選択
Fig. 7 Domain state selection in a requirement state.

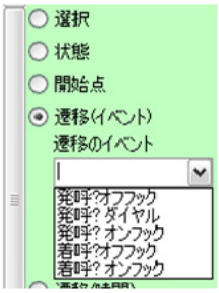

図 8 要求状態遷移図でのイベントの選択
Fig. 8 Input event selection for the requirement transition.

3.5 要求の正しさのチェック機能

要求の状態遷移図はドメインの状態遷移図に沿って作成される．このとき，要求の状態遷移図が正しくドメインの状態遷移図の規定に合っていることがチェックされ，そうでない場合はエラーが表示される．これは，以下のエラーチェックを行う．

(1) ドメインの規定により起こりえないことを要求が想定するエラー

(1–1) 入力イベントが起こりえないエラー

要求の状態 S からイベント $D?E$（ドメイン D から E を入力）による遷移がある場合，ドメイン D の状態遷移図においては，S でのドメイン D の状態 S_d から入力 E による遷移が規定されていなければならない．

(1–2) 出力イベントが起こりえないエラー

要求の状態 S_x からイベント $D_k?E$ で状態 S_y への遷移がある場合，S_x でのドメイン D_i の状態を S_{xi}，S_y でのドメイン D_i の状態を S_{yi} とすると，D_k のドメインでは，S_{xk} から E による入力遷移とそれに続く 0 回以上の出力遷移によって S_{yk} に遷移し，それ以外のドメイン D_j では，S_{xj} から 0 回以上の出力遷移によって S_{yj} に遷移できなければならない．

(2) ドメインの規定により起こりうることを要求が想定していないエラー

要求の状態 S から起こりうるすべての入力イベント（S のドメイン状態とドメインの状態遷移図から得られる）は，S からの遷移に現れていなければならない．

これらのチェックによるエラーが発生した場合，要求の状態遷移図を修正するか，またはドメインの状態遷移図を修正する必要がある．

3.6 マシンの状態遷移記述の自動生成

ドメインの状態遷移図と要求の状態遷移図を作成し，それらの間の正しさをチェックしてエラーがないことを確かめた後で，マシンの仕様を状態遷移記述として自動生成することができる．これは，マシンがドメインからの入力に対してどのように出力を出していくかを，状態遷移の形式で規定したもので，ここでは SDL [13] 記法を参考にして生成する．SDL は通信システムの仕様記述を目的にしたもので，テキスト表記とグラフィック表記があるが，本システムでは簡単化のためテキスト表記を採用した．**図 9** はマシンの状態遷移記述の全体構成である．システム記述，ドメイン記述，シグナル記述はそれぞれ存在するシステム全体のファイル名，存在するすべてのドメイン名，すべてのイベント名を生成する．主な記述はプロセス記述であり，状態遷移記述によってマシンの仕様を表す．

マシンの状態遷移記述を生成するには，プロブレム図においてマシンをダブルクリックすればよい．電話交換機の

```
マシンの状態遷移記述::=
        システム記述
        ドメイン記述
        シグナル記述
        プロセス記述

システム記述::= system ファイル名
ドメイン記述::= domain 略ID：ドメイン名　[略ID：ドメイン名]...
シグナル記述::= signal 略ID 入力/出力　信号
プロセス記述::=process マシン名[状態記述]...

状態記述::=
        state 状態名;
        {入力記述}...
入力記述::=
        input ドメイン名?イベント名;
        {判定記述|出力記述}
判定記述::=
        decision;
        {(条件):出力記述}...
        enddecision
出力記述::=
        [output ドメイン名!イベント名;]...
        nextstate 状態名;
```

図 9 マシンの状態遷移記述の全体構成
Fig. 9 The syntax of machine state transition description.

```
state ダイヤル中;
    input 発呼 ? オンフック;
        nextstate アイドル;
    input 発呼 ? ダイヤル;
        decision;
        (完了):
            output 発呼 ! ダイヤル成功;
            output 着呼 ! 呼び出し;
            output NW ! 予約;
            nextstate 呼び出し中;
        (未完):
            nextstate ダイヤル中;
        enddecision;
```

図 10 自動生成されたマシンの状態遷移記述の抜粋
Fig. 10 A part of generated machine state transition description.

