〔非公開〕

ＴＲ－Ｃ－０．０８６

設計プロセスの蓄積と再利用

浜　田　雅　樹
Masaki　HAMADA

１９９３　２．１０

ＡＴＲ通信システム研究所

設計プロセスの蓄積と再利用

平成5年 1月

浜田雅樹

ATR通信システム研究所

〒619-02 京都府相楽郡精華町光台2-2

Tel : 07749-5-1242

Fax : 07749-8-2013

e-mail : hamada@atr-sw.atr.co.jp

目次

1. はじめに

本資料は，ATR通信システム研究所で，平成2年2月から平成5年2月までに行われた「設計知識の構造化と活用」の研究の成果「設計プロセスの蓄積と再利用」について述べる．

設計プロセスの蓄積と再利用は，ソフトウェアの保守を支援する手法から成る．この手法は，設計者が設計プロセスを記録しながら設計し，保守者がその記録を用いて保守コストを削減する．

本資料では，その研究成果の1つである，プロトタイプシステムDIG (Design Information System)について概説し，更にDIGの実現において残された課題および研究の課題等について述べる．

2. DIG

DIGは，設計者に対して設計プロセスを記録しながら設計する機能および保守者に対して設計プロセスの記録を用いて影響波及解析を行うための機能を提供する．

DIGのシステム構成を図1に示す。

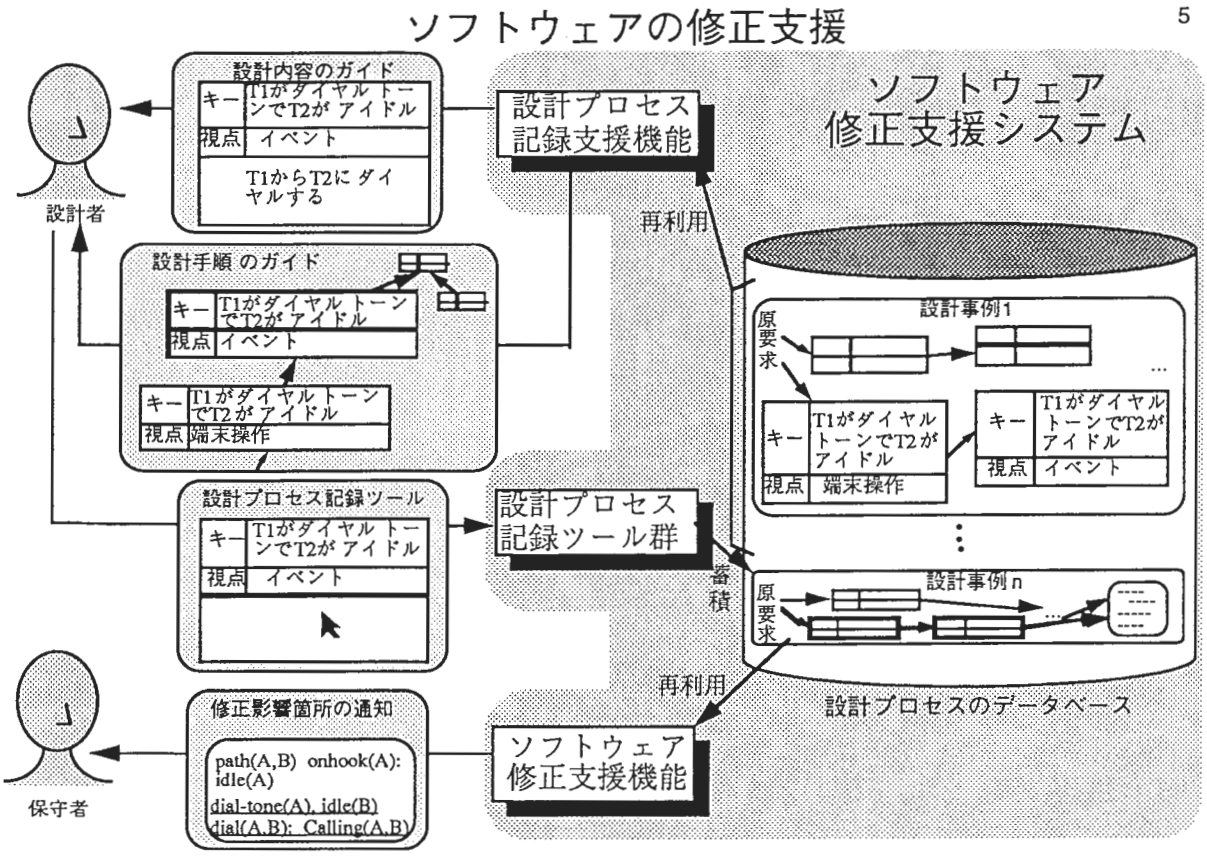


図1

(1) 設計プロセス記録ツール群

設計ビューエディタと設計生産物を記述するDFDエディタを提供している。設計ビューエディタには、視点"機能項目"からの設計結果等を記述するのに適したテキストベースのものと、視点"プロセス･インタラクション"からの設計結果等を記述するのに便利なグラフィカルなものが存在し、設計者が視点に応じて使い分けられるようになっている。

