命題論理に基づいたやわらかいソフトウェア設計支援システムとその応用

福沢 尚司　宋 国煥　富樫 敦　白鳥 則郎
東北大学電気通信研究所 / 情報科学研究科　〒980-77 仙台市青葉区片平 2-1-1　電話：(022)217-5454
電子メール：{fukuzawa,song,togashi,norio}@shiratori.riec.tohoku.ac.jp

あらまし：　通信システムの分野において，これまで高水準の情報サービスのための形式的仕様の記述によるソフトウェア開発法が研究されてきた．そのための記述言語として形式記述技法 (Formal Description Techniques : FDTs) が提案されてきたが，ユーザの要求と形式記述との間の隔たりは大きく，その記述は困難であった．その上，システム要求は開発の初期段階においてしばしば変化し，その影響はシステム全体に影響を及ぼす．そこで我々は命題論理に基づいたやわらかい方法論を提案してきた．これはシステム要求の変化に対応して形式仕様を自動的に獲得するものである．また，我々はシステム要求の論理エラー検証法，詳細化法についても研究を行ってきた．本稿では，我々のソフトウェア開発法を支援するやわらかい設計支援システムについて述べる．さらに，この支援システムを用いた応用例として，実際に CATV システムを設計し，通信ソフトウェア設計への適応性を示す．

A Support System for Flexible Software Design based on Propositional Logic and Its Application

Shoji Fukuzawa, Kukhwan Song, Atsushi Togashi and Norio Shiratori
{Research Institute of Electrical Communication, Graduate School of Information Sciences} Tohoku University, 2-1-1 Katahira, Aoba, Sendai 980-77, Japan　phone: (022)217-5454
email: {fukuzawa,song,togashi,norio}@shiratori.riec.tohoku.ac.jp

Abstract:　In the field of communicating systems, the approach describing specifications formally has been researched before in order to develop the (communication) software which provide the advanced information services, so that Formal Description Techniques (FDTs) have been proposed as specification languages. However, the gap between the specification and user requirements is so large that it is very difficult to describe system requirements. Futher more, it is often necessary to modify or change system requirements which may influence the whole system design in the early stage. Therefore, we have proposed a new flexible methodology for describing system requirements based on propositional logic, which copes with the modifications or changes in the system requirements, in order to obtain the formal specification of the system automatically. We have also proposed a verification method for system requirements that may contain some kinds of logical errors, and a refinement mothod of system requirement. In this paper, we discuss a configuration of a flexible design support system supporting our software development mothodoligy. We also show that a CATV system as a real application example of communication software can be applied well in our support system.

1 はじめに

情報処理システムが大規模複雑化すると共に，その設計のための効率的で信頼性の高い形式記述法が求められている．通信システムの分野においては，これまで高水準の情報サービスのための形式仕様の記述によるソフトウェア開発法が研究されてきた．そのための記述言語として形式記述技法 (Formal Description Techniques : FDTs), SDL, Estelle, LOTOS 等が提案されてきたが，その読解，記述は厳密で難解であり，ユーザの要求と形式記述との間の隔たりは大きく，その記述は困難であった．その上，システム要求は開発の初期段階においてしばしば変化し，その影響はシステム全体に影響を及ぼす．そこで我々は命題論理に基づいたやわらかい方法論 [4, 5] を提案してきた．これはシステム要求の変化に対応して，FDTs の基本構造であり，それ自身形式記述技法として用いられる状態遷移システム (State Transition System: STS) を，システムの形式仕様として自動的に獲得するものである．また，我々はシステム要求の論理エラー検証法，詳細化法についても研究を行ってきた [2].

