Vol.2013-SE-180 No.3
Vol.2013-EMB-29 No.3
2013/5/27

プロブレムフレームにおけるシーケンス図の活用

張　碩†1　　紫合　治†2

システムの要求分析に UML を適用することがあるが，UML はもともとシステムの解決策を検討するツールであり，初期の分析では解決策ではなく問題そのものを扱う「プロブレムフレーム」が有効に使える．プロブレムフレームではもとの問題をいくつかのサブ問題に分割することを推奨するが，サブ問題の規定の仕方，その振舞いの規定方法については特に明示されていない．ここでは，プロブレムフレームのサブ問題の規定にシーケンス図を使う方式について述べ，この方式を支援するシステムについて説明する．

Utilizing the Sequence chart in problem frames

ZHANG SHUO†1　OSAMU SHIGO†2

For analyzing the system requirements, UML has been utilized with its computer aided graphic tools. However, UML is basically applicable to express the solution structure of the system, not the problem itself. In the early stage of problem analysis, the concept of the problem frames seems to be more suitable to express the problem, not the solution. In the problem frames, sub-problem decomposition has a major role to analyze the problem. The paper proposes a method to utilize the sequence chart to represent the dynamic behavior of each sub-problem, and describes a graphic tool to support the method.

1. はじめに *【*の文字書式「隠し文字」】

従来，ソフトウェアの要求分析のツールとして UML がよく使われてきた．例えば UML のユースケースごとに主要フローと例外フローを規定する．フローの規定は一般にシナリオが使われ，シーケンス図などがよく利用される．シナリオやシーケンス図は問題のある側面の例示になっている．問題の分析に従って多くの例を調べ，シナリオやシーケンス図として規定し，これらの例の間の矛盾点を調べ，さらに全ての例を満足する一般則を見出し，システムの仕様を規定する．要求分析の一般的な手順としては，このような帰納推論がよく使われる．

要求分析の初期段階では，問題そのものを徹底的に調べることが重要であり，プロブレムフレーム[1]が有効に適用できる．プロブレムフレームでは問題の詳細化としてサブ問題分割を行うが，分割されたサブ問題の内容の規定方法については特に明示されていない．ここでは，サブ問題を例で表す方式を検討する．具体的には，サブ問題をシーケンス図で規定する方式とその支援ツールについて述べる．この支援システムによって，プロブレムフレームによる問題の分析能力に UML の振る舞い記述（シーケンス図や状態遷移図）を併用した支援がなされる，システムは，問題を一般的な機能記述ではなく，いくつかの例（動作例等）に分割し，分析し，最後にそれらを一般化した状態遷移仕様を得る機能を提供する．

†1 東京電機大学　情報環境学研究科
Graduate School of Information Environment, Tokyo Denki University.
†2 東京電機大学　情報環境学部
School of Information Environment, Tokyo Denki University.

2. 関連研究

プロブレムフレームに対するビジュアルな支援ツールとして，紫合等[2]は状態マシンをベースにしたツールを提案している．しかし，そこではプロブレムフレームの重要な概念であるサブ問題については何等支援していない．また，ドメイン規定については所与(given)を前提としているが，実際には問題の分析に従って徐々に詳細な規定がなされることも多く，その場合の支援機能も必要になる．

プロブレムフレーム専用のツールでなく，UML ツールによってプロブレムフレームを支援する方式も提案されている[3]．しかし，UML は本来，問題そのものではなく解決案を規定するためのものであり，例えばプロブレム図をクラス図で記述するのは無理がある．マシンとドメインの区別も明示的ではなく，共有現象の表現にインタフェースクラスが使われ，各現象がメソッドで表現される等，もとのプロブレム図に比べてかなり複雑になり，理解性が落ちる．

Bianco1 等[4]はプロブレムフレームにシーケンス図を併用する方式を提案している．そこでは，すべて UML で記述しているため，上記の UML ツールによる場合と同様の問題がある．また，ここでは，サブ問題という概念はなく，単に「多くのシナリオ」によって問題を分析することになる．我々の方式では，サブ問題への分割と合成の枠組みでシーケンス図を導入する．

シーケンス図から状態マシンへの変換については，Whittle 等[5]や紫合[6]の研究があり，これらは状態の識別の仕方について提案している．本研究では，シーケンス図中に状態名を明記することによって，この変換を実現している．我々はシーケンス図を単なる信号のやり取りの規定ではなく，