(2) 設計プロセス獲得支援機能

設計プロセス情報を効率的に獲得し、また設計者による視点のばらつきを少なくするため、設計ガイド機能を提供する。

(3) 修正支援機能

設計プロセスの記録を利用してソフトウェアの影響波及解析を支援する機能を提供している。

現在 (1), (3)の試作が完了し、(2)は検討･試作中である。システムは、SPARC Station2上で動作する(smalltalk80で開発)。

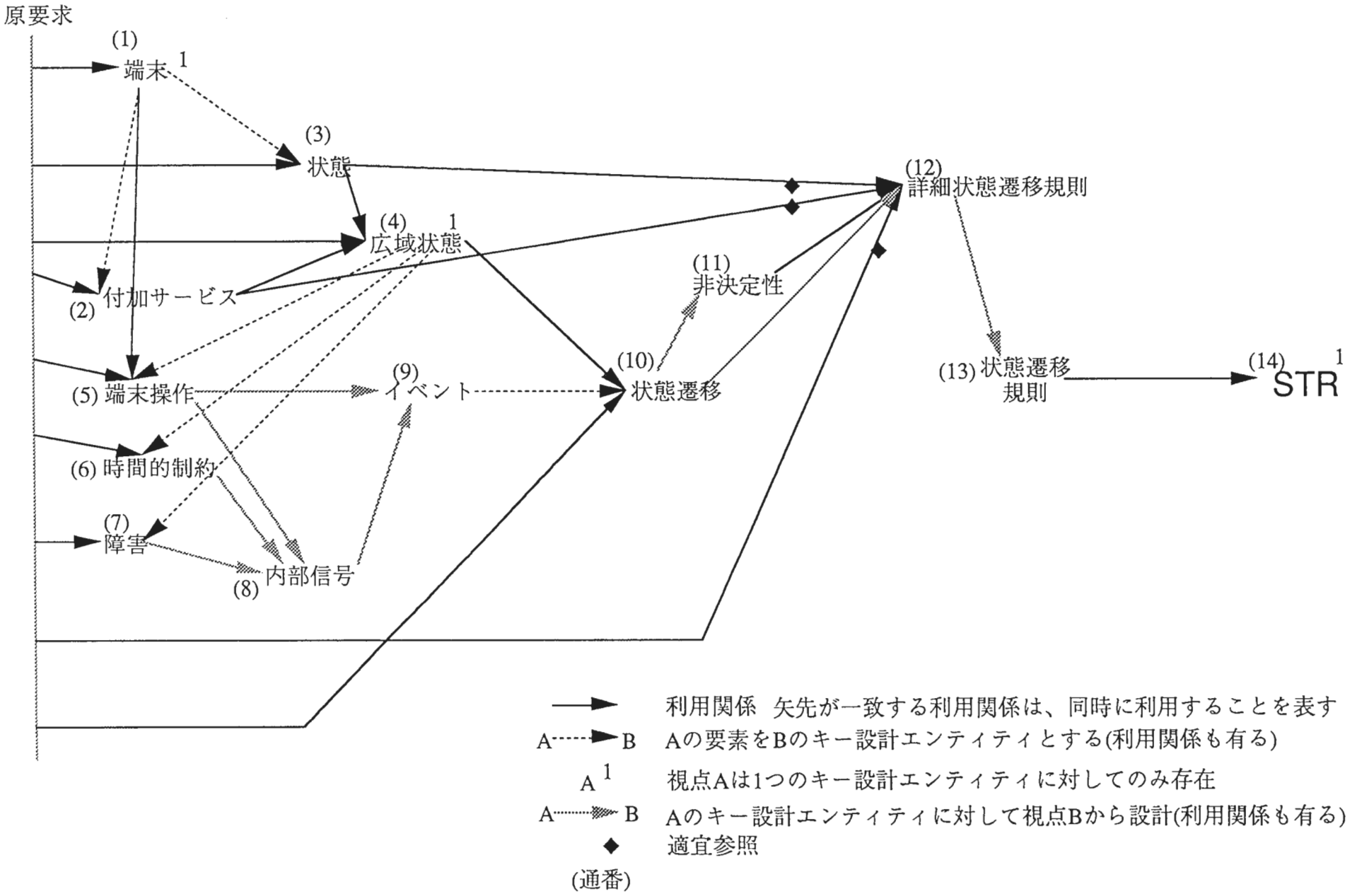
3. DIGを用いた設計プロセスの記録方法

本章では，DIGを用いて設計プロセスを入力しながら設計する方法について説明する．説明は，STR (State Transition Rule)により通信サービスを設計する具体例を用いる．本設計例は，視点が予めすべて決っている場合の例である．適用する領域によっては，視点の一部が予め決り，残りの視点は設計者が設計時に決定することがある．

注) 以下の資料は，以下の読み変えを行う(視点名称の変更).

変更前	変更後
遷移先状態	状態遷移
状態遷移	詳細状態遷移規則
状態遷移規則	一般状態遷移規則

STR設計プロセスにおける視点利用関係



設計プロセスの詳細

通番: 1

設計ビュー: [<サービス>; 端末]

キー設計エンティティ: サービス名(サービスで1つ)

設計方法: 原要求を参照しながら、サービスに関与する端末のリストを作成する。

利用設計ビュー: 原要求

留意点:

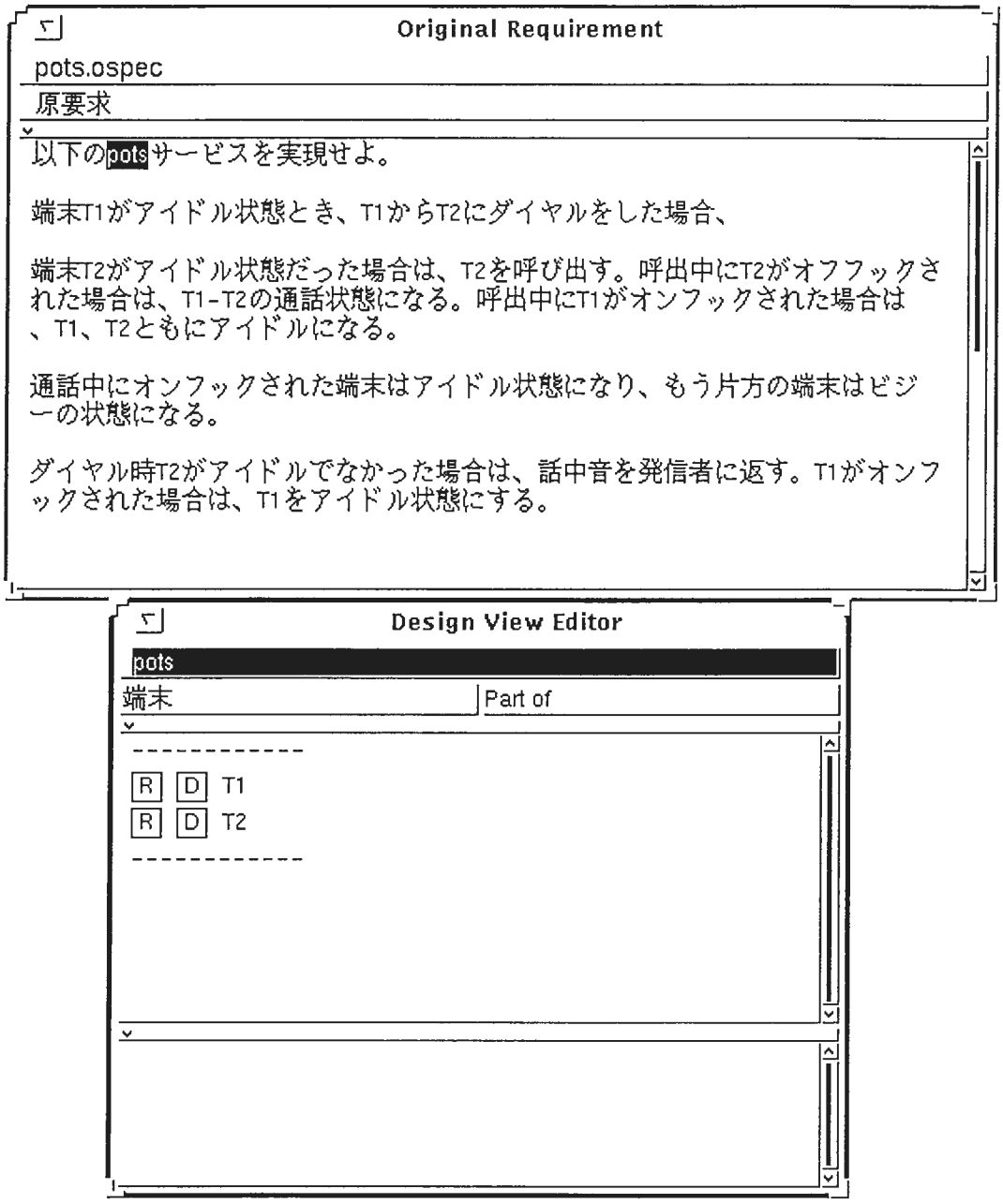


Fig.01

通番:	2
設計ビュー:	[<端末>; 付加サービス]
キー設計エンティティ:	[<サービス>; 端末]に記述されている各端末
設計方法:	各端末にどんな付加サービスが設定されているのかをリストアップする。例えば、端末T2に対して、"端末T3へのコールフォワード"等。
利用設計ビュー:	[<サービス>; 端末]、原要求
留意点:	付加サービスがない場合は、設計ビューを空白とする(何もない場合空白とすることは、以下すべてに共通)。