本稿では，我々のソフトウェア開発法を支援するやわらかい設計支援システムと，その実装について述べる．さらに，この支援システムを用いた応用例として，実際に CATV システムを設計し，通信ソフトウェア設計への適応性を示す．

本稿の構成は以下の通りである．まず第 2 節ではシステム要求から STS の合成法，論理エラー検証法および詳細化法を含む我々のやわらかいソフトウェア開発法について述べる．第 3 節では設計支援システムと，CATV を例にその応用を示す．第 4 節はまとめである．

2 やわらかいソフトウェア設計支援

2.1 やわらかいソフトウェア開発

近年，“やわらかいシステム”という新しい考えが，高度情報システムの鍵となる概念として研究されてきた [6]．やわらかいシステムの概念に基づいて，我々はやわらかいソフトウェア開発法を次の性質を満たすものとして考える．

- ユーザの要求の変化に適応できる．
- 形式的でない，不完全である，あいまいであるといった性質を持つユーザの要求を，正しく獲得できる．
- ユーザの要求が段階的に形成されるのにあわせ，機能の拡張を行える発展性を持つ．

ソフトウェア開発は，ソフトウェアに対するユーザの要求から形式仕様への，また，形式仕様から実行可能なプログラムへの変換を，自動合成や検証のための様々なツールを用いて行う一連の流れとみなすことができる．我々はこれを一般にユーザ要求の集合 $\mathcal{R}$ から形式仕様の集合 $\mathcal{S}$ への変換 $\mathcal{T} : \mathcal{R} \rightarrow \mathcal{S}$ と考えることとする．形式仕様の集合 $\mathcal{S}$ からプログラムの集合 $\mathcal{P}$ への変換も同様に考える．本稿では $\mathcal{T} : \mathcal{R} \rightarrow \mathcal{S}$ の部分に議論の範囲を絞る．以下に，やわらかいソフトウェア開発法を定義する．

定義 2.1 （やわらかいソフトウェア開発法） $\mathcal{R}$ を要求の集合，$\mathcal{M}$ を $\mathcal{R}$ を満たすシステムとする．$\mathcal{R}'$ をユーザの要求の変化によって $\mathcal{R}$ から変更の加わった要求の集合とし，$\Delta\mathcal{R}$ を $\mathcal{R}'$ の $\mathcal{R}$ からの変更部分の集合とする．$\mathcal{R}$,$\mathcal{R}'$,$\mathcal{M}$ が与えられ，$\mathcal{M}$ は $\mathcal{R}$ を満足し，$\mathcal{R}'$ を満足しないと仮定する．開発法 $\mathcal{D}$ は，$\mathcal{R}'$ を満足するシステム $\mathcal{M}'$ を $\mathcal{M}$ と $\Delta\mathcal{R}$ より $\mathcal{D}$ を用いて合成可能であるとき，やわらかいソフトウェア開発法であるという． □

2.2 システム要求から形式仕様の合成

我々は，やわらかいソフトウェア開発法の実現のために，ユーザの要求からシステムの要求記述の獲得と，そこから形式仕様を合成する新しい方法を提案してきた [3, 5]．ここでその方法のあらましを述べる．

ρ を，システムの要求記述のための命題論理における素命題の集合とする．命題論理は目的のシステムに依存し，それぞれの素命題により，システムの具体的な性質を表す．通常，命題論理は素命題と論理演算子 $\wedge, \vee, \neg$ により構成される．一般に，命題論理は加法標準形 $\gamma_1 \vee \ldots \vee \gamma_n$ で表される．ここで，γ_i はリテラル[1]の論理積である．システム要求は本質的に，システム実行時における基本的な機能の羅列によって記述される．

定義 2.2 （機能要求） 機能要求は 5 項組 $\phi = \langle id, i, f_{in}, o, f_{out} \rangle$ であり，ここで id は機能名，i は機能の入力記号，f_{in} は機能の実行前にシステムが満たすべき条件であり，ρ に含まれる素命題を用いたリテラルの積で表すもの，o は機能の出力記号，f_{out} は機能の実行後にシステムが満たすべき条件であり，ρ に含まれる素命題を用いた命題論理で表すものである． □

機能要求 ϕ は $id : f_{in} \xrightarrow{i/o} f_{out}$ と表すことがある．o は省略可能である．

定義 2.3 （要求記述） システムの要求記述は次の 3 項組 $\mathcal{R} = \langle \Phi, \gamma_o, \gamma_e \rangle$ であり，ここで Φ は機能要求の集合，γ_o は ρ に含まれる素命題を用いたリテラルの積で表される初期条件，γ_e は ρ に含まれる素命題を用いた命題論理で表される最終条件である．γ_e は省略可能である． □

