ソフトウェア工学 90－4
（１９９３．２．４）

設計履歴とペトリネットを用いたソフトウェア変更支援

村瀬康人　小野康一　新井浩志　深澤良彰　門倉敏夫

早稲田大学理工学部

ソフトウェアの再利用や保守を容易化するために、ソフトウェアの変更支援に対する要求が高まっている。我々は、ソフトウェア設計時の判断過程を設計履歴として蓄積し、それを用いる事により変更支援を行う手法について研究している。設計履歴の利用方法としては、設計履歴とドキュメントを関係付ける事により、ドキュメント間の対応をとったり、設計履歴を辿る事により、設計判断の変更による波及解析や、設計候補案の代替案利用可能性などの判断などを行う。また、本システムでは、これらの解析を行う際にペトリネットを用いる。そして、このペトリネットの発火系列を調べる事により、ドキュメントの対応付けや、波及解析などを行う。本稿では、設計履歴のペトリネットによるモデル化の方法、及びそれを用いた解析手順について述べる。

Software Modification Support with Design Histories and Petri-Net

Yasuhito MURASE, Kouichi ONO, Hiroshi ARAI, Yoshiaki FUKAZAWA, Toshio KADOKURA

School of Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169, Japan

Software modification support is requested to facilitate software reuse and maintenance. We are studying a method to support software modification with design histories. Design histories are used in order to analyze influences by the correspondence between documents and design histories. And this system uses Petri-net in analyzing. It analyzes influences and corresponds each document by monitoring firing series of a Petri-net. In this paper, we describe the modeling and analyzing method of design histories with Petri-net.

１．はじめに

ソフトウェアの再利用や保守を容易化するために、ソフトウェア変更支援に対する要求が高まっている。ソフトウェア変更作業の中で、ソフトウェアを理解するために費やされている時間と労力は大きく、ソフトウェア理解を容易化する事は、ソフトウェアの変更を容易化する事につながる。

ソフトウェアを理解する為には、次の２つの事を理解する必要がある。一つは、そのソフトウェアは何を実現したものであるかという仕様に対する理解であり、もう一つは、なぜその様に実現されたかという設計に対する理解である。例えば、ハッシュ探索を行うモジュールがあったとすると、そのモジュールが探索を行うモジュールであると理解する事が仕様に対する理解であり、二分探索や線形探索等の他の探索アルゴリズムを用いず、ハッシュ探索を用いた理由を理解する事が設計に対する理解である。

仕様に対する理解を行う時に有効な手段として、ドキュメントの参照が挙げられる[1]。ここでドキュメントとは、要求仕様書や機能仕様書をはじめ、ソースコードやメモ書き等を含めた、ソフトウェア開発時に作成される書類を表す。変更担当者は、これらのドキュメントを状況に応じて使い分け、またそれぞれの対応をとりながら参照していると考えられる。

設計に対する理解を行う時に有効な手段として、近年、設計履歴の蓄積とその利用が注目を浴びている[2][3]。設計履歴とは、なぜその様に設計したかという設計の判断の根拠やその際に参照したドキュメントの箇所や知識、他の候補案などを集めたものである。これを用いる事により、変更担当者は設計時の設計者の意図を正確に、また容易に理解する事ができる。

この様に、ソフトウェアの変更時のドキュメント参照や、設計履歴の利用は有効な手段である。しかし、現状では以下の様な問題がある。

（１）ドキュメントはそれぞれ個別に保存されており、ドキュメント間の対応付けがされていない。その為、変更担当者はあるドキュメントに対応する他のドキュメントの箇所を参照する際に、自分でその対応をとらなければならない。例えば、機能仕様書の機能に対応するソースコードを参照する場合、その対応を文書の内容を把握してどの部分が対応するかという判断をしながら、調べなければならない。

（２）設計履歴は文書情報として残されているが、それらの間のトレースや関係把握は変更担当者に任されている。例えば、ある設計の決定が他の部分の決定に影響を与えている場合、変更担当者は文書からそれを読取り、ある判断の変更が他のどの判断に影響を与えるかという解析を行わなければならない。また単なる文書情報として残されている設計履歴は、変更担当者にとって理解しやすいものであるとは言えない。