Vol.2013-SE-180 No.3
Vol.2013-EMB-29 No.3
2013/5/27

信号のやり取りの意味も理解しながら規定するので，状態名の明記は妥当であると考える．

3. 分析の手順

プロブレムフレームのサブ問題の規定にシーケンス図を使った分析の手順の概要を図 1 に示す．まず，問題全体の概要をプロブレム図で規定する．ただし，この段階ではまだ共有現象等の詳細な規定は省略する．次に，問題をいくつかのサブ問題に分割する．この時，分割したサブ問題の集合によって，問題をできるだけ漏れなくカバーすることを確かめる．次に，サブ問題ごとに，その振舞いの例示として，シーケンス図を記述する．シーケンス図の記述によって，もとの問題のドメインに関係する共有現象(イベントや状態等)が徐々に明らかになってくる．全てのサブ問題に対してシーケンス図を記述した後で，今度はそれらをまとめてもとの問題図に合成する．この合成によって，もとの問題図の各インターフェースに関わる共有現象（シーケンス図で規定されたイベント等）が明らかになる．また，もとの問題図の上で，各サブ問題の振舞いがどのようになるかをチェックする．この合成によって，サブ問題間の矛盾等が見出されることもあり，その場合は，サブ問題のシーケンス図を修正する．場合によってはサブ問題の分割自体を変更することも必要になる．シーケンス図の合成の一環として，システム全体の状態遷移仕様を作成する．この状態遷移仕様は，すべてのシーケンス図に適合したものである．状態遷移仕様を見ることによって，これまでサブ問題個別に理解していた振る舞いが，システム全体の振舞いとして理解できるようになる．

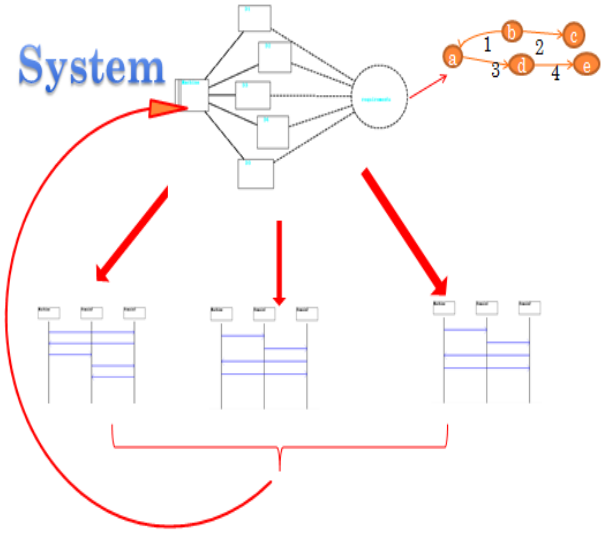

図 1　分析の手順

以上の手順を支援するツールとして，プロブレムフレームの方式にシーケンス図を組み込んだ支援システムを作成した．ツールは，まず問題全体の概要をメイン問題図で規定し，それをもとにサブ問題への分解を行い，サブ問題ごとに問題図（サブ問題図）とシーケンス図(サブ問題シーケンス図)を作成する．全てのサブ問題に対してシーケンス図を作成したのち，それらの結果をメイン問題図に合成し，さらにマシンの仕様として，状態遷移記述を生成する．ここで，シーケンス図で規定したイベントが問題図のインターフェースに反映され，最終的に，マシンの状態遷移仕様での入力イベントと出力イベントとして整理される．次節で，支援システムの詳細な機能について述べる．

4. 支援システム

4.1 メイン問題図の作成

問題の概要を表現するために，まずメイン問題図を作成する．問題図は，プロブレムフレームの記法に従って，問題の解となる「マシン」，問題の環境に存在する「ドメイン」，ドメインに対する要請を規定する「要求」の３種類の要素と，それらの間の関連からなる．マシンは解決策であり，開発対象のコンピュータやソフトウェアになる．ドメインは制御される装置や人間など，問題世界に存在する実体を示す．要求はドメインに対する要請(ドメインの振舞いの規定等)を示す．要素間の関連は，要求とドメイン間の要求参照と，マシンとドメイン間のインターフェースからなる．ここでは要求とマシン間には直接の関連はないものとする．つまり，要求は問題(ドメイン)に対する規定であり，解決策(マシン)に対しては無関心であるものとする．マシンとドメインの間のインターフェースによって，マシンはドメインから情報を受けてドメインを操作することを規定する．一般にプロブレムフレームはインターフェースと要求参照に共有現象を規定するが，本システムでは初めにメイン問題図を描いた時点ではまだ規定せず，サブ問題に分解してから徐々に共有現象を明らかにしていくものとする．図２はツールの入力画面を示す．左上にあるマシン，ドメイン，要求のボタンを押すことによって，下のパネルにそれぞれの要素を作成できる．作成されたマシンやドメインのサイズと位置はマウスで調整する．LINE をセレクトして，マシンとドメインをクリックすれば，インターフェースをつなげることが出来る．ラインが被らないように，中間点をつけ，その点を移動することでレイアウトを調整できる．要求参照(点線)は dotted line をセレクトしてドメインと要求を結ぶ．ドメインや関連線を削除することができるが，ドメインを削除すれば，関連の線は自動的に削除される．要素と関係に名前と内容をつけることができる．ただし，本手法では，メイン問題図作成の時点ではまだドメインやインターエースの詳細（共有現象等）は規定しないため，内容はここでは規定せず，サブ問題のシーケンス図登録時に規定する．作成されたメイン問題図はファイルに保存，読出しをすることができる．