本稿では形式仕様として，状態遷移システムを考える．

[1] リテラルとは，素命題または素命題の否定 ($\neg$) である．

定義 2.4 （状態遷移システム） 状態遷移システムは，6項組 $M = \langle Q, \Sigma, \Delta, \rightarrow, q_o, Q_e \rangle$ であり，ここで Q は状態の集合，Σ は入力記号の集合，Δ は出力記号の集合，$\rightarrow$ は状態の遷移（とそれに伴う出力）の関係であり，$\rightarrow \subset Q \times (\Sigma \times \Delta) \times Q$ で定義される．q_o は初期状態，Q_e は最終状態の集合である． □

状態遷移システム M において，通常 $(p, i, o, q) \in \rightarrow$ を $p \xrightarrow{i/o} q$ と表記する．この表記は，“状態 p において，入力 i が与えられると，システムは出力 o を伴って状態 q に移行する”ことを意味する．ここで，状態遷移システムにおける全ての状態は初期状態から到達可能であると仮定する．さらに，ある素命題 A と状態 $q \in Q$ について，A が q において成立しているかどうか，つまり q における A の真理値があらかじめ定義されていると仮定する．

本稿では，論理ペトリネット (Logical Petri Net: LPN)[5, 3] を用いる．LPN は，抑止アークを入出力の双方に許すことで，機能要求中の命題論理における否定を表現するものである．LPN は，5項組 $LPN = \langle P, T, I, O, \mu_o \rangle$ であり，ここで P, T はそれぞれプレースとトランジションの集合，$I = \langle I_p, I_n \rangle$ は入力関数の組，$O = \langle O_p, O_n \rangle$ は出力関数の組，$\mu_o : P \rightarrow \{0, 1\}$ は初期マーキングである．LPN には2種類のアーク，ポジティブアークとネガティブアークが存在し，それぞれ一般のペトリネットにおけるアークと抑止アークに相当する．詳細は [5, 3] による．

システムの要求記述は LPN に変換され，LPN をその実行アルゴリズム [1] に従って発火させることにより，形式仕様を自動的に合成する．

2.3 論理エラーの検証

システム要求には，初期の要求相互の衝突のために論理エラーが含まれる場合がある．システム要求に含まれる可能性のある以下の4種類の論理エラーが検出可能である．

(1) 最終状態成立不可能： 初期状態からどの機能を用いても最終条件が満たされる状態（最終状態）への到達が不可能であること．

(2) デッドロック： 最終状態を除く，初期状態から到達可能なある状態において，実行可能な機能が存在しないこと．

(3) 発生不可能な機能： ある機能が，そのシステムにおいて全く実行されないこと．初期状態より到達可能で，最終状態であって初期状態とは異なるものを除く全ての状態で，その機能要求の実行前条件が満たされないこと．

(4) 無限ループ： 初期状態から到達可能なループが存在し，そこに含まれるいかなる状態からも最終状態への到達が不可能であること．

論理エラーは LPN を用いて検出される．要求記述から LPN への変換を行ない，LPN の可達木を生成してそれを調べることにより，上述の論理エラーを検出することができる．それとともに，可達木中において，ある遷移をたどった後に最終状態への到達可能性を失う最初の遷移 (critical arcs) を見つける [3].