この様に、現状のドキュメントの参照や設計履歴の利用には、単純で機械的な作業が多く含まれている。

我々は、ソフトウェア変更時におけるドキュメント間の対応付けとその参照、及び設計の判断間における関係把握を容易化するシステムを考案した。これにより、変更担当者はドキュメントの検索や、設計履歴間の依存関係確認のための設計トレースのような単純で機械的な作業から開放され、ソフトウェア及び変更理解や変更方針決定に専念する事ができる。

２．システム概要

本システムの概要を、全体の概観、利用の手順の点から説明する。

２．１．全体概観

本システムでは、ドキュメント参照時に起こる、ドキュメント間の対応付けの容易化をはかるために、設計履歴を中心にそれぞれのドキュメントを対応付ける。すなわち、ドキュメントを設計作業の生産物と位置付け、設計履歴とそれぞれのドキュメントを結びつける。これにより、例えば要求仕様書のある部分に記述されている要求が、機能仕様書のどの部分に記述されているかという様な対応をとるには、要求仕様書から順に設計履歴を辿り、辿り着いた機能仕様書の部分が、その要求仕様書の部分に対応する機能の記述であると言える。

設計履歴の理解性、利用性の向上、及びシステムを自動化する手段として、設計履歴をペトリネットによりモデル化する。ペトリネット[4]とはプレース、トランジション、アークのネットによりシステムをモデル化し、トークンの数により状態を表現する事ができるモデル化の手法である。このペトリネットを用いる事により、単なる文書情報であった設

計履歴を動的な情報として扱え、また計算機上で容易に扱う事ができる。また、図によりシステムを表現する為、理解性も向上する。

２．２．利用手順

図１に本システムの利用手順を示す。

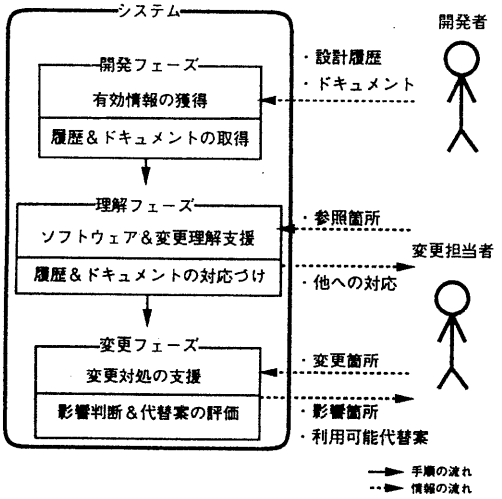

図１：利用手順

本システムは大きくわけて、開発フェーズ、理解フェーズ、変更フェーズの３つのフェーズに分けることができる。

開発フェーズは、開発者がソフトウェアの依頼を受けてそれを作成するフェーズである。本研究ではソフトウェアの設計を、問題の提起－問題の解決作業の連鎖と捉えモデル化し、そのモデルにしたがって設計履歴とその生産物であるドキュメントを記録する。例えば、開発者は”出力装置として何を用いるか？”を問題として入力し、”ＣＲＴを用いる”や、”ＣＲＴとプリンタを使い分ける”等を解決策として入力する事になる。

理解フェーズは、変更担当者が変更依頼を受けて、現行のソフトウェアをドキュメントを用いて理解し、変更の方針を決定するフェーズである。変更担当者は、変更を行うためにソフトウェア自体と変更箇所の理解を行う必要がある。その際に、各種ドキュメントを相互に対応をとりながら参照を行う。例えば、操作に対する変更であれば、その操作を記述した外部仕様を参照し、またその部分に対応した機能が記述された機能仕様書やその部分の詳細な設計が記述された内部仕様書を参照する。そこで本システムでは、最初に変更担当者にあるドキュメントで変更対応箇所を入力してもらい、次に設計情報を用いてその他のドキュメントの対応箇所を調べ、変更担当者に出力する。例えば要求仕様書中の”検索結果を出力する”という箇所を変更対応箇所として入力した場合、それに対応する機能である”ＣＲＴの表示エリアに赤色で表示する”という機能仕様書の部分や、ソースコード中の画面出力の手続などが、変更担当者に提示される。