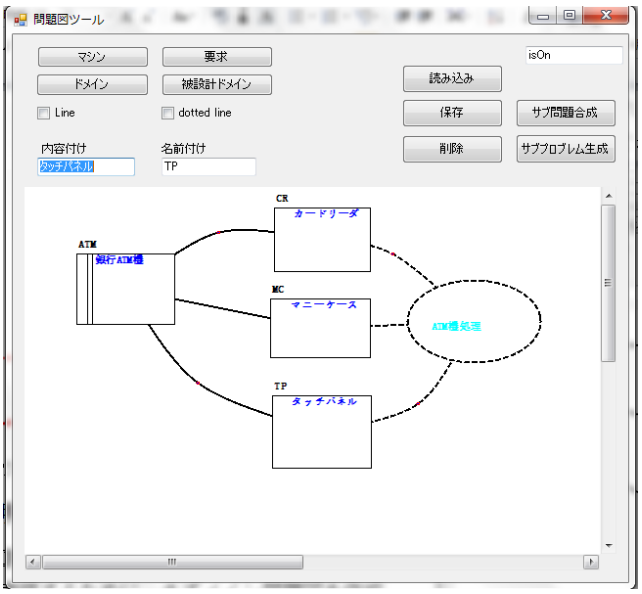

図 2　メイン問題図作成

4.2 サブ問題の分解

メイン問題の作成が完了した後，サブ問題に分解する作業に入る．図 2 のメイン問題図作成画面の右上のサブプロブレム生成ボタンを押すと，メイン問題図をベースにしたサブ問題分割画面（図 3）に遷移する．図でサブ問題名をサブプロブレム分解のカラムに入力する．それぞれのサブ問題に使われるドメインとマシンをチェックボックスで選ぶ（マシンは必ず一つ目に表示される）．図 3 で表の 1 行が 1 つのサブ問題に対応する．各行のシーケンス図生成ボタンを押すとサブ問題のシーケンス図作成画面に，サブプロブレム生成ボタンを押すとサブ問題図の生成画面に入る．

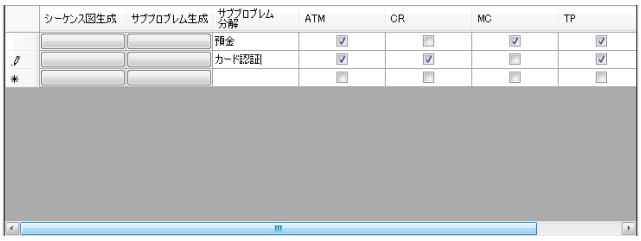

図 3　サブ問題分解

4.3 サブ問題シーケンス図

図 4 に図 3 の「預金」サブ問題のシーケンス図生成ボタンを押して生成されるシーケンス図の初期画面を示す．サブ問題分割で選択されたドメインがシーケンス図のオブジェクトとして上方に生成される．なお，オブジェクトの順番はマウス操作で変更でき，その場合，イベントのやり取りを示す矢印線はオブジェクトとともに移動する．なお，本システムではシーケンス図の生命線（縦線）は単なる線とし，UML のシーケンス図の活性区間までは規定しない．シーケンス図の初期画面に対して，サブ問題を分析・検討し，ドメインの適切な振舞いを明らかにし，その結果をマシンとドメイン間のイベントのやり取りの例示としてのシーケンス図で規定する．さらに，マシン自体の状況を検討し，マシン状態変化を考える．図 5 に示すように，シーケンス図のメッセージ線をクリックすることでイベント名を規定する（右の Create Event のテキストボックスに名前を入力する）．