例を用いて，自動生成したマシンの状態遷移記述におけるプロセス記述の抜粋を**図 10** に示す．抜粋は要求の状態遷移図において，ダイヤル中状態からの 3 つの遷移（発呼のオンフックによるアイドルへの遷移，ダイヤルの完了による呼び出し中への遷移，ダイヤル未完によるダイヤル中への遷移）に対して生成された仕様である．マシンの状態遷移記述のうち，state，input，decision，nextstate の各文は要求の状態遷移図から生成する．output 文は，要求の状態遷移図中のドメイン状態の変化と，ドメインの状態遷移図を調べることによって生成する．たとえば，ダイヤル中状態からダイヤルを入力しダイヤル完了条件が成立して呼び出し中に遷移する場合，要求の状態遷移図より，ドメインの状態は，「発呼：ダイヤル中 → 呼び出し中，着呼：スタンバイ → 呼び出し中，NW：スタンバイ → 予約中」と変化する（課金は変化しない）．発呼ドメインの状態遷移図（図 4）により，発呼にダイヤル成功を出力すればダイヤル中から呼び出し中に変わることが分かる．そこで，

表 2 システムの操作性の実験に対する問題のサイズ
Table 2 The experimental problem size.

	ドメイン数	ドメイン状態遷移図				要求		仕様	内容
		入力	出力	状態数	遷移数	状態数	遷移数	行数	
問題1	8	7	15	17	22	6	16	74	電子レンジ
問題2	7	4	13	16	19	8	8	46	洗濯機

表 3 問題記述に要した時間（分）
Table 3 The time (minutes) required to express the problem.

問題	システムの適用	プロブレム図	現象表	ドメイン状態図	要求状態図	仕様	エラー修正	合計
問題1	システム	8	7	10	15	0	20	60
問題2	システム	7	5	8	13	0	0	33
問題2	手書き	14	6	15	14	16	11	76

“output 発呼!ダイヤル成功;” が生成される．同様にして，“output 着呼!呼び出し;” と “output NW!予約;” が生成される．ここで，出力イベントは “ドメイン名!イベント名” と記述する．

マシンの仕様としては，SDL 記法に加えて，C 言語プログラムを生成することも可能である．これを実行することにより，仕様の実行シミュレーションができる．

4. システムの評価

4.1 適用性について

2 章でも述べたように，本システムはプロブレムフレームの考え方に対して以下のような制約を持つ．

① 振舞いフレームに限定．
② 副問題の分解や合成の機能はない．
③ 1 つのプロブレム図には要求とマシンはそれぞれ 1 つのみ．
④ マシンと直接関係のあるドメインしか持たない．
⑤ マシン以外のドメインどうしのインタフェースは持たない．
⑥ ドメインは有限状態マシンでモデル化できること．
⑦ 仕様現象はイベント，要求現象は状態に制限．

このうち，①，⑥，⑦は我々の基本的な方針による．また，②，③を解消するにはさらなる研究が必要であり，本研究のスコープから外した．④，⑤は，状態マシンの合成 [14] 等を導入することによって制限を除くことができると考えており，今後の機能強化の対象と考えている．

これらの制約のもとで，以下に示すような，組込みシステムの例題に対して本システムが有効に適用できることを確かめることができた．

- Jackson の本の振舞いフレームの例題：片側交互通行用信号，水門制御問題等
- 学生教育向けの題材：炊飯器，洗濯機，電子レンジ，簡単な自販機，エレベータ制御等
- その他：少し複雑な ATM，懸賞付き自販機 [15]，電話交換機等

このうち，洗濯機制御では，④の制約のため「水槽の水」ドメインが使えず，給水栓や排水栓と水位センサの関係が記述できないという問題が生じた．この解決策としては，これらの 3 つのドメインをまとめて 1 つのドメインとすることで対処できる．

なお，本システムが適用できない例として，飛行船制御があった．これは，プロペラの出力レベルと飛行船の動きの関係が連続的で有限状態マシンのモデルに合わないためであった．