次に，検出された論理エラーの要求段階への反映について述べる．やわらかいソフトウェア開発法の定義より，やわらかい開発法は検出された論理エラーをユーザの要求を獲得する段階に役立てることができるとき，よりやわらかいと言える．我々の方法では形式仕様のシステム要求からの合成を基礎としており，また論理エラーの原因は初期の機能要求の衝突にあるため，論理エラーは形式仕様の段階で現れるが，システム要求の段階で取り扱う立場に立つ．論理エラーの修正のためにシステム要求を書き換える方法は数多く存在すため，修正法の候補をガイドラインとして提案することで，論理エラーのシステム要求への反映法としている．反映法は次の4ステップで記述される．

Step 1: 論理エラーの検出

システムの要求記述から合成された形式仕様に論理エラーが存在するかどうかを上記の方法で調べる．

Step 2: 論理エラーの種類分け

Step 1 で論理エラーが検出された場合，その種類と，それに関連する情報を調べる．

Step 3: ユーザに論理エラーを示す

検出された論理エラーの種類と，それが起こった形式仕様上での位置をユーザにグラフィカルに示す．このステップにより，ユーザは問題を理解し，システム要求を修正する必要性を考慮することができる．

Step 4: ユーザにガイドラインを示す

論理エラーを種類に応じて分析し，ユーザにガイドラインを提示して論理エラーを解消するための要求記述変更の助けとする．

2.4 システム要求の詳細記述

本稿ではシステム要求の詳細化を，素命題の集合を分解して，それぞれの詳細化レベルでシステム要求が使用可能な素命題の集合とする方法を用い，目的のシステムから得られる素命題間の制約に基づいてその素命題の集合の分解を行なうことにより実現している．これにより，それぞれの詳細化レベルにおいて独立に機能を考えられる．素命題間の制約 $A_{\Leftarrow\gamma}$ の直接的な意味は，状態遷移システム中の，条件 γ が満たされる状態においてのみ，素命題 A で表される新しい概念と，A に関する新しい機能（その実行前条件に A または $\neg A$ が含まれる機能）が定義され得ることである．言い替えれば，素命題 A は条件 γ を詳細化した概念であり，条件 γ が満たされる状態において，新しい性質 A または $\neg A$ が成立する可能性と，それらに関する新しい機能が考えられるという意味である．

我々の詳細化法を用いた形式仕様合成の手順は以下に要約される．

(1) 素命題の集合 ρ を得る．ρ は目的のシステムの性質を表すものである．

(2) ρ を分析し，制約の集合 $\mathcal{C}$ を得る．
(3) ρ を $\mathcal{C}$ を用いて，階層的に $\rho(\eta_i)$[2]. に分解する．
(4) $\rho(\eta_i)$ を用いて，それぞれの階層におけるシステム要求 $\mathcal{R}(\eta_i)$ を記述する．
(5) それぞれの階層の形式仕様 $M(\eta_i)$ を $\mathcal{R}(\eta_i)$ より得る．

3 設計支援システム

3.1 環境と構成

ここでは，我々の開発法を支援する設計支援システムの実装について，反映器とユーザインタフェースに焦点を当てて説明する．これらにより，詳細化された記述を可能にしたシステム要求 [2] を獲得することができ，やわらかいシステムを実現することができる．図 1 に設計支援システムの概略を示す．

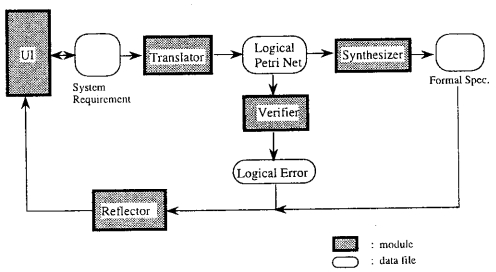

図 1： 設計支援システム概略図

設計支援システムは，ユーザからシステム要求を獲得するためのユーザインタフェース，LPN への変換器，状態遷移システム (State Transition System: STS) 合成器，論理エラー検出器，反映器から構成される．

- ユーザインタフェース
  ユーザインタフェースは GUI(graphical user interface) を用いてシステム要求の獲得を簡単に行なえるよう工夫したものである．入力すべきデータについての簡単な説明が示され，ユーザはそれを参照しながら，ボタン，メニュー，アイコン，スクロールバー等を用いた視覚的にわかりやすい操作による入力を行なうことができる．さらに，ユーザは制約に関する情報を与えるだけで，機能の分割によって詳細化されたシステム要求 [2] を記述することができる．