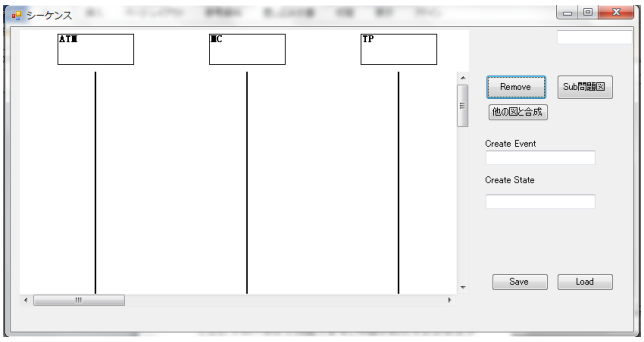

図 4　サブ問題シーケンス図

通常のシーケンス図では状態まで書かないが，マシンの仕様として状態遷移記述を生成するために，本ツールではマシンの状態を規定することが出来る．状態はマシンの生命線のメッセージのやり取りの間をクリックして状態名を入力することで，その位置の状態を規定できる．入力した状態は入力イベント間の時間帯と関係する．つまり，指定した場所の直前の入力イベントの後から直後の入力イベントの前まではこの状態になる．これによって，マシンの状態遷移では安定状態（入力待ちの状態）のみ扱うことになる．イベントの出力中は特に状態として扱わない（UML や SDL の状態遷移と同様）．イベントの上にイベント名を表示され，状態名は生命線に着く．

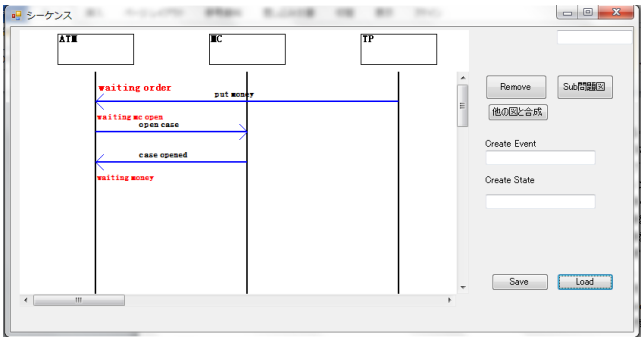

図 5　シーケンス図イベントと状態入力

イベント線の移動や削除もマウス操作で簡単にできるが，この場合，マシンの状態は直前の入力イベントとともに移動，削除される．

シーケンス図の作成で，同じイベントのやり取りがいくつかのシーケンス図に出てくる場合，規定済のシーケンス図を別のシーケンス図に挿入する機能によって効率よくシーケンス図を書くことができる．シーケンス図の挿入では，

「他の図と合成」ボタンを押し，挿入すべきシーケンス図データを指示して，挿入すべき場所を生命線上で指定すれば，その位置にシーケンス図が挿入される．

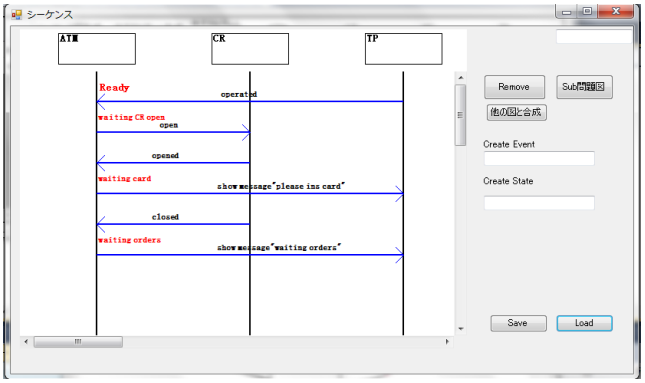

図 6　シーケンス図の合成例 1

図 7 は，図 5 のシーケンス図の 1 番目の入力イベントの直後に図 6 のシーケンス図を挿入した例である．二つのシーケンス図のオブジェクトが一致すれば，イベントは自動的に合わせる．異なるオブジェクトは後ろに追加される．図 6 のオブジェクトにまるをつけたところは合成する前になかったオブジェクトを追加したもので，四角に囲まれた部分は図 5 に図 6 のシーケンス図を挿入した部分を示す．

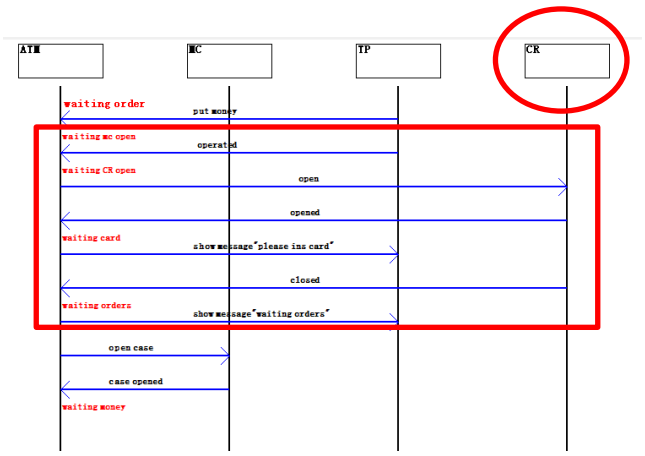

図 7　シーケンス図の合成例 2

4.4　サブ問題図

図 5 のシーケンス図作成画面の右上のサブ問題図ボタンを押せば，シーケンス図の情報をもとにしたサブ問題図が生成される．このとき，シーケンス図中のイベントは，サブ問題図の対応するインターフェースの共有現象に反映される．つまり，シーケンス図のドメインからマシン（マシンからドメイン）へのイベントによって，サブ問題図ではそのドメインとマシンを結ぶインターフェースの入力(出力)共有現象が生成される．サブ問題図というのは元の問題の部分的な問題で，元の問題のドメインの一部を使って詳細を規定する．図 8 に，図 5 に対して生成されたサブ問題図を示す．