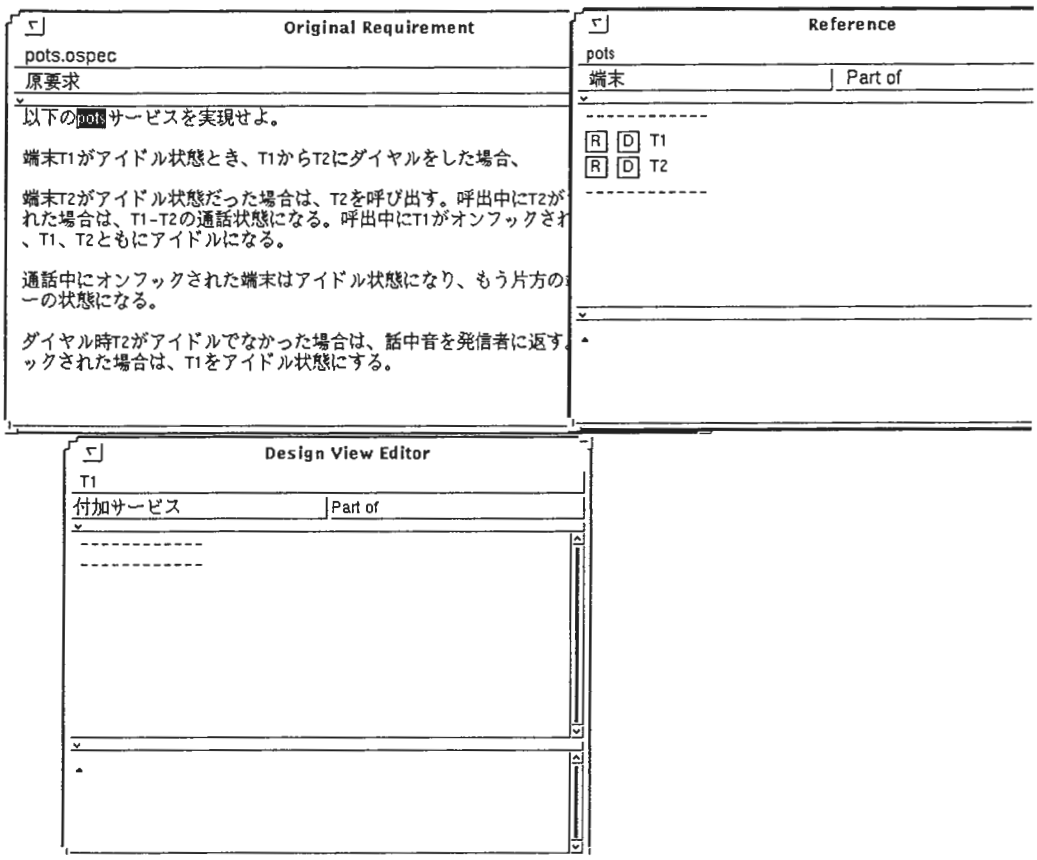


Fig.02

通番:	3
設計ビュー	[<端末>; 状態]
キー設計エンティティ	[<サービス>; 端末]に記述されている各端末
設計方法	各端末に想定される状態(例えば、ダイヤルトーンがなっている状態等)をリストアップする。
利用設計ビュー	[<サービス>; 端末]、原要求

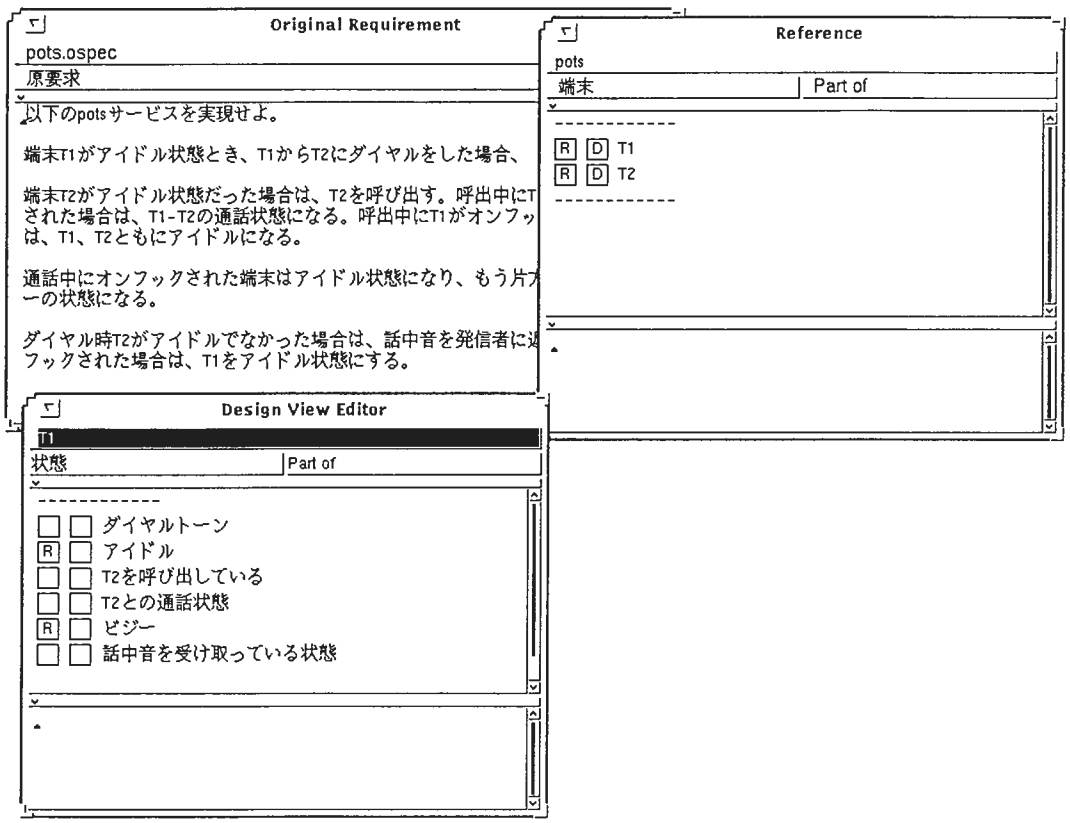


Fig.03

通番:	4
設計ビュー:	[<サービス>; 広域状態]
キー設計エンティティ:	サービス名
設計方法:	通信システム全体としてどんな状態を取り得るかをリストアップする。広域状態は、端末の状態と、その端末に設定されている付加サービスの組み合せで表現する。設計の際、原要求に記述されていない常識的な状態を補う(例におけるダイヤルトーンなど)。
利用設計ビュー:	原要求、全端末に対する[<端末>; 状態]、全端末に対する[<端末>; 付加サービス](以上同時)
留意点:	広域状態を構成する場合、どうでもよい端末の状態は除外する(但し、これは必須ではない。広域状態数の爆発防止が目的)。

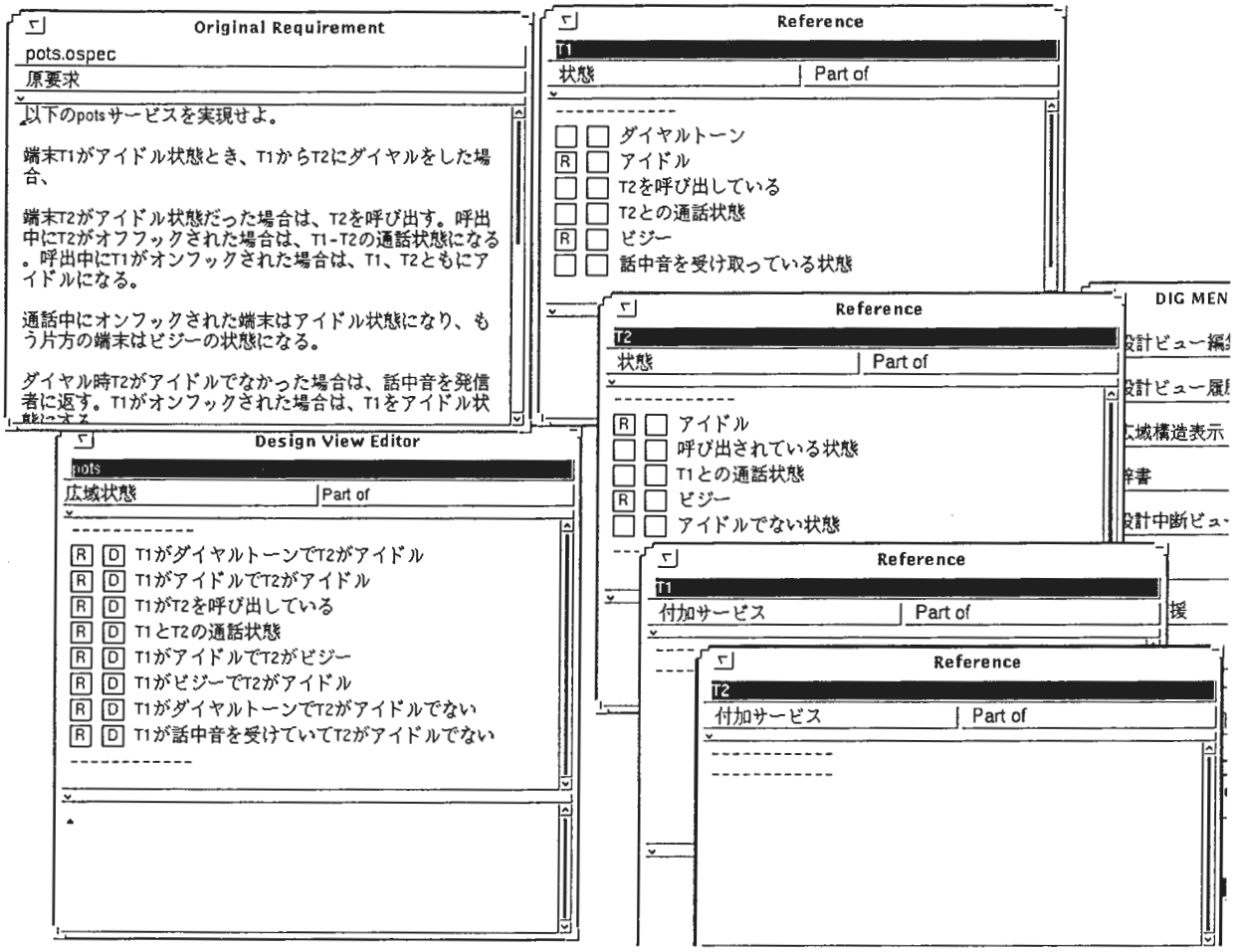


Fig.04

通番:	5
設計ビュー:	[<広域状態>; 端末操作]
キー設計エンティティ:	[<サービス>; 広域状態]に記述されている各広域状態
設計方法:	各広域状態時に想定される端末の操作をリストアップする。
利用設計ビュー:	原要求、[<サービス>; 広域状態]、[<サービス>; 端末]
留意点:	設計エンティティには、どの端末に対してどんな操作が行われるかを表す名称を設定する。