変更フェーズは、理解フェーズで絞り込まれた変更対応箇所を実際に用いて、変更箇所の指摘を行い、その変更の波及解析を行うフェーズである。本システムは、上位の設計判断から下位の設計判断へと順に設計トレースを変更者に提示し、変更者にその判断を変更するかどうかを質問形式で尋ね、変更を行うべき判断を探し、その箇所を入力してもらう。例えば、”ＣＲＴを用いて出力する”という解決策を用いるかどうかを変更担当者に尋ね、その返答を入力してもらう。そして、変更を行うべき判断を全て洗い出したら、その変更による波及解析を行い、解析結果、及びその変更の波及により変更が必要となったドキュメントの箇所を変更担当者に提示する。また、代替案の利用可能性を自動的に調べ変更担当者に提示する。例えば、”ＣＲＴとプリンタを用いる”という候補案が利用できなくなった場合、その代替案である”ＣＲＴのみを用いる”という案が用いられるかどうかを調べ、その結果を提示する。

３．システム構成

本システムの構成を図２に示す。本システムは大きく分けてユーザインターフェース部、設計履歴取得部、解析部から構成される。

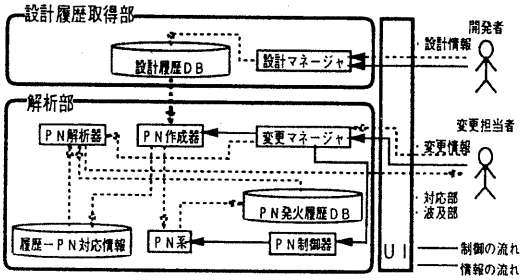

図２：システム構成

ユーザインタフェース部はユーザからの設計情報の入力や、解析結果の提示などを行う。

設計履歴取得部は本システムの定める設計モデルに従って設計情報を入力、蓄積する部分であり、開発フェーズで用いられる。設計マネージャは開発者に対して、設計情報の要求をする部分であり、履歴データベースはそれを蓄積する部分である。

解析部は設計履歴を用いてドキュメント間の対応をとったり、変更の影響範囲を解析する部分であり、理解フェーズと変更フェーズを支援する。ペトリネット作成器は、設計履歴を用いてそれに対応するペトリネットを作成する。ペトリネット系は与えられたペトリネットを規則に従って発火させる。ペトリネット制御器は、ペトリネットのトランジションの発火の管理を行う。ペトリネット発火履歴データベースは、発火したペトリネットの発火系列を記憶する。ペトリネット解析器は、ペトリネット発火履歴を用いてドキュメントの対応箇所、変更の影響解析を行う。

4．設計履歴

本システムの設計履歴を設計モデル、その入力手段である設計カード、及びリンクについて説明する。

4．1．設計モデル

設計過程は、問題提起、問題の解決作業の連鎖と捉える事ができる。問題提起とは、設計を進めていく上で考えなければならない事を挙げる事であり、問題の解決作業とは問題を解決するための手法の発見を行う事である。

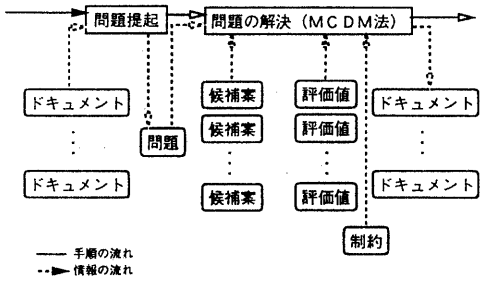

図3：設計モデル

問題提起段階は、入力としてドキュメントをとり、出力として問題が提起される。すなわち、ドキュメントを参照し、次に何を考えなければならないかを考えた結果が問題の提起となる。例えば、”データベース中で用いる探索アルゴリズムにはどの様なものが適しているか？”とか、”出力装置には何を用いるか”等が問題になる。

問題提起段階では、開発者は問題とその問題の原因となるドキュメントとその箇所を入力する。これにより、問題とその原因箇所の対応をとる事ができる。

次に問題提起段階で出力された問題を解決するわけであるが、本システムでは問題解決の手段としてMCDM（Multiple Criteria Decision Method）法[6]を適用する。