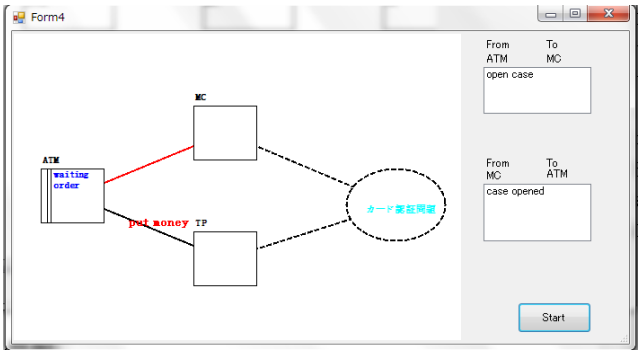

図 8　生成されたサブ問題図

4.5　メイン問題図の合成

いくつのサブ問題に対してシーケンス図を規定した後，シーケンス図中のイベントや状態をまとめて，元のメイン問題図に合成することができる．図 9 で示すように，メイン問題図で右下のサブ問題合成ボタンを押してサブ問題を指定すれば，そのサブ問題がメイン問題図に追加合成される．これを繰り返すことによって，すべてのサブ問題を合成することができる．合成したあと，インターフェースの線をクリックすると，画面右にそのインターフェースの共有現象として，シーケンス図のイベントを入力と出力に分けてまとめて表示する．また，右下のサブ問題一覧にあるサブ問題をダブルクリックすれば，そのサブ問題のシーケンス図によるアニメーションが実行される．アニメーションでは，ドメインとマシン間でイベントが行き来する様子と，その時のマシンの状態を動画で表現する．これによって，サブ問題の振舞いの視覚的な理解を促す．図 9 は，図 5 のシーケンス図のアニメーション中で，ATM から MC に open case イベントが出力された時点の図を示す．

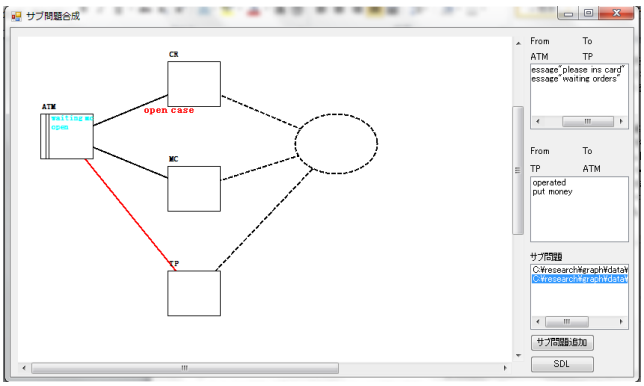

図 9　メイン問題図合成とアニメーション

4.6　状態遷移記述の生成

図 9 のメイン問題図合成画面の右下の SDL ボタンを押すことによって，合成したシーケンス図からの状態遷移記述を生成することができる．

状態遷移の生成は以下の通りである．まず各シーケンス図から，マシンの生命線に沿って入力イベント毎に，遷移

データ（事前状態名，入力イベント名，次の入力イベントまで（または終わりまで）の出力イベント名のリスト，事後状態名）を作成し，すべてのシーケンス図の全ての遷移データを取り出す．次に，遷移データを，第 1 キーを事前状態名，第 2 キーを入力イベント名でソートし，同じ遷移データは消去し重複を除いておく．ソートされた遷移データで，同じ事前状態名をまとめることによって，SDL に準じた状態遷移記述を生成する．

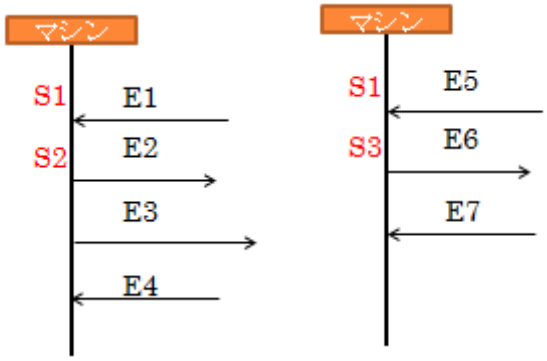

図 10　状態遷移記述合成

例えば, 図 10 のシーケンス図から以下の状態遷移記述が得られる．

```
S1
  Input    E1
    Output    E2
    Output    E3
    NextState    S2
  Input    E5
    Output E6
    NextState S3
```