Fig.05

通番:	6
設計ビュー:	[<広域状態>; 時間的制約]
キー設計エンティティ:	[<サービス>; 広域状態]に記述されている各広域状態
設計方法:	各広域状態の存続に関する時間的制約を記述する。例えば、端末T1がダイヤルトーンという状態は60秒間のみ存続可能等。
利用設計ビュー:	原要求、[<サービス>; 広域状態]
留意点:	原要求には指定されていないことが多い。常識的な範囲で設計者が設計する。

通番:	7
設計ビュー:	[<広域状態>; 障害]
キー設計エンティティ:	[<サービス>; 広域状態]に記述されている各広域状態
設計方法:	各広域状態で想定される障害をリストアップする。
利用設計ビュー:	原要求、[<サービス>; 広域状態]
留意点:	原要求にはない常識的な障害も付加する。

通番: 8

設計ビュー: [<広域状態>; 内部信号]

キー設計エンティティ: [<サービス>; 広域状態]に記述されている各広域状態

設計方法: 端末操作、時間的制約、障害がそれぞれが起こったときに発生する内部信号を各広域状態別にリストアップする。例えば、T2でオンフックが起こったときは、sig-onhook(T2)という内部信号が他の端末に分配されるので、sig-onhookという内部信号を、オンフックが起こり得る広域状態に対して設計する。

利用設計ビュー: 同じキー設計エンティティを持ち、視点が端末操作、時間的制約、障害の設計ビュー(別々に)。

留意点: 参照では、まず視点が端末操作の設計ビューを参照し、そこから設計される内部信号を記述する。次に、その設計ビューを閉じてから、次に参照する設計ビューを開き、それから設計される内部信号を記述するという形で行う(順番は任意)。

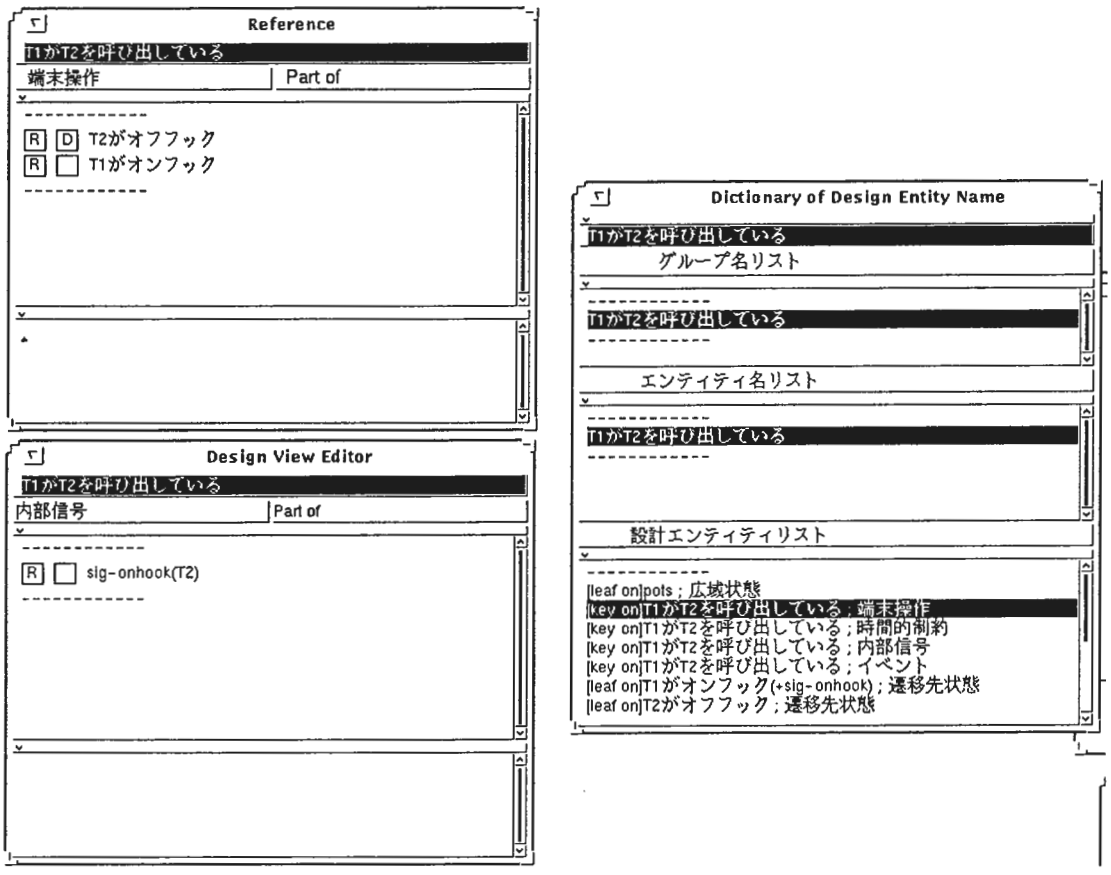


Fig.08

通番:	9
設計ビュー:	[<広域状態>; イベント]
キー設計エンティティ:	[<サービス>; 広域状態]に記述されている各広域状態
設計方法:	各広域状態で想定されるイベントをリストアップする。端末操作と内部信号がイベントとなる。
利用設計ビュー:	同じキー設計エンティティを持ち、視点が端末操作、内部信号の設計ビュー。
留意点:	内部信号をイベントとする場合、以下の考慮が必要となる。例えば、端末T1がオンフックされた直後内部信号sig-onhook(T1)が発生する。しかし、この両者を分けて扱った場合、先に起こるオンフックイベントで中間的に移行する状態は、ユーザの要求から見れば細かすぎる。したがって、イベントとしては、T1のオンフックと+sig-onhookをまとめて1つのイベントとして考えると、ユーザレベルからみた広域状態の遷移と整合がとれる。このイベントの分解は、STRを記述する段階で行う(後述)。