4.2 操作性について

支援システムの操作性に対する評価として，比較的簡単な 2 つの問題（電子レンジ制御と洗濯機制御）にシステムを適用した．これらの問題の規模を**表 2** に示す．また，**表 3** にこれらの問題の分析・記述に要した時間を示す．なお，問題 2 は，比較のため手書きも行った．手書きは，パワーポイントでコピー&ペーストを多用しながら作成した．なお，問題 2 の手書きでのエラー修正時間は，マシン仕様に対するレビューと修正にかかった時間である．

問題 1 は，UML による設計（クラス図と状態遷移図）が存在しており，それをもとに問題図やドメインの状態図の構成がほぼ頭の中で完成していた状態で始めた．また，要求は，少し考えながら作成した．その結果，要求の誤り（遷移漏れ）2 件とドメイン（タイマ設定）の誤り 2 件のエラーが出て，それに対する対応策の検討とシステムを使った修正作業に 20 分かかった．問題 2 は，すでに問題の分析は終わっており，すべてのドメインの状態遷移と要求の状態遷移について頭に入っている状態で始めた．この場合は，システムの操作にかかった時間はほぼ入力操作のみである．

このサイズの問題だと，プロブレム図に 15 分，ドメイン

状態遷移図に 10 分，要求状態遷移図に 15 分で，合計 40 分程度で記述できる．問題 2 の場合は，作図に 33 分であり，手書きの場合は 49 分（仕様とエラー修正以外の部分）であり，作図に対するシステムの効果として 3 割削減であった．ただし，要求記述では，パワーポイントでの要求状態のコピー&ペーストを効果的に使えたので，手書きでもほとんど差がなかった．時間的に最も効果があるのはマシン仕様の自動生成であった．エラーのない仕様作成まで含めると，システムの効果は 76 分/33 分=2.3 倍であった．これより，支援システムの効果が確かめられる．

4.3 支援機能の有効性について

(1) 要求記述のガイド機能

我々は以前類似のツールを開発し UML との比較実験を行った [16]．簡単な例題の状態遷移図を 6 名の学部学生（卒研生）に UML と提案方式で作成してもらい評価した．時間的にはほとんど差がなかったが，結果を見ると，UML では被験者による差が大きかったが，提案方式では差が少なかった．これより，要求のガイドによって，誰もが同じような個人差のない要求記述ができるという効果が期待できる．

4.2 節の電子レンジの例ではもともと UML の記述例があり，それをもとに提案システムで作成した．UML の例では，制御部の状態は環境の状態と対応しておらず，たとえば，ドアの開閉は状態の明示的な変化を起こさないという設計であった．調理中にドアを開くとどうなるかは，状態遷移図中の内部処理の変数の代入や値の判定文を調べてやっと分かるようになっていた．一方我々のシステムでは，要求の状態はドメインの状態を 1 つに決める必要があり，ドアをドメインとするとドア開とドア閉は別の要求状態となる．これによって，調理中にドアを開ければ停止する等の本質的な振舞いが，要求の状態遷移として明示的になるという効果があった．

(2) チェック機能

電話交換機の問題分析では不完全な要求から完全な要求に至る方式を試みた．すべてのドメインの状態遷移図を作成後，まず初めに簡単な不完全な要求記述として，

アイドル=[A?offHook] ⇒ ダイヤル = [A?dial]
⇒ 通話 = [A?onHook] ⇒ アイドル

（A は発呼ドメイン）なる状態遷移を入力した．このチェック結果として，ダイヤル⇒ 通話と通話⇒ アイドルで着呼ドメイン (B) の状態変化は出力遷移だけでは不可能（入力遷移が必要）であるとのエラーが出た（3.5 節 (1–2)）．つまり，これらの遷移の途中に入力待ちの要求状態が抜けていると推測される．そこで，これらの遷移の途中に，着呼ドメインからの入力イベント（B?offHook と B?onHook）を待つ状態として，B?offHook 待ちの呼出中と B?onHook 待ちの **B 終了待ち**を挿入した．これによって，