図 11 にこのようにして生成された状態遷移記述の例を示す．ここでは，状態遷移記述は SDL に準じた形式としているが，これを C 言語形式にすれば，問題の振舞いのシミュレーションをすることが可能になる．

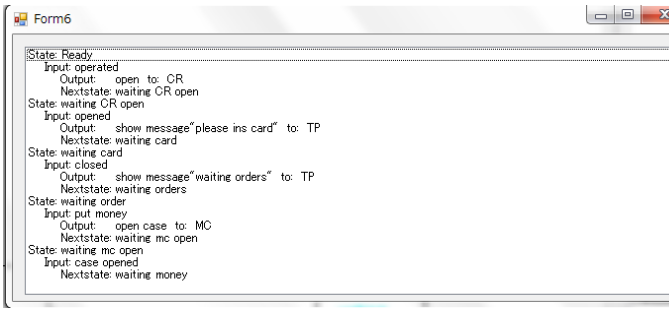

図 11　状態遷移記述実例

5. 適用例

5.1 電話交換機の例

前節で述べたシステムの評価のために簡単な電話交換機の例を適用した．図 12 にメイン問題図を示す．

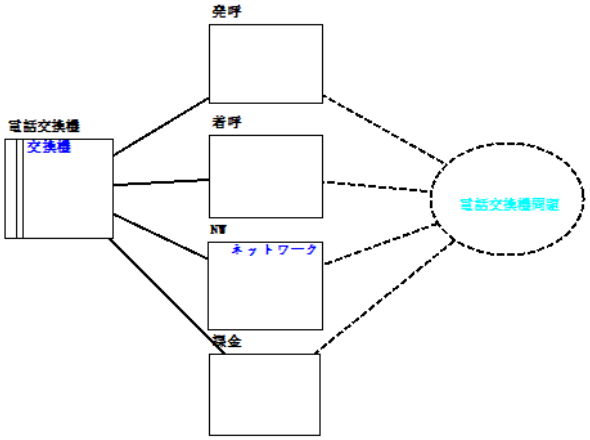

図 12　電話交換機問題図

電話交換機を分析して，自分のダイアル状態，相手に電話を掛けた時の相手の状態等によっていくつの例を検討し，以下のようなサブ問題を作成した．

a. ダイアル成功，相手は電話に出て通話を開始した
b. ダイアル失敗，電話番号を間違えた
c. ダイアル成功，相手が電話にでないので自分で切る
d. ダイアル成功，しかし相手は通話中
e. 通話の終了，自分から切る
f. 通話の終了，相手から切る

このサブ問題分割に対して，図 13 に示すように，必要なドメインを選択した．

サブプロブレム分解	交換機	発呼	着呼	ネットワーク	課金
通話成功	☑	☑	☑	☑	☑
番号間違い	☑	☑	☐	☐	☐
相手が出ない	☑	☑	☑	☑	☐
相手が通話中	☑	☑	☑	☑	☐
着呼終了	☑	☑	☑	☑	☑
発呼終了	☑	☑	☑	☑	☑

図 13 電話交換機問題分解

サブ問題 a では，図 14 のシーケンス図が示すように，交換機の初期状態はアイドルで，発呼側がオフフック(電話を上げる)すれば交換機の状態がダイアル中に遷移する．その状態で，ダイアル「入力中」のイベント（ダイアルイベントで完了していない場合）が来ても状態は変わらず，ダイアル「入力完了」イベントが来るとネットワークを予約し，着呼側を呼出して呼び出しチェック状態に遷移する．呼出

チェック状態で呼び出しが OK であれば，着呼側が電話に出るのを待つ（呼び出し中状態）．そこで相手が電話に出れば（着呼オフフックイベント発生）交換機は回線を接続し課金を開始し，発呼と着呼の両方に通話開始を指示して通話中状態に遷移する．ここではすべてのドメインが使われる．