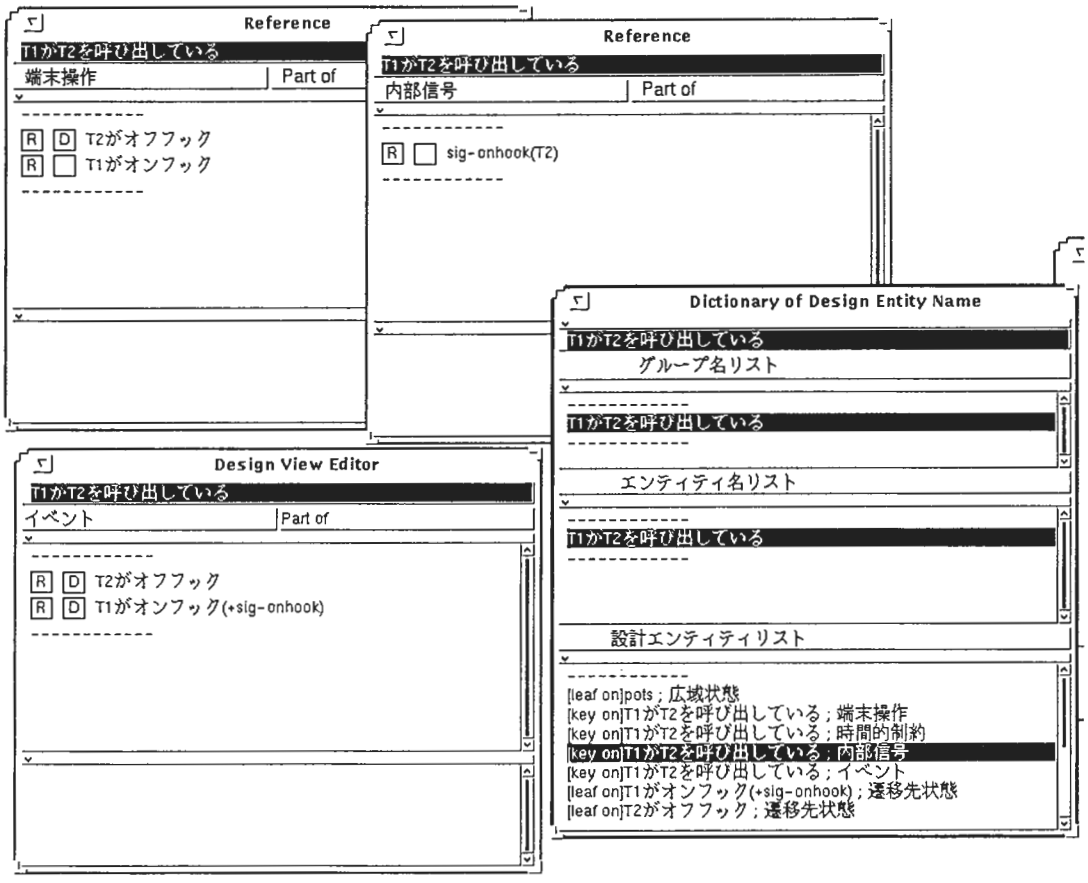


Fig.09

通番:	10
設計ビュー:	[<イベント>; 遷移先状態]
キー設計エンティティ:	全での[<広域状態>; イベント]に記述された各イベント
設計方法:	それぞれのイベントが想定される広域状態(前状態)と、イベント後の広域状態(後状態)を記述する。イベントの前状態は、それが記述されている[<広域状態>; イベント]のキー設計エンティティ全てになる。
利用設計ビュー:	原要求、全[<広域状態>; イベント]、[<サービス>; 広域状態]
留意点:	手順としては、原要求、[<サービス>; 広域状態]を常に開いた状態で、まず任意の[<広域状態>; イベント]を開き、そこに設計中の設計ビューのキー設計エンティティがイベントとして記述されている場合には、開いた設計ビューのキー設計エンティティを前状態として記述する。さらに、原要求等を参考に後状態を記述する(fig.10-1)。次に、その開いた設計ビューを閉じ、別の[<広域状態>; イベント]を開く。以上をすべての[<広域状態>; イベント]を開くまで行えば、1つのイベントに対する視点:遷移先状態の設計が完了し、次に、まだ視点: 遷移先状態から設計されていないイベントを、[<広域状態>; イベント]の記述中から見つけ、それについて設計を行う。この際、利用関係の記録上、すべての[<広域状態>; イベント]を一通りは開く。開いた設計ビューに、設計中のキー設計エンティティとなっているイベントが記述されていない場合、何も設計エンティティは記述せずに閉じる。つまりその設計ビューを利用して設計エンティティが1つも設計されていないのだが、設計ビュー全体の設計としては利用されたという関係を残す必要が有る(fig.10-2は、開いた[T1がアイドルでT2がアイドル; イベント] に、設計中のキー設計エンティティ"T1がオンフック(+sig-onhook)"がないため何も設計(=追加)していない例)。

Fig.10-1

Fig.10-2

通番:	11
設計ビュー:	[<イベント>; 非決定性]
キー設計エンティティ:	全[<イベント>; 遷移先状態]の各キー設計エンティティ(＝イベント)
設計方法:	利用している[<イベント>; 遷移先状態]内に記述されている前状態で2つ以上同じものが存在する場合、その前状態を記述する。
利用設計ビュー:	同じキー設計エンティティを持ち、視点が遷移先状態の設計ビュー
留意点:	単純な作業。

通番:	12
設計ビュー:	[<イベント>; 状態遷移]
キー設計エンティティ:	全[<イベント>; 遷移先状態]の各キー設計エンティティ(＝イベント)
設計方法:	[<イベント>; 遷移先状態]の非決定性を解消する。具体的には、[<イベント>; 非決定性]に記述されている前状態を、原要求を詳細に調べて、さらに[<端末>; 付加サービス]、[<端末>; 状態]などに記述されている状態の条件を追加する。図に具体例を示す(本図は今まで示してきた例とは別個に作成したもの)。非決定性がない場合は、同じキー設計エンティティの視点:遷移先状態と同じ内容を記述する。その際、端末の状態、端末の付加サービス、原要求は参照する必要はない。
利用設計ビュー:	同じキー設計エンティティを持ち、視点が遷移先状態、非決定性の設計ビュー。非決定性になにか記述されている時は、更に、全[<端末>; 状態]、全[<端末>; 付加サービス]、原要求を参照する。
留意点:	

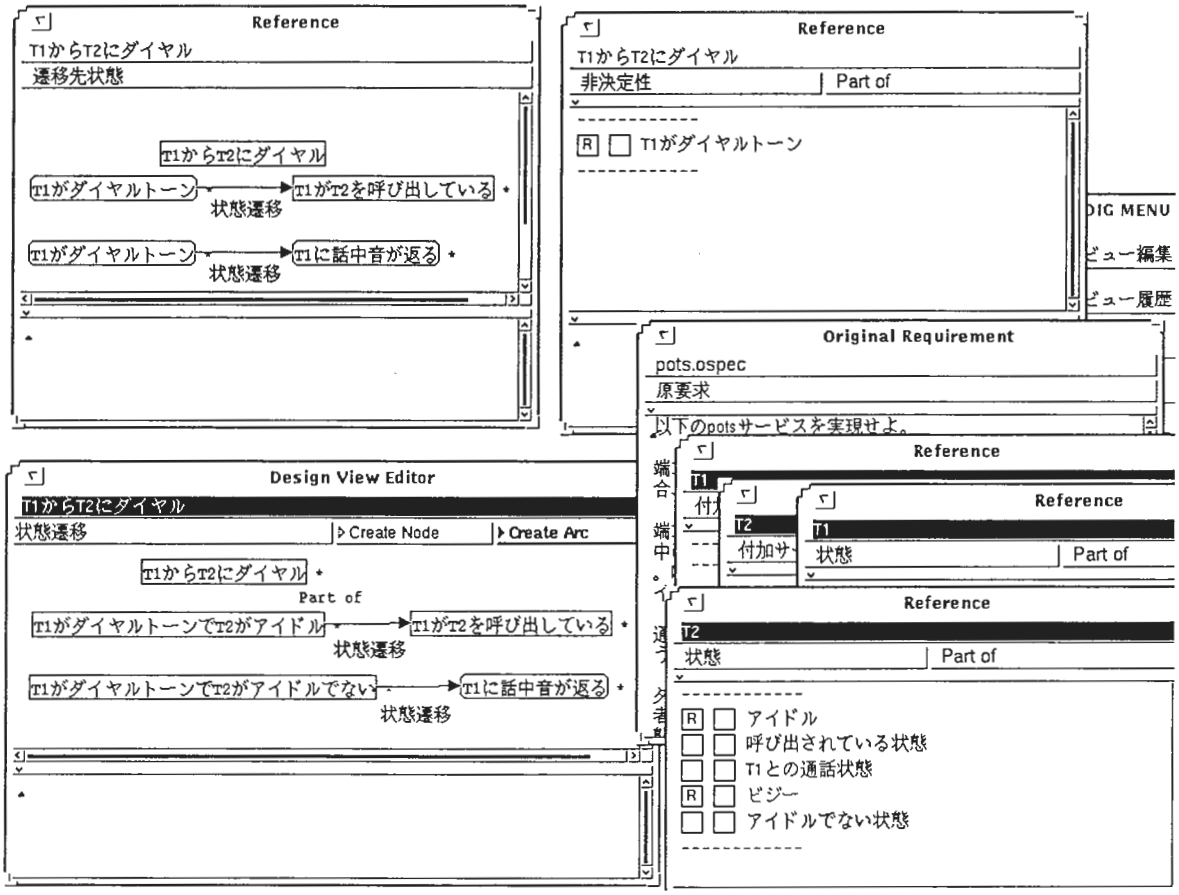


Fig.12

通番: 13

設計ビュー: [<イベント>; 状態遷移規則]