- LPN への変換器
  システムの要求記述は，2 つの変換規則を適用することにより，標準形 (canonical form) に変換される [3]．1 つは選言を除去するための規則であり，もう 1 つは実行前条件に含まれ，実行後条件に含まれない条件の保存規則である．さらに標準形は LPN へと変換される．

- STS 合成器
  形式仕様として，STS を LPN 形式のシステム要求から合成する．LPN を発火させることにより生成される可達木を利用した合成法を用いる．

- 論理エラー検出器
  LPN とその可達木を調べることにより，論理エラーを検出する．エラーの種類と，そのエラーに関する情報を取り出し，次の反映器に渡す．

- 論理エラー反映器
  検出された論理エラーとそれを含む STS をユーザに示し，ユーザがシステム要求を変更してその論理エラーを解消する助けとなるガイドラインを提示する．ガイドラインにより，論理エラー解消のための要求記述，STS 双方に関する情報を適宜示す．

3.2 設計手順

ここでは，支援システムを用いてシステム要求と形式仕様を獲得する手順について述べる．この支援システムにおいて，ユーザは次の 4 ステップからなる作業手順に従って設計を行なう．

(1) システム要求名を与える
始めにシステムの要求名を与える．これは目的のシステムの要求記述を記録するファイル名として用いられる．要求名を与えたら，ユーザインタフェースのメインメニューで保存する．要求名は，新しい名前をタイプして与えるか，すでに存在する名前をファイル呼び出しのメニューで選択することによって与える．

(2) 基本情報を入力する
次にユーザは基本情報を専用ウィンドウ上で入力する．基本情報は素命題，入力記号および出力記号であり，それらを用いてシステムの要求記述 (Requirement Description: RDP) を行なう．このとき，素命題と入力記号の入力は必須であるが，出力記号の入力は省略可能である．

(3) RDP を入力する
RDP の入力を専用ウィンドウ上で行なう．このウィンドウでは命題論理の演算子を含むいくつかの演算子と，前ステップで得られた素命題，入力記号および出力記号がそれぞれメニューとなっている．ユーザはそれらメニューから選択していくことにより，RDP の入力を行なう．この方法を用いることにより，タイプミスによるエラーを防ぎ，記述を容易にする．

詳細化された RDP を入力する場合，RDP を記述する前に，素命題間の制約の入力を専用ウィンドウで行なう．制約は RDP の場合と同様にメニューから選んでいくことで入力する．与えられた制約に基づき，支援システムにより素命題の階層的分解が自動的に行なわれる．そして，分解された素命題の集合より，その詳細化レベルでの RDP の記述に使用できる素命題のメニューが作成される．

(4) RDP を実行する

最後に，メインウィンドウでRDPの実行を選択すると，STSを得てグラフィカルに表示させることができる．表示された状態および遷移は，マウスを使って自由にレイアウトを変更することができる．

3.3 支援システムの応用例

我々の設計支援システムの実際的な応用例として，簡単なCATVシステムに合成と詳細化を適用した設計の例を示す．CATVシステムの端末はホストコンピュータに接続され，コントローラのボタン操作により，いろいろなサービスをTV上で受けることができる．ここでは，CATVサービスの基本的な機能のみを考える．

(1) 電源の *on/off*, (2) チャンネルの *up/down*,
(3) 音量の *up/down*, (4) ミュートの *on/off*,
(5) チャンネルの指定, (6) プログラムタイマ,
(7) フォースチューン.

上記の機能(1)から(6)は一般的なTVの機能と同様であり，一方機能(7)はCATVシステムに特徴的な機能である．フォースチューンとは，ホストコンピュータからの信号によりある特定のチャンネルの放送を強制的に流す機能である．上記の機能を記述するため，次の4つの素命題を考える．

- *Power*: TVの電源がonである．
- *Mute*: ミュート機能によりTVの音声がoffになっている．
- （プログラムタイマの）*Set*: TV番組が予約されている．
- *Ftune*（フォースチューン）：ホストコンピュータにより，特定のチャンネルの放送が流されている．