MCDM法は複数の代替案に対して、複数の属性がある場合の代替案選択法である。例えば、交通手段として何を用いるかという問題に対して、”飛行機”、”電車”、”バス”という、候補案がある場合を考える（図3）。MCDM法では、まず選択の際に考慮に入れなければならない属性を洗い出す。ここでは、”費用”、”時間”、”快適さ”がそれにあたる。また同時に、それぞれを評価するための評価基準を設定し、評価値を付ける。例えば費用に関するものであれば、”安い”、”高い”等が、評価基準になり、それらを良い順に3点、2点の様に評価値を付ける。この時、それぞれの属性の重要さが全て同じであるという事は稀である。そこで、それぞれの属性に重みをつける。最後に、それぞれの候補案のそれぞれの属性に対して評価を行い、効用値をそれぞれの重みと評価値の積を足したものとして求める。例の場合は、バスが効用値17であり、最も有用な候補案であるといえる。

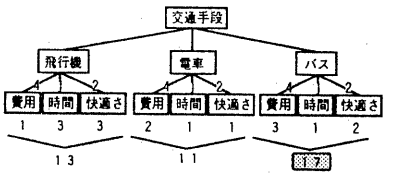

図4：MCDM法

以上のMCDM法を本システムの設計モデルにあてはめると次のようになる。

まず問題解決の候補案を列挙する。次に候補案の選択の際に考慮に入れなければならない属性を洗い出す。次にそれぞれの属性の評価基準を設定し、評価を行う。次に候補案作成及び選択に際して、参考

にしたドキュメントや設計生産物の参照関係、及びその候補案選択に際しての制約を入力する。そして、それぞれの属性に重みを付け、効用値を計算し、選択可能な候補案の中で最も効用値の大きい候補案を選択する。

本システムではＭＣＤＭ法を候補案選択法としてよりも、もしろ、候補案選択時の設計者の意図を記録する手段として用いる。すなわち、設計者がその候補案を選択したのは、その候補案をこの様に評価したからだという事を記録する手段として用いる。

４．２．設計履歴カード

以上の設計モデルを入力するために、本システムでは、図５の様な設計履歴カードを用いる。

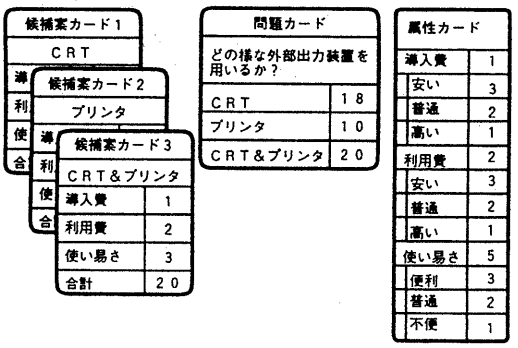

図５：設計履歴カード

設計履歴カードは次の３つのカードで構成されている。

○問題カード　○候補案カード　○属性カード

問題カードは、問題とその解決候補案、そしてそれらに対する評価値から構成されている。候補案カードは、それぞれの候補案に対して一枚づつ作成され、候補案とそれぞれの属性に対する評価値が入力される。属性カードには、候補案選択の際に考慮に入れなければならない属性とその重み、またそれぞれの評価基準を記述する。

ここでいう、候補案、属性、評価値は、ＭＣＤＭ法で用いられる、候補案、属性、評価値にそれぞれ対応している。

４．３．リンク

それぞれの設計履歴は単独に存在するわけではない。それぞれの履歴が相互に関係しあい、制約しながら存在する。そこで、それらを表現するために、本システムでは次の３つのリンクを用意した。

○親子リンク　○制約リンク　○参照リンク

親子リンクとは、問題の親子関係を表すためのリンクである。ここで親子関係とは、その問題の基となるべきドキュメントとその結果作成された問題の関係、及び、問題と解決策の関係をいう。その際に、ドキュメント全てが参照されるということは稀である。例えば、要求仕様書を参照したとしても、参照箇所はその一部であり、全てとは限らない。その為、本システムでは、参照箇所を設計者に始点と終点を指定してもらう事により入力する。