キー設計エンティティ: 全[<イベント>; 遷移先状態]の各キー設計エンティティ(＝イベント)

設計方法: 各[<イベント>; 状態遷移]において、冗長性を排除する。具体的には、前状態と後状態で変化していない状態要素の省略を試みる。その際、省略したことによって前状態の同じものができないようにする(前述の非決定性と同じ考え方)。

利用設計ビュー: 同じキー設計エンティティを持ち、視点が状態遷移の設計ビュー

留意点:

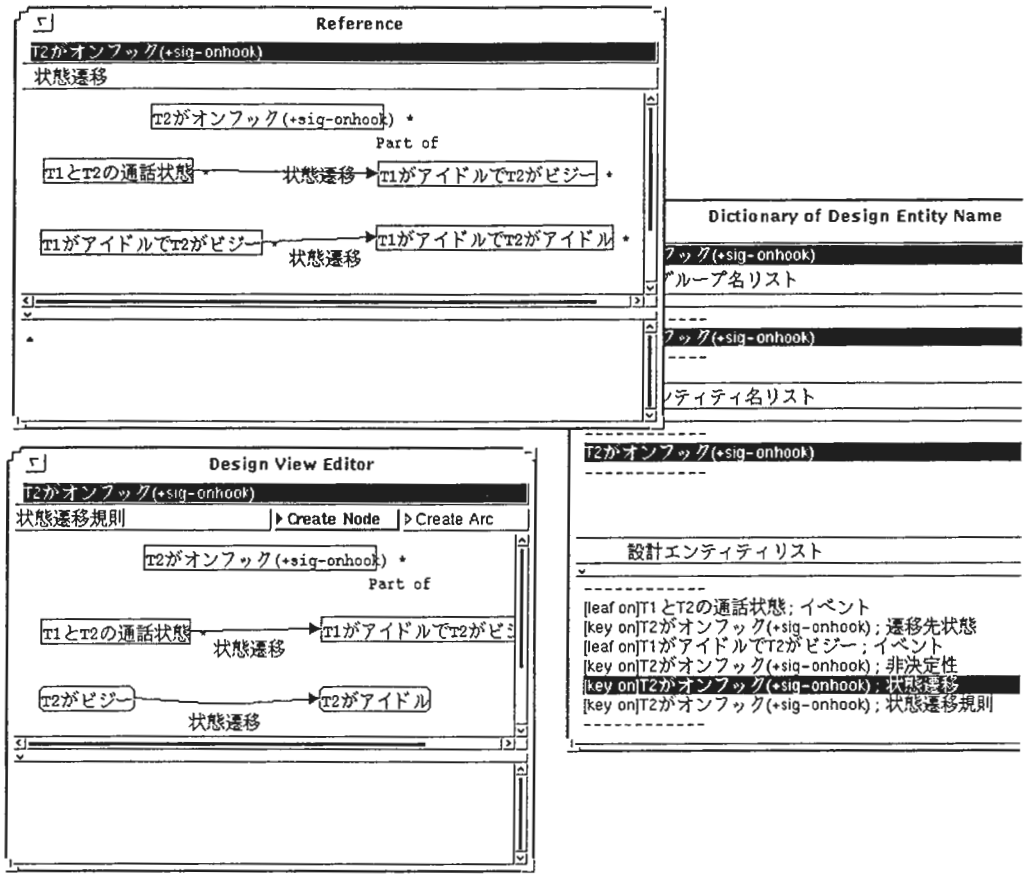


Fig.13

通番:	14
設計ビュー:	[<サービス>; STR]
キー設計エンティティ:	サービス
設計方法:	[<イベント>; 状態遷移規則]を順次参考にしながら、STRを記述する。本作業は、任意の[<イベント>; 状態遷移規則]を開き、STRを記述した後、閉じて、別の[<イベント>; 状態遷移規則]を開き、また設計するという作業を繰り返す。
利用設計ビュー:	[<イベント>; 状態遷移規則]
留意点:	内部信号の扱いを注意する必要が有る。fig.14-1にあるイベント"T2がオンフック(+sig-onhook)"で起こる状態遷移は、2つのイベントで起こることを表している。状態遷移規則にある最初の遷移からSTRの最初の2つの規則が設計されている。通話状態の広域状態はSTRは、path(A,B), path(B,A)と表される。on-hook(A)が起こると、A側からのつながりが切れるため、最初のSTR規則が生成される。また、オンフックイベントが起こっていない端末では、相手でオフフックが起こったと言う知らせsig-onhookを受けてそちら側からの接続を切る(端末が変数化されているので、2つ目の規則ではT2もAで記述されている点に注意)。 STRでは端末が変数かされるため、端末の別が意味を持たない場合があるので注意する。例えば、fig-14-2において、[T2がオンフック(+sig-onhook); 状態遷移]の最初の遷移は、ルールとして記述した場合は、[T2がオンフック(+sig-onhook); 状態遷移]を利用して作ったSTRの2つの規則と同じになるのでこの時は何も作成しない。