このCATVシステムにおける入力記号は，コントローラの操作を表す{*power,ch-up,ch-dn,vol-up,vol-dn,mute,tenkey(x),set-pr*}と，ホストコンピュータからの信号を表す{*sig-ftune*}とする．*tenkey(x)*の意味はコントローラの数字キーxの入力によりチャンネルが選ばれることを示す．出力記号は例題の簡単化のため省略する．

基本情報の入力が完了すると，支援システムによりそれに基づいたRDPのためのメニューが作られる．RDPを全体またはある詳細化レベルの記述として，GUIを用いて設計する．

はじめに，全体的なRDPの入力の様子を図2に示す．上述の手順で与えられたメニューウィンドウからそれぞれの要素を選択していくことにより，入力を行なっている．

図2のRDPを入力後，実行ボタンをクリックすることにより，STSが合成される．

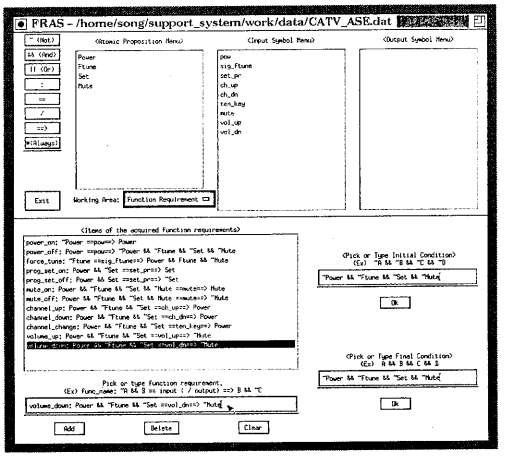

図2： 全体的なRDPの入力画面

次に，詳細化されたRDPの場合を示す．この例のCATVシステムにおいては，次の素命題間の制約Cが存在する．

$$C = \{Set_{\Leftarrow Power}, Mute_{\Leftarrow Power \wedge \neg Ftune \wedge \neg Set}\}.$$

素命題の集合$\rho = \{Power, Ftune, Set, Mute\}$は，制約により次の3段階の詳細化レベルに分解される．

$$\rho_1 = \rho(1) = \{Power, Ftune\},$$
$$\rho_2 = \rho(Power) = \{Set\},$$
$$\rho_2 = \rho(Power \wedge \neg Ftune \wedge \neg Set) = \{Mute\}.$$

図3にこの例における制約の入力を示す．この入力ウィンドウを利用して，素命題と画面左上に示されたオペレータより成るメニューから選んでいく形で制約を記述することができる．記述された制約に基づき階層的に分解された素命題の集合が自動的に得られる．図3の左上にある“Decompose”ボタンを押すことで，素命題の集合の分解が実行される．

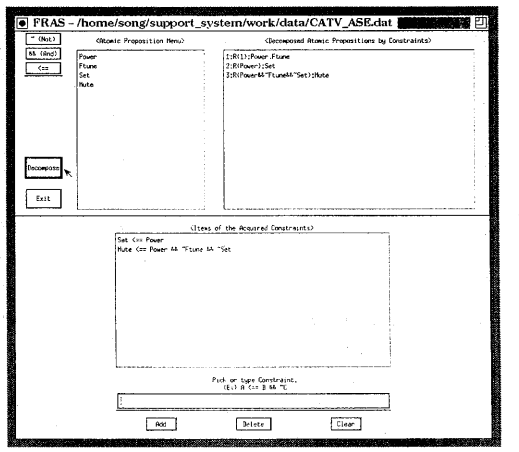

図3： 制約の入力による素命題集合の階層的分解

以上の手順により，この例において3つの階層化レベルのRDPを全体的なRDPと同様の方法で得ることができた．また，それぞれの得られたRDPを実行することにより，STSを獲得することができた．図4に階層化されたSTSを示す．