制約リンクとは、ある設計履歴の属性が他の決定に影響を与えている場合にその候補案とその属性の間を結ぶためのリンクである。本システムにおいて制約は、属性に対する評価値を用いて行う。例えば、他の設計履歴に、”利用可能なプリンタ台数”という属性があった場合、他の設計履歴の候補案でプリンタを利用する時、この属性がその候補案に制約を与える事になる。その場合、利用可能な台数が評価基準に設けられているはずであるから、その評価基準を用いて、制約を与える。また、制約を与えたいが、それに対する属性がない場合には、その制約を与える設計履歴にその属性を追加する事により対処する。

参照リンクは、解析には影響を与えないが、参照した方が理解を容易にする様なドキュメントに対してつなぐリンクである。

４．４．設計履歴の例

図６に設計履歴の例を示す。例においては、”どの様な外部出力装置を用いるか？”という問題が、入力となるドキュメントから出ている。そしてその問題に対して、候補案”ＣＲＴに出力する”、”プリンタに出力する”、”ＣＲＴとプリンタを使い分ける”が挙げられている。そして、その候補案を選択するための属性として、”導入費”、”利用費”等が在り、それぞれの属性に対して、評価基準が設定される。そして、その評価基準を用いて、それぞれを評価し、それぞれの属性に重みを付け、その結

果として効用値が計算され、最大となるものを選択する。そして、選択された候補案に対して、ドキュメントが作成されている。

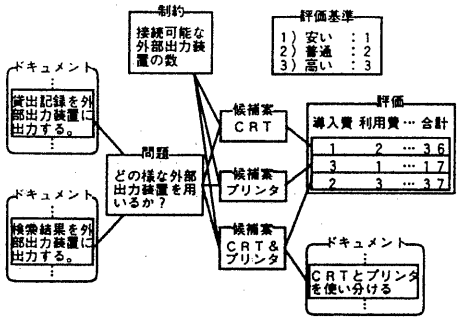

図６：設計履歴の例

５．ペトリネットによるモデル化

５．１．ペトリネットの説明

ペトリネットは、次の様な５つ組で表現すことができる。

N＝<P, T, F, W, M>
P：プレース集合　　T：トランジション集合
F：アーク集合　　W：アークの重み
M：トークンの配置

ペトリネットは、プレース、トランジション、それらを結ぶアーク、マーキングによって表される。ペトリネットの実行はトランジションを発火させる事であり、トランジションの発火はその入力プレースからトークンを取り去り、新しいトークンを生成し、出力プレースに分配する事である。また、トークンとはプレース内に存在し、トランジションの実行を制御するもので、このトークンをプレースに割り当てる事をマーキングと言う。

また、本システムではペトリネットの表現力をより豊かにするために、個体トークンをもったペトリネットを用いる[4]。これは、普通のペトリネットにおけるトークンがどれでも同じであるのに対して、各トークンが識別子を持ち、区別可能にしたものである。この個体トークンを持ったペトリネットは従来のペトリネットと同様に扱う事ができる事が知られている。

５．２．割当て

ペトリネットの各構成要素に、本システムの情報を次のように割り当てる。

P：ドキュメント（D）、属性（A）、制御用（G）
T：問題（B）、候補案（H）
F：親子アーク（O）、制約アーク（C）
W：評価値、制約値、その他は１
M：随時決定

プレースには参照の対象となる、ドキュメントと属性及び制御用を割り当てる。ただし、ここで制御用プレースとは、ある問題トランジションを発火可能にしたり発火禁止にしたりするプレースの事である。トランジションには決定行為の対象となる、問題と候補案を割り当てる。アークには評価や制約や参照関係など、相互の関係を表す情報を割り当てる。アークの重みには、評価値や制約値など、量的情報を割り当てる。トークンには、決定行為の流れや、物等を割り当てる。