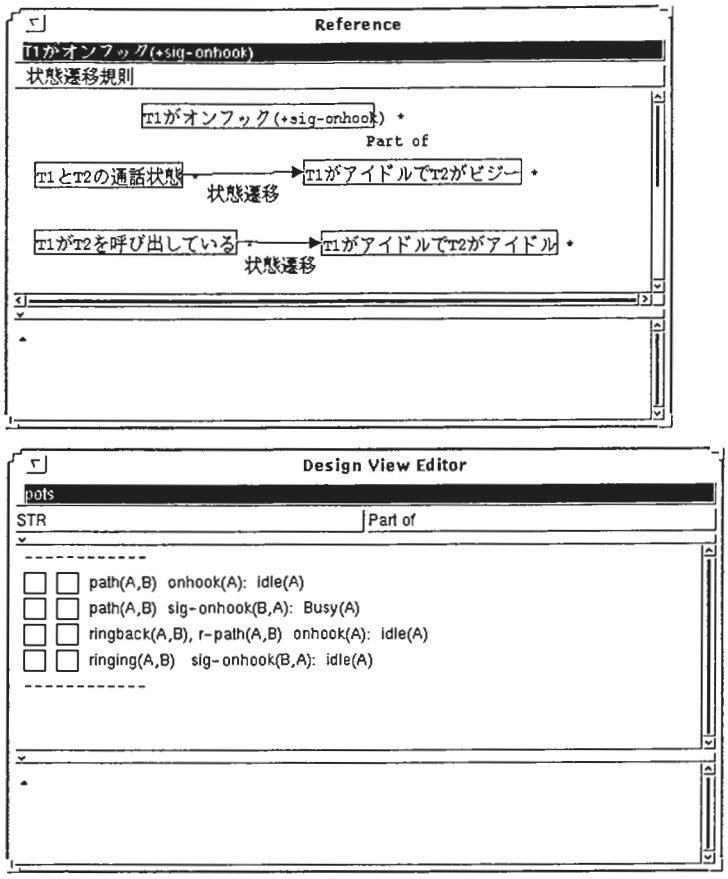


Fig.14-1

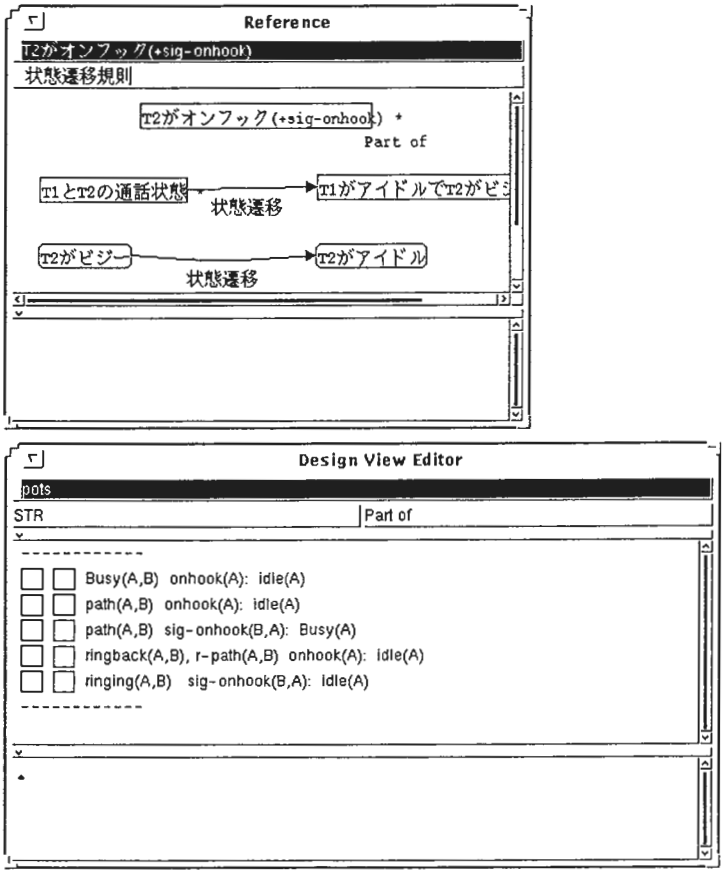


Fig.14-2

視点	キー設計エンティティ	設計ビューに記述される設計エンティティ, 関係
端末	サービス名	端末名, part-of
付加サービス	端末名	サービス名, part-of
状態	端末名	状態名, part-of
広域状態	サービス名	広域状態名, part-of
端末操作	広域状態名	端末操作名, part-of
時間的制約	広域状態名	時間的制約内容, part-of
障害	広域状態名	障害名, part-of
内部信号	広域状態名	内部信号名, part-of
イベント	広域状態名	イベント名, part-of
状態遷移	イベント名	広域状態名、遷移関係
非決定性	イベント名	広域状態名, part-of
詳細状態遷移規則	イベント名	広域状態条件、遷移関係
状態遷移規則	イベント名	広域状態条件、遷移関係
STR	—	—

表 STR設計プロセスにおける設計ビュー一覧

4. DIGを用いた修正支援(デモシナリオ)

本章では，DIGを用いたソフトウェアの修正方法について，DIGのデモシナリオを用いて説明する．デモは，前半が設計プロセスの記録方法のデモ，後半が修正(影響波及解析)方法のデモとなっている．

(1) 環境

ホスト	CS46上
アカウント	dig
ロケーション	digのホーム(/home2/dig)

(2) 起動

デモは2通りの方法が考えられる．

<< CS46のコンソール上で行う場合 >>

ホームディレクトリで，

startdemo

を起動する．

<< 他のマシン(CS51, 52)で行う >>

1) acount name = test で login する。

2) xinit

3) cd ~hamada/demo

4) startohp (説明用OHPを表示させない場合は不用)

5) xhost cs46

6) rsh cs46

7) 聞いてきた passwdに 対して リターンのみ入力

8) login: に 対して dig で login する。
(ここまでで、 cs46に account digで login している)

9) cs51 (cs52の 場合は cs52) を 起動

(3) その他

説明用のOHPは，以下の方法で表示される．表示は，xvというパブリックドメインのソフトウェアを使用．ohpは，gif形式で格納されている．

CS46上で表示させる場合は，ホームディレクトリで

xv ohp/DIG-Panel*&

を実行する．

CS51, 52で表示させる場合は，上記の4)を実行する．

(4)デモ操作

導入 ステップ1のパート1

それでは、設計プロセスを獲得し影響波及解析を支援するツールDIGについてご説明します。
まず、設計プロセスの獲得方法について、お見せします。

	Step1 アイコンをクリックする（左ボタン）	メイン・メニューが開く
開かれたウィンドウがDIGのメインメニューです。各種エディタの起動、記録した設計プロセスの参照、影響波及解析支援機能の起動などを行います。	メインメニューの "検索" を押しながら "設計ビュー履歴" を選択する（左ボタン）	ViewHistory ウィンドウが開く
	"<POTS;原要求>" を選び（左ボタン）し、"ビュー表示" を選択（中ボタン）する	DesignView ウィンドウが開く（<POTS;原要求>）
これがユーザが実現したいソフトウェアの要求が記述してある原要求です。 今、この原要求には、普通の電話サービスについて書かれています。 設計者はこの原要求を読んで、実現するソフトウェアの概要を掴もうとします では、この電話サービスであるPOTSについて設計を進めて行きます。	原要求にある「… POTS …」の "POTS" という文字列を選択（左ボタンで " と P の間をダブルクリックする）し、"設計進展" を選択（中ボタン）する	視点設定用ウィンドウ（ダイヤログボックス）が開く
設計者は、今POTSについて設計しようとしている訳ですが、これをどういう視点から設計するのか、その視点をこのウィンドウで設定します。		
まず、POTSに関与する端末について設計しようということで、設計者はカテゴリとして "アーキテクチャ"、視点名として "端末" を選びます。	アーキテクチャを選択する（左ボタン） 端末を選択する（左ボタン）	
設計ビューを記述するエディタの種類を指定します。現在は、テキストベース、グラフィカルな図形ベースの物、表形式の物などを記述するためのエディタを提供しています。ここでは、テキストエディタを選びます。	テキストエディタを選択する（左ボタン） accept を選択（左ボタン）	DesignViewEditor ウィンドウ（<POTS;端末>）が開く
開いたウィンドウは設計ビューを入力するエディタです。 キー設計エンティティとして、先ほど選んだ "POTS" が設定されています。 また、視点として "端末" が設定されています。 設計ビューにはPOTSサービスに関与する端末をリストアップした結果を記述します。		
どうも、原要求にPOTSに関与する端末について書いてあるようなので、そちらを参照します。	ViewHistory ウィンドウで、"<POTS;原要求>" を選び（左ボタン）し、"ビュー表示" を選択（中ボタン）する	DesignView ウィンドウが開く（<POTS;原要求>） 参照するかどうか尋ねてくる
	参照問い合わせウィンドウで、"yes" を選択する（左ボタン）	
このように他の設計ビューを参照する際には、参照するか否かを尋ねてきますので "yse" を選択します		

この原要求を参照してPOTSに関与する端末のリストアップを行っていくわけですが、設計済の例がありますので、そちらの方をお見せします。	ViewHistoryウィンドウの"<POTS;端末(設計済)>"を選び（左ボタン）、ビュー表示を選択（中ボタン）する。	設計ビューウィンドウ<POTS;端末(設計済)>が開く 参照するかどうか尋ねてくる
	さりげなくYesを選択する（左ボタンでクリック）	
	DesignViewウィンドウ（<POTS;端末(設計済)>）をDesignViewEditorウィンドウ（<POTS;端末>）の上にさりげなく重ねる。	
POTSに関与する端末として、このような端末が設計されています。 このようにして、設計作業を進めて行きます。	DesignViewウィンドウ（<POTS;端末(設計済)>）をWindowSystemのアイコンにする（ウィンドウ左上の▽マークをクリック）	
	設計ビューエディタウィンドウ（<POTS;端末>）で、さりげなく関係"Part-of"を設定する（中ボタンでメニュー選択）	
	設計ビューエディタウィンドウ（<POTS;端末>）で、設計完了（中ボタン）を選択する	
	Step1のメインメニューで"設計終了"を選択する	