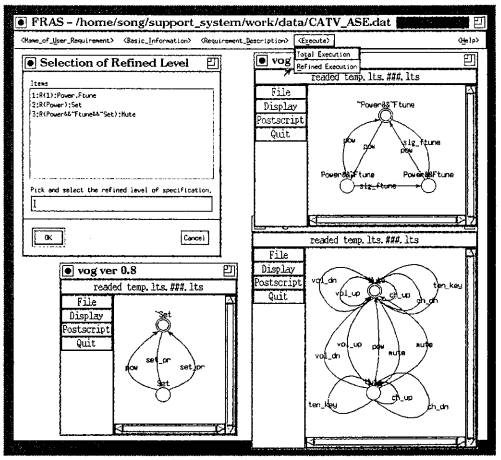

図4： 詳細化されたシステムのSTS

最後にRDPに論理エラーが含まれる場合について述べる．この場合，支援システムにより論理エラーの検出とそれに関する情報の抽出が行なわれ，ユーザに示される．その上で，検出された論理エラーの種類を選ぶことにより，それに関係する状態および遷移がSTS上に示され，その訂正のためのガイドラインを示すウィンドウが新たに開かれる．

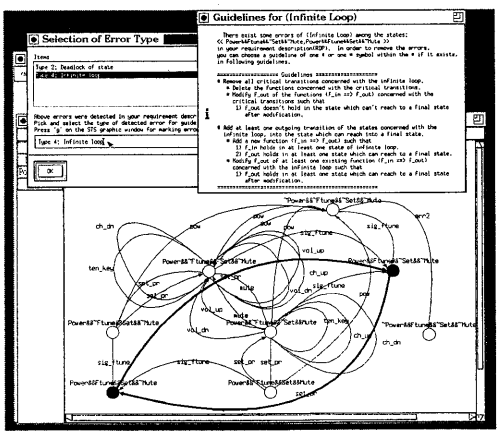

図5： 論理エラーの検出とガイドラインの提示

図5は，検出されたエラーのうち，無限ループを選んだ場合のガイドラインと，関係する状態，遷移が示されている画面である．

4 まとめ

本稿ではやわらかいソフトウェア開発法という新しい方法について論じ，その実現としてのソフトウェア開発法とその設計支援システムを紹介した．また，実装した支援システムの概要を，システム要求の記述を容易にするユーザインタフェースに焦点をあてて紹介した．さらには，実際のアプリケーションへの応用例としてCATVシステムの例を示し，コミュニケーションソフトウェアへの本開発法の適用性を示した．

最後に今後の研究課題を以下にまとめる．

(1) CBR(case based reasoning)の考え方を採り入れて，この設計支援システムをさらに使いやすくユーザフレンドリなものにする．

(2) 要求記述法(requirement description method: RDP)に命題論理だけでなく，述語論理やテンポラルロジックへの拡張を行なうことにより，記述性の向上を計る．

References

[1] Song K. H., Togashi, A., Shiratori, N., An Automatic Generation of Formal Specifications and Its Verification using Logical Petri Net, *Tech. Rep. of IEICE*, CST-94-29, pp.101–108, 1994 (in Japanese).

[2] Song K. H., Togashi, A., Shiratori, N., Verification and Refinement for System Requirements, IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E78-A, No.11, pp.1468–1478, Nov. 1995.

[3] K. H. Song, A. Togashi, N. Shiratori: A Requirement Description Method based on Propositional Logic and Its Semantic Description by State Transition System, to appear in Trans. of Information Processing Society of Japan, Vol. 37, No. 4, 1996 (in Japanese).

[4] K. H. Song, S. Fukuzawa, K. Takahashi, H. Kaminaga, N. Shiratori: A Flexible Design Support System for Requirement and Formal Specification, *Tech. Rep. of IEICE*, SSE-95-67, pp.79–84, 1995.

[5] Togashi, A., Usui, N., Song K. H., Shiratori, N. A Derivation of System Specifications Based on a Partial Logical Petri Net, to appear in the *Proc. of ISCAS'95*, Seattle, Washington, April 29- May 3, 1995.

[6] Shiratori, N., Sugawara, K., Kinoshita, T., Chakraborty, G., Flexible Networks: Basic Concepts and Architecture, IEICE Trans. Commun., Vol.E77-B, No.11, pp.1287-1294, 1994.