５．３．表記法

本システムにおけるペトリネットの表記法を図７に示す。

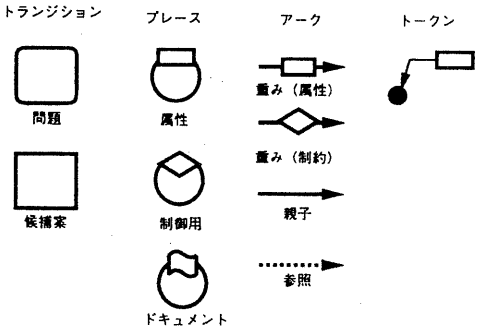

図７：表記法

トランジションはボックスで表し、その中に問題や候補案の名前を記述できる。プレースは丸で表し、その上に属性名などが記述できる。アークは矢印で表し、その線上にそのアークの重みや識別子を記述する。トークンはそれから矢印を出し注釈としてその属性を記述する。

５．４．ペトリネットの例

図８に図７をペトリネットを用いて表現した例を示す。

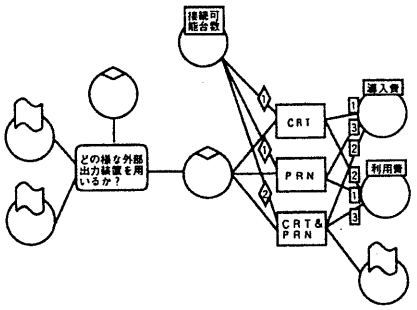

図８：ペトリネット例

６．発火制御

以下にペトリネットの発火制御の方法を説明する。

６．１．発火規則

ペトリネットを発火させる時に、複数の発火系列を考える事ができる場合がある。その様な発火を制御するために、発火順序に対する基準を設ける必要がある。そこで本システムでは次の４つの規則を設定した。

１）発火要求を最優先
２）設計時に用いられた発火系列が優先
３）代替案選択に際しては、効用値が大きいものを優先
４）設計時の時系列順

１）の発火要求を最優先という規則は、優先的に用いたい発火系列が存在する場合、変更担当者は、発火要求としてそれをシステムに要求し、システムはそれを優先的に発火させる規則である。

２）の設計時に用いられた発火系列が優先という規則は、実際の変更に際して、設計時に用いられた候補案よりも、より効用値の高い候補案が利用可能な状態になる時がある。例えば、制約に、利用可能なプリンタの台数というものがあり、変更前は２台であったが、３台まで使えるようになるというような、制約が弱くなるような変更がある。その様な場合、変更箇所をなるべく小範囲に押えるために、効用値が小さくても設計時に用いられた候補案を用いる。ただし、これは最後に、発火系列レポートの中で報告され、またそれにより発火系列を変更する事が可能である。

３）の代替案選択に際しては、効用値が大きい方を優先するというのは、設計時に用いられた候補案が、変更により用いる事ができなくなった場合に適用される。これは２）とは逆に制約がきつくなった場合、他の有効な代替案を用いる事により、変更の影響範囲を小さく押えると共に、変更担当者の代替案を考える作業を軽減するために用いられる。ただし、この規則は１）と２）の規則が適用されなかった時のみに用いられる。

４）の時系列順というのは、トークンの競合が起きた際に、設計時に先に決定された方に優先的にトークンを与える為に用いる。これに関しても発火後に作成されるレポートの中で報告され、発火系列を変更する事ができる。

６．２．発火手順

理解フェーズでは、あるドキュメントで指定された変更対応箇所を他のドキュメントに対応させることにより、ドキュメント参照を容易化する。

ドキュメントの対応箇所を辿るには親子リンクを親ドキュメント、子ドキュメントへと辿る事により、あるドキュメントで変更対応箇所と指定された箇所を、その原因となるドキュメントとその結果作成されたドキュメントに対応をとる事ができる。

そこで、まず変更対応箇所の親ドキュメントを辿ることにより、変更の原因となるドキュメントへの対応をとる。ＰＮ作成器において、設計履歴情報をもとに次の様なＰＮを作成する。

$N_1=<DUAUG, BUH, O_{rev}UCUE, W, M_1>$

O_{rev}は親子リンクの向きを反転させたものである。また、初期トークンの配置は、親子アークの出ていないドキュメント（以下最終ドキュメント）に対応するプレースと対応箇所と指定された箇所以外の制御用プレースである。このＰＮを発火させ、発火しなかったプレース、もしくは、親子アークが入っていないドキュメント（以下初期ドキュメント）の対応するプレースにトークンが必要数溜らなかった部分が、初期ドキュメントにおける対応箇所である。トークンの必要数とは、そのプレースに接続されている入力アークの重みの和である。