導入 ステップ２のパート１

設計が進んだ段階でも、同様にして設計プロセスを記録しながら設計を進めて行きます。今度はその様子をお見せします。

	Step2 アイコンをクリックする（左ボタン）	メイン・メニューが開く
	メインメニューの "検索" を押しながら "設計ビュー履歴" を選択する（左ボタン）	ViewHistory ウィンドウが開く
設計が進んだ段階の例として、"<POTS;広域状態>" から更に設計を進めていく例をお見せします。	"<POTS;広域状態>" を選び（左ボタン）し、"ビュー表示" を選択（中ボタン）する	DesignView ウィンドウが開く（<POTS;広域状態>）
"<POTS;広域状態>" は、通信システム全体として、どんな状態を採り得るかをリストアップしたものです。 こんどは、それぞれの広域状態について、どんな端末操作が想定されるのかを設計していきます。	"T1 がダイヤルトーンで T2 がアイドル" を選択（左ボタン）し設計進展（中ボタン、メニュー選択）を選ぶ。	視点設定用ウィンドウ（ダイヤログボックス）が開く
先ほどと同様に視点の設定を行います。	ふるまいを選択する（左ボタン） 端末操作を選択する（左ボタン） テキストエディタを選択する（左ボタン） accept を選択（左ボタン）	DesignViewEditor ウィンドウ（<T1 がダイヤルトーンで T2 がアイドル;端末操作>）が開く
何もない状態で設計するのは難しいので、原要求や、過去に記録した設計ビューである<POTS;端末> などを参照して設計を行います。	ViewHistory ウィンドウで、"<POTS;原要求>" を選び（左ボタン）し、"ビュー表示" を選択（中ボタン）する	DesignView ウィンドウが開く（<POTS;原要求>） 参照するかどうか尋ねてくる
先ほどと同様に参照の場合には "yes" を選択します。	参照問い合わせウィンドウで、"yes" を選択する（左ボタン）	
	ViewHistory ウィンドウで、"<POTS;端末>" を選び（左ボタン）し、"ビュー表示" を選択（中ボタン）する	DesignView ウィンドウが開く（<POTS;端末>） 参照するかどうか尋ねてくる
	参照問い合わせウィンドウで、"yes" を選択する（左ボタン）	
これらの設計ビューを参照して設計を行っていきます。 既に設計済の例がありますのでそちらをお見せします。	ViewHistory ウィンドウで、"<T1 がダイヤルトーンで T2 がアイドル;端末操作(設計済)>" を選び（左ボタン）し、"ビュー表示" を選択（中ボタン）する	DesignView ウィンドウが開く（<T1 がダイヤルトーンで T2 がアイドル;端末操作(設計済)>） 参照するかどうか尋ねてくる
	さりげなく Yes を選択する（左ボタンでクリック）	
	DesignView ウィンドウ（<T1 がダイヤルトーンで T2 がアイドル;端末操作(設計済)>）をDesignViewEditor ウィンドウ（<T1 がダイヤルトーンで T2 がアイドル;端末操作(設計済)>）の上にさりげなく重ねる。	

端末操作として、このようなものがリストアップされています。 このような設計を行うのに、原要求や他の設計ビューを参照しながら設計したという関係をシステムが利用関係として自動的に記録します。	DesignView ウィンドウ（<T1 がダイヤルトーンで T2 がアイドル;端末操作（設計済）>）をWindowSystem のアイコンにする（ウィンドウ左上の▽マークをクリック）	
このように、設計が進んだ段階でも同様にして、設計プロセスを記録しながら設計を進めて行きます	設計ビューエディタウィンドウ（<T1 がダイヤルトーンで T2 がアイドル;端末操作>）で、設計完了（中ボタン）を選択する	
	設計ビューエディタウィンドウ（<T1 がダイヤルトーンで T2 がアイドル;端末操作>）で、さりげなく関係 "Part-of" を設定する（中ボタンでメニュー選択）	
	Step2 のメインメニューで "設計終了" を選択する	

導入 ステップ3のパート1

最後に、今までに述べてきたような方法で獲得した設計プロセスを用いて、要求が修正された場合の影響波及の解析を支援する方法についてお見せします。

	Step3 アイコンをクリックする（左ボタン）	メイン・メニューが開く
	DNA Maintenance アイコンをクリックする（左ボタン）	DNA Maintenance ウィンドウが開く
	メインメニューの "検索" を押しながら "設計ビュー履歴" を選択する（左ボタン）	ViewHistory ウィンドウが開く
	"<POTS;原要求>" を選び（左ボタン）し、"ビュー表示" を選択（中ボタン）する	DesignView ウィンドウが開く（<POTS;原要求>）
例えば、ダイヤルして電話をかけるようにしていた所を、フラッシュするだけで特定の所に電話がかかるように変更することになったとします。 修正支援機能では、複数の修正案を作成する機能を有しています。ここで新しい修正案を検討してみます。	DNA Maintenance ウィンドウで新規修正案ボタンをクリックする（左ボタン）	新規修正案名の入力用ウィンドウが現れる
	新規修正案名の入力用ウィンドウに "case2" を入力する	DNA Maintenance ウィンドウに新規修正案 "case2" が表示される
では、まず、原要求に記述されているどんな設計対象の性質が設計で利用されているかを調べてみます。	DesignView ウィンドウの "原要求" と記述されている箇所で参照情報（中ボタン、メニュー選択）を選ぶ。	UsedRelation ウィンドウが開く
原要求には様々な設計対象の性質が書かれているわけですが、設計者はその中から設計で重要であると考えた設計対象の性質に着目して設計を行います。 以下は、設計者が原要求に書かれている設計対象の性質の中で着目した設計対象の性質を表しています。 今度の修正は、端末操作が変更されるので、更に詳しく調べてみます。	端末操作を選び（左ボタン）、設計ビューを表示する（中ボタン、メニュー選択）を選ぶ。	設計ビューのキー設計エンティティ一覧が表示される。
今回の修正では、T1 がダイヤルトーンの時の端末操作が変わるので、この2つが変更されることがわかります。 それぞれについて調べるわけですが、まず、この時（T1 がダイヤルトーンで T2 がアイドル）について調べます。	"T1 がダイヤルトーンで T2 がアイドル" を選び（左ボタン）、内容を表示する（中ボタン、メニュー選択）を選ぶ。	DesignView ウィンドウ（<T1 がダイヤルトーンで T2 がアイドル;端末操作>）が開く。
赤く表示されているのは、この設計ビューの中で、原要求を利用して設計された部分を表しています。 今回の修正では、これは変わりますので、修正が必要であることをマークします。	"T1 がダイヤルトーンで T2 がアイドル" の所で修正対象にする（中ボタン、メニュー選択）を選ぶ。	DNA Maintenance ウィンドウに修正項目が表示される
このように修正対象にするを選ぶと、先ほどの修正案にその設計ビューが修正すべきものとして登録されます。		

更にこの設計対象の性質を利用して設計されている他の設計対象の性質を探します。	DesignView ウィンドウの "T1 がダイヤルトーンで T2 がアイドル" の所で、参照情報（中ボタン、メニュー選択）を選ぶ。	UsedRelation ウィンドウが開く
先ほどの "T1 がダイヤルトーンで T2 がアイドル" の "端末操作" という設計対象の性質を利用して設計されたのは、"内部信号"、"イベント" であることがわかります。 それぞれについてチェックする必要があるわけですが、まず、イベントについて調べて行きます。	UsedRelation ウィンドウで "イベント"を選択（左ボタン）し、設計ビューを表示する（中ボタン、メニュー選択）を選ぶ。	設計ビューのキー設計エンティティ一覧が表示される
"T1 がダイヤルトーンで T2 がアイドル" は変わりますので先ほどと同様にして、内容を見てみます。	UsedRelation ウィンドウで "T1 がダイヤルトーンで T2 がアイドル" を選択（左ボタン）し、内容を見る（中ボタン、メニュー選択）を選ぶ。	DesignView ウィンドウ（<T1 がダイヤルトーンで T2 がアイドル;イベント>）が開く
赤く表示されているのは、先ほど修正対象にした設計ビューを利用して設計された部分を表しています。 今回の修正では、これも変わりますので、修正対象にします。	"T1 がダイヤルトーンで T2 がアイドル" の所で、修正対象にする（中ボタン、メニュー選択）を選ぶ。	DNA Maintenance ウィンドウに修正項目が表示される。
以下同様に行っていきますと、設計生産物（STR）までたどりつきます。 予め直前まで行った例がありますのでそちらをお見せします。	DNA Maintenance ウィンドウで non-dial call を選ぶ（左ボタン）	
	修正項目の一番最後を選び（左ボタン）、設計ビューを開く（中ボタン、メニュー選択）を選ぶ。	DesignView ウィンドウ（<???;状態遷移規則>）が開く
	DesignView ウィンドウの…の所で参照情報（中ボタン、メニュー選択）を選ぶ。	UsedRelation ウィンドウが開く
STR はこの設計における最終的な設計生産物です	UsedRelation ウィンドウで、STR を選び（左ボタン）、設計ビューを表示する（中ボタン、メニュー検索）を選ぶ。	UsedRelation ウィンドウにキー設計エンティティ一覧が表示される。
	UsedRelation ウィンドウで "POTS;STR" を選び（左ボタン）、内容を見る（中ボタン、メニュー選択）を選ぶ。	DesignView ウィンドウ（<POTS;STR>）が開く
このように STR に影響を受けると考えられる箇所が赤く表示されました。 ここでは、これらの規則に対して、影響がでている可能性があることがわかりました。		

5. 積み残しの課題