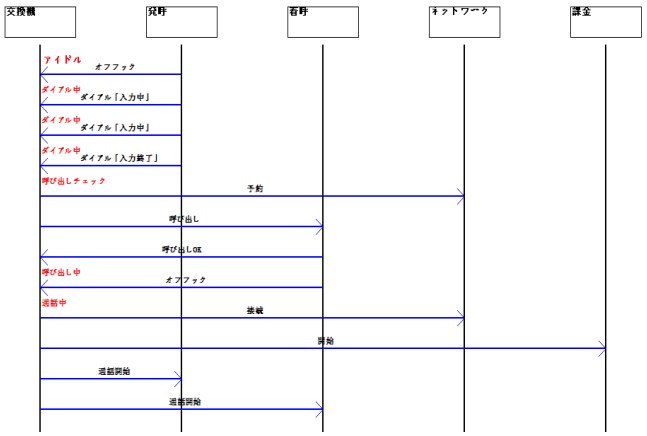

図 14 サブ問題 a のシーケンス図

サブ問題 b では，番号を間違えたので，交換機と発呼しか関連しないため，ドメインはこの二つにする．この場合のシーケンス図を図 15 に示す. 交換機はダイアル中の状態で，ダイアル「誤り番号」イベントを受けたら，発呼側に番号間違いメッセージを送信し，発呼終了まち状態に遷移し，発呼の終了（オンフック）イベントを待つ．発呼側がオンフックすると，アイドル状態にする．

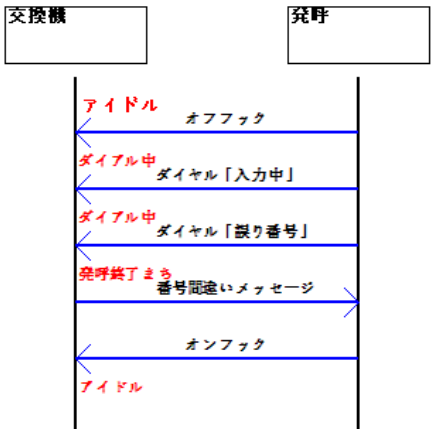

図 15 サブ問題 b のシーケンス図

サブ問題 c では，図 16 に示すように，相手が電話に出ないため通話は始まらず，課金ドメインは関係がない．この場合のシーケンス図は，アイドル状態から呼び出し中状態まではサブ問題 a と同じであるが，相手が出ないため，発呼側が呼び出し中にオンフックするケースになる. このとき，電話交換機はネットワークの予約を取止め，着呼側に呼び出し停止を指示して，アイドル状態にもどる．

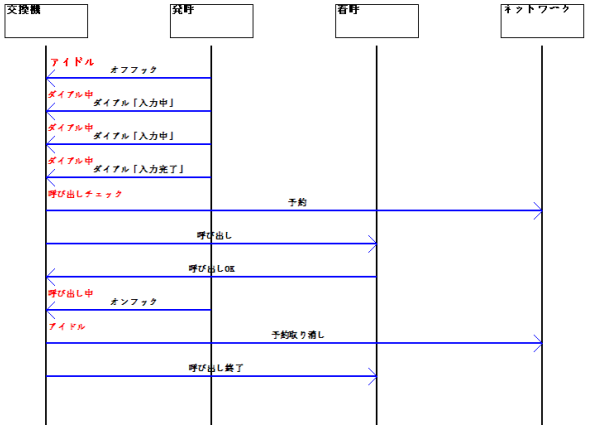

図 16 サブ問題 c のシーケンス図

サブ問題 e は, 図 17 が示すように通話中の状態から始まり，発呼側がオンフックすると電話交換機は課金を止めネットワークを切断し着呼終了まちの状態に遷移する．そして着呼側のオンフックが確認されればアイドル状態にもどる．

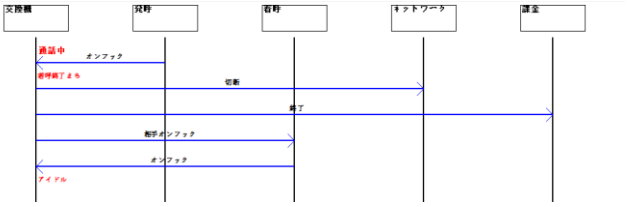

図 17 サブ問題 e のシーケンス図

サブ問題 d と f はそれぞれサブ問題 c と e に類似であるため，ここでは省略する．

以上のシーケンス図を合成することにより，以下のような状態遷移記述を得られた．

State: アイドル
　Input: オフフック　from: 発呼
　　Nextstate: ダイアル中
State: ダイアル中
　Input: ダイアル[入力中]　from: 発呼
　　Nextstate: ダイアル中
　Input: ダイアル[入力完了]　from: 発呼
　　Output:　予約　to:　ネットワーク
　　Output:　呼び出し　to:　着呼
　　Nextstate: 呼び出しチェック
　Input: ダイアル「誤り番号」　from: 発呼
　　Output:　番号間違いメッセージ　to:　発呼
　　Nextstate: 発呼終了まち
State: 呼び出しチェック
　Input: 呼び出し OK　from: 着呼