次に、その初期ドキュメントを基に作成したドキュメントを辿る。そのために、以下の様なＰＮを作

成する。

N_2=<DUAUG, BUH, OUCUE, W, M_2>

ここでM_2は、N_1を発火させた結果のトークンの内、正常に初期ドキュメントに辿りつくことができたプレースと、変更対応箇所と指定された箇所を除く制御用プレースである。

次に変更フェーズでは、理解フェーズで絞り込まれたドキュメントと設計履歴を用いて実際の変更箇所を特定し、それに対する波及解析を行なう。変更すべき判断は、ドキュメント作成フェーズで絞り込まれた履歴にあると考えられる。そこで、変更担当者にこの履歴を提示し、実際に変更すべき変更箇所を特定する。

次に、この変更箇所に対応する設計判断を発火禁止とし、初期ドキュメントに対応するプレースにトークンを充分数配置する。トークンの充分数とは、そのプレースに接続されている出力アークの和である。

N_3=<DUAUG, BUH, OUCUE, W, M_3>

これにより発火できなかったトランジションが、変更影響箇所ということができる。また、この段階で、発火可能なトランジションは、実際の設計時に用いられた候補案のみに限り、その他の代替案については発火させない。

次に、制約が満たされなくなっている箇所について、他の代替案について制約の確認を行ない、もし利用可能であればそれを提示することにより、設計変更に対する代替案選択を容易化する。その為、まず次の様なペトリネットを作る。

N_4=<DUAUG, BUH, OUCUE, W, M_4>

このネットは、構造はN_3と同じであり、初期マーキングM_4はN_3を発火させた結果のトークンの状態をそのまま用いたものである。このペトリネットを発火させる事により、代替案を用いて影響範囲を縮小できる部分とその範囲を自動的に知る事ができる。

6．3．発火レポート

本システムでは、ペトリネットの発火系列や、他の系列が考えられる箇所、他の代替案を用いた方が従来の選択よりも効用値が上がる箇所の報告を、発火レポートで行う。

発火レポートは、図9の様な構成になっている。まず、上部に全ての発火系列の報告を行う。中間部では、他の発火系列を用いた方が効用値が上がる場合の系列を報告する。下部では、代替案利用可能性の報告を行う。

変更担当者は、この発火レポートを用いて、他の系列を用いたり、代替案の利用を容易に行う事ができる。

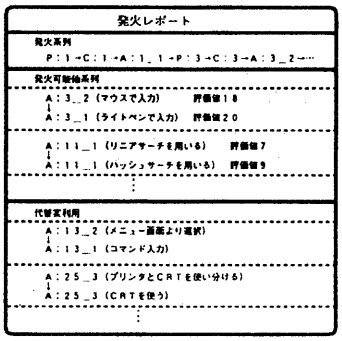

図9：発火レポート

7．まとめ

設計履歴とペトリネットを用いたソフトウェアの変更支援手法について報告した。これにより、ソフトウェア変更時に起こるドキュメント間の対応付けや変更の波及解析などの作業を支援する事ができ、変更担当者が従来行っていた単純で機械的な作業から開放し、ソフトウェア理解、変更に専念する事ができるようになる。

本システムは、現段階でプロトタイプが可動状態にある。今後このシステムを用いて、データを収集し、評価していきたいと思う。

8．参考文献

[1]花田収悦：ソフトウェアの仕様化と設計、日科技連出版社(1986)

[2]島健一：ソフトウェアの設計における設計履歴情報の蓄積、活用法、情報処理学会ソフトウェアの工学研究会81-4(1991)

[3]安達、浜田、竹中：設計プロセスを利用した修正支援法、情報処理学会ソフトウェア工学研究会80-16(1991)

[4]W．ライシッヒ著、長谷川健介、高橋宏治訳：ペトリネット理論入門、シュプリンガー・フェアラーク東京株式会社

[5]市川惇信：意志決定論、共立出版(1983)