アイドル → ダイヤル → 呼出中 → 通話
→ **B 終了待ち** → アイドル（イベント省略）

の主ループができた．再び，これに対するチェックで，ダイヤルと呼出中で A?onHook イベントが，また通話で B?onHook イベントが発生しうるとのエラーが出た（3.5 節 (2)）．この修正として新たな遷移を追加し，さらに，dial 入力時の判定（未完，完了，ダイヤルエラー等）を加えて最終的な電話交換機の要求状態遷移図が得られた．このように，チェック機能を活用し，断片的な要求から徐々に完全な要求を得ることができた．

複雑な ATM の例では，要求の状態遷移図を完成させた後のチェックで，「パスワードエラー時の再入力」と「銀行での認証中」という 2 つの状態からキャンセルが押されたときの遷移が抜けているというエラー（3.5 節 (2)）が出た．そこで，キャンセルによりカードを排出し「カード取出し待ち」状態へ移る遷移を追加した．この修正によって，今度は銀行のドメインで，認証中からアイドルへの出力遷移がないというエラー（3.5 節 (1–2)）が出た．つまり，銀行ドメインは，認証中からは，銀行の判断による遷移（パスワードエラー，ID エラー，認証 OK 等による遷移）のみ存在し，ATM 装置からの指示による出力遷移がなかった．そこで，銀行ドメインの状態遷移図に認証中から認証中断命令によるアイドル状態への遷移を新たに追加し，エラーをなくすことができた．これらのエラーは分析者の事前レビューでは発見されなかったもので，システムのチェック機能の有効性が確認できた．

(3) 仕様生成機能

4.2 節で述べたように，簡単な例でも仕様生成機能によって手書きの場合の約 2 倍の効果があった．複雑な ATM の例では，ドメイン数は 8 個，ドメインの状態数は 37，遷移数は 63（うち入力遷移は 15），要求の状態数は 14，遷移数は 22 となり，その結果自動生成された仕様記述は 169 行であった．このように複雑なマシンの仕様をいきなり間違いなく作るのは大変だが，要求とドメインの状態遷移図を作成し，それからマシンの仕様を自動生成することにより，複雑なものも誤りなく作成することができた．

5. 関連研究

5.1 プロブレムフレーム

Jackson のプロブレムフレームの本 [4] が出て以来ほぼ 10 年を経たが，この間，プロブレムフレームの研究として，新しいフレームの提案，副問題の合成と分解，要求からの仕様生成，ツール化等の研究が行われてきた [17]．このうち，ツール化と仕様生成が本論文のテーマとなる．

ツール化については，Ourusoff [18] は，教育の立場もふ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ツールとして OpenPF [5] 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エベント計算（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かなり難しいと思われる．

適用分野を組込みシステムによく出てくる振舞いフレームに限定すると，実務者向けのプロブレムフレーム用ツールとして，UML を活用する方式が提案されている [6], [7]．これらは，既存の UML ツールを使って，問題を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少し無理がある．たとえば，プロブレム図をクラス図で表現した場合，外部のドメインと解となるマシンがともにクラスとなり，その区別が明確でない．また要求はクラス図には表せない．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となり，実装を意識した表現になってしまう．

要求からの仕様生成の研究としては，Seater ら [19] の problem frame transformations や Rapanotti ら [20] の problem reduction 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持たない）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．要求やドメインの記述として，前者は Alloy [21] の記述を，後者は Gentzen 風の定理証明記述を使うため，手法としては汎用的で，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．たとえば，Seater らの例では「片側交互通行用信号」問題 [4] で，「車」という深いドメインから始め，もとの要求を「区間内には同時に北向きと南向きの車がいることはない」とし，そこから最終的には赤信号と緑信号のパルス発生数の規定という仕様に変換している．一方，Jackson [4] は同じ問題で間接ドメインはなく，要求は，「信号 1 と 2 の状態を，{50 秒停止，120 秒北向き通行，50 秒停止，120 秒南向き通行 } の繰返しとする」のように書いている．Seater らと Jackson のどちらが良いかは場合によるが，我々のシステムは Jackson の表現に即した状態マシン記述を取り入れて，仕様生成機能を実現した点で，他の仕様生成の研究とは異なるといえる．