Nextstate: 呼び出し中
Input: 通話中 from: 着呼
Output: 予約取り消し to: ネットワーク
Output: 相手が通話中 to: 発呼
Nextstate: 発呼終了まち
State: 呼び出し中
Input: オンフック from: 発呼
Output: 予約取り消し to: ネットワーク
Output: 呼び出し終了 to: 着呼
Nextstate: アイドル
Input: オフフック from: 着呼
Output: 接続 to: ネットワーク
Output: 開始 to: 課金
Output: 通話開始 to: 発呼
Output: 通話開始 to: 着呼
Nextstate: 通話中
State: 通話中
Input: オンフック from: 発呼
Output: 切断 to: ネットワーク
Output: 終了 to: 課金
Output: 相手オンフック to: 着呼
Nextstate: 着呼終了まち
Input: オンフック from: 着呼
Output: 切断 to: ネットワーク
Output: 終了 to: 課金
Output: 相手オンフック to: 発呼
Nextstate: 発呼終了まち
State: 着呼終了まち
Input: オンフック from: 着呼
Nextstate: アイドル
State: 発呼終了まち
Input: オンフック from: 発呼
Nextstate: アイドル

この状態遷移記述はまだ不完全である．例えば，ダイアル中に発呼がオンフックを出して中断する場合の遷移が規定されていない．本システムには取り入れていないが，発呼電話の状態遷移(ドメインプロパティ)規定によって，オンフック後の電話はいつでもオフフックされる可能性があることが分かれば，このような遷移規定漏れは自動的にチェックできる[2].

5.2 評価

システムの適用によって，問題分析作業の流れが標準化され，分かりやすくなった．また，サブ問題の分析で，その振舞いをシーケンス図で表現することにより，サブ問題を考える基準やガイドになった．そして，すべてのサブ問題のシーケンス図から，メイン問題図を合成し，さらに状態遷移記述を生成することによって，小さなサブ問題ごとに例を考えることができ，最初から全体の状態遷移を考えるより理解しやすくなった．

このシステムの適用によって,初めに例をたくさん調べ,その後それらをあつめて一般則を導き出すという方式が初期の問題分析にとって適切であることが確かめられた．ただ，電話交換機の例では，矛盾な状態やイベントまでは検出されなかったので，今後，より複雑な問題に対する適用によってシステムの有用性を評価していく必要がある．

6. おわりに

プロブレムフレームにシーケンス図を活用して，問題の初期分析の段階の手法として,サブ問題ごとに例を規定し,それらをまとめて分析し，一般則としての状態遷移仕様を見出す方法を述べ，それを支援するツールについて説明した．また，簡単な電話交換機の分析を例にして，支援システムを使ってプロブレムフレームの問題図,サブ問題分割,サブ問題ごとのシーケンス図の規定，シーケンス図をまとめた状態遷移記述の生成に至るまで分析を進め，本手法と支援ツールの評価を行った．結果的には，サブ問題ごとに動作例をシーケンス図で規定することにより，元のプロブレム図の各ドメインとマシンの関係が明らかになり，問題全体の把握がしやすくなった．また，状態遷移仕様の自動生成によって,問題全体としての振舞いが明らかになった．

この適用評価で，状態遷移仕様のまとめた段階で遷移の漏れ(ダイアル中の発呼オンフックによる終了の遷移)を発見した．一般に，問題をサブ問題に分割して規定する方法では，分割したサブ問題が必要なすべての振舞いを規定していることを確認することはできない．従って，今回の適用例のような漏れが発生する．5.1 でも書いたが，今後の課題として，サブ問題の合成によってドメインの状態遷移規定も生成，チェックし，それによってこのような漏れを自動的にチェックする機能の導入を検討していきたい．

今回は，簡単な電話交換機での評価にとどまったが，今後は,より複雑な問題の分析に本システムを適用評価して,その有効性を確かめることが必要である．一方で問題分析に慣れた技術者だけでなく，不慣れな技術者も含めて，プロブレムフレームとシーケンス図を使った手法やツールが有効かどうかを調べていく必要がある．

Vol.2013-SE-180 No.3
Vol.2013-EMB-29 No.3
2013/5/27

参考文献

1) マイケル・ジェクソン,”プロブレムフレームソフトウェア会派問題の分析と構造化”,株式会社　翔泳社.2006 年 5 月.
2) 紫合治,横山薫: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,情報処理学会論文誌,Vol.53, No.2, 2012.
3) Colombo,P., et.al.:Towards a Meta-model for problem frames:conceptual　issues and tool building support, Fourth International Conference on Software ngineering Advances, 2009.
4) Vieri del Bianco1,V.and Lavazza,L.:Enhancing problem frames with scenarios and histories in UML-based software development, Expert Systems,　Vol.25,No.1,2008,
5) Whittle.J.and Schumann,J.:Generating statechart designs from scenarios, 22th International Conference on Software Engineering, 2000.
6) 紫合治:環境の仕様を用いたシナリオから状態マシンの生成, ソフトウェア工学の基礎ワークショップ 2004(FOSE04), 2004.