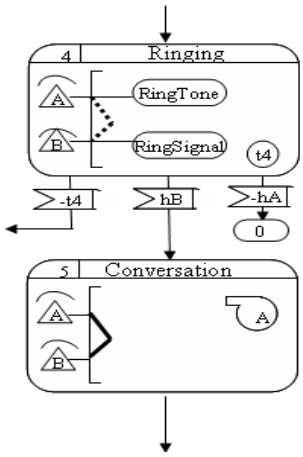

図 11 SDL の Pictorial Elements
Fig. 11 SDL state diagram with pictorial elements.

5.2 プロブレムフレーム以外

一般のモデル検査 [11] では，Kripke 構造は結果の解析のために利用するのに対して，我々の方式では，要求の記述として直接 Kripke 構造を利用する．つまり，Kripke 構造を，分析者や設計者が考えている要求を表現するために利用している．また，我々のモデルでは，Kripke 構造での原始命題を外部環境であるドメインの状態とし，原始命題間の関係をドメイン状態遷移図で規定する．これによって，各状態で発生すべきイベント等の制約条件が求まり，その制約を要求規定のガイドとして利用し，仕様となる状態遷移記述の自動生成に利用することができる．

初期の SDL [22] の Pictorial Elements（**図 11**）の概念は，状態指向の SDL を定義していたが，その扱いがコメント的であったため，現在の SDL [9] では除かれている．我々のシステムでは，要求の状態遷移図は，Pictorial Elements をドメインの状態に対応させることにより，状態指向の SDL 状態遷移図を形式化したものであるととらえることができる．

6. おわりに

プロブレムフレームにおけるプロブレム図を作成し，それをもとにドメインや要求の状態遷移図を作成し，それからマシンの状態遷移仕様を自動生成する，プロブレムフレームを用いた状態遷移分析の統合的なシステムについて述べた．

対象問題としては，組込みシステムでよく出てくる振舞いフレームに限定し，振舞いフレームに適した状態マシンをドメインや要求の記述に適用した．プロブレム図ツールでは，後でドメインや要求の状態遷移図を作成することをふまえ，ドメインのイベント（仕様現象）や状態（要求現象）の管理を重視した．ドメインの状態遷移図ツールでは，プロブレム図との互換性を重視し，プロブレム図で記述した情報から容易にドメインの状態遷移図を作成できる

ようにした．要求の状態遷移図ツールでは，すべてのドメインの状態遷移図の規定に沿った要求の記述をガイドできるように，要求の各状態でドメイン状態を選択できるようにした．マシンの状態遷移記述を自動生成する機能は，プロブレム図やドメインの状態遷移図，要求の状態遷移図の分析を行った後，その結果のまとめとして，次の段階へと進んだソフトウェアの設計に有効なマシンの仕様を作成することを目的とした．さらに，これらの図の間の整合性をチェックする機能により，ドメインや要求の記述の誤りや遷移抜け等を自動的に検出できるようにした．これによって，複雑なシステムの分析においても，各ドメインの規定（所与）と要求の記述（願望）を分離し，それぞれは比較的単純で理解しやすい形で記述でき，それらの合成結果としてマシンの状態遷移仕様を誤りなく生成できるようにした．4.1 節でも述べたが，支援システムはいろいろな制限がついているが，それにもかかわらず多くの組込みシステムの要求分析に適用でき，その効果を確かめることができた．

今後の課題として，まずは評価の多様化をあげる．今回の評価実験は開発者自らシステムを使用しての考察であり，主観的であったのも事実である．今後は第三者による客観的な評価を行いたい．さらに実システムに対しての適用・評価も行っていきたい．次に，システムの機能強化として，より複雑な場合の対処が課題となる．1 つは，間接ドメインの導入である．これらの間接ドメインの規定（ドメインプロパティ）を状態マシンモデルで表せる場合は，状態マシンの合成を使って，要求からの仕様生成を実現できると思われる．ただし，5.1 節で述べたように，深いドメインではそのプロパティは状態マシンより述語論理的な表現が適切かもしれない．第 2 の強化として，プロブレムフレームの重要な概念である副問題への分割・合成機能の研究開発がある．これには，プロブレム図の分割や合成にともなって，ドメインや要求の状態遷移図を適切に分割・合成する方式の研究が必要になろう．これができれば，複雑な問題は，いくつかの比較的簡単な副問題を合成することによって自動的に得ることができる．今後，これらの課題への対応を進めていきたい．

参考文献

[1] OMG: UML 2.0 Superstructure Specification (2004), available from ⟨http://www.omg.org/⟩.

[2] IBM Rational ソフトウェア，available from ⟨http://www-06.ibm.com/software/jp/rational/⟩.

[3] JUDE/Biz, available from ⟨http://jude.change-vision.com/jude-web/index.html⟩.

[4] Jackson, M.: *Problem Frames: Analyzing & Structuring Software Development Problems*, Addison-Wesley Publishing Company (2001).

[5] Tun, T.T., Yu, Y., Haley, C. and Nuseibeh, B.: Model-based Argument Analysis for Evolving Security Requirements, *4th International Conference on Secure Software Integration and Reliability Improvement*, pp.88–97 (2010).

[6] Lavazza, L. and Bianco, V.D.: A UML-based approach for representing problem frames, *Proc. 1st International Workshop on Applications and Advances of Problem Frames*, pp.39–48 (2004).

[7] 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).

[8] 紫合　治：ジャクソン法（JSP）による状態遷移設計，情報処理学会論文誌，Vol.50, No.12, pp.3041–3051 (2009).

[9] Bray, I.K.: *An introduction to requirements engineering*, Addison Wesley (2002).

[10] Alfaro, L. and Henzinger, T.A.: Interface Automata, *ACM SIGSOFT, FSE01*, pp.109–120 (2001).

[11] Clarke, Jr., E., Grumberg, O. and Peled, D.A.: *Model Checking*, The MIT Press (1999).

[12] 紫合　治：状態指向のステートチャート，ソフトウェア工学の基礎ワークショップ（FOSE05），pp.25–30 (2005).

[13] 若原　恭，長谷川晴朗：仕様記述言語 SDL，株式会社カットシステム (1996).

[14] Magee, J. and Kramer, J.: *Concurrency, State Models and Java Programming*, John Wiley and Sons, Ltd. (2006).

[15] 中谷多子，青山幹夫，佐藤啓太：ソフトウェアパターン，共立出版 (2001).

[16] 大川　敦，加藤大輝，紫合　治：インタフェースの情報を用いた状態遷移図の作成，情報処理学会研究報告 2006-SE-151，pp.33–40 (2006).

[17] Cox, K., Hall, Jon, G. and Rapanotti, L.: Editorial: A roadmap of Problem Frames research, *Information and Software Technology*, Vol.47, Issue 14, pp.891–902 (2005).

[18] 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).

[19] 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).

[20] 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).

[21] Jackson, D.: Alloy: A Lightweight Object Modeling Notation, *ACM Trans. Software Engineering and Methodology*, Vol.11, No.2, pp.256–290 (2002).

[22] CCITT: Recommendations Z.101 to Z.104, Functional Specification and Description Language (SDL), Yellow Book, Volume VI=Fascicle VI.7, Geneva (1981).

紫合 治 （正会員）

1971年東京電機大学工学部電子工学科卒業．同年日本電気(株)入社．ソフトウェア生産技術研究所，NECアメリカ等にて，ソフトウェア工学，インタネットセキュリティ等の研究開発に従事．2003年より東京電機大学情報環境学部教授，現在に至る．情報処理学会20周年記念論文賞（1980年），大河内記念技術賞（1984年），情報処理学会山下記念研究賞（2010年）受賞．ソフトウェア科学会会員．

横山 薫

2009年東京電機大学情報環境学部情報環境学科卒業，2011年同大学大学院情報環境学研究科修士課程修了，同年4月NECソフト株式会社に入社，現在に至る．在学中はソフトウェア分析設計支援ツールに関する研究に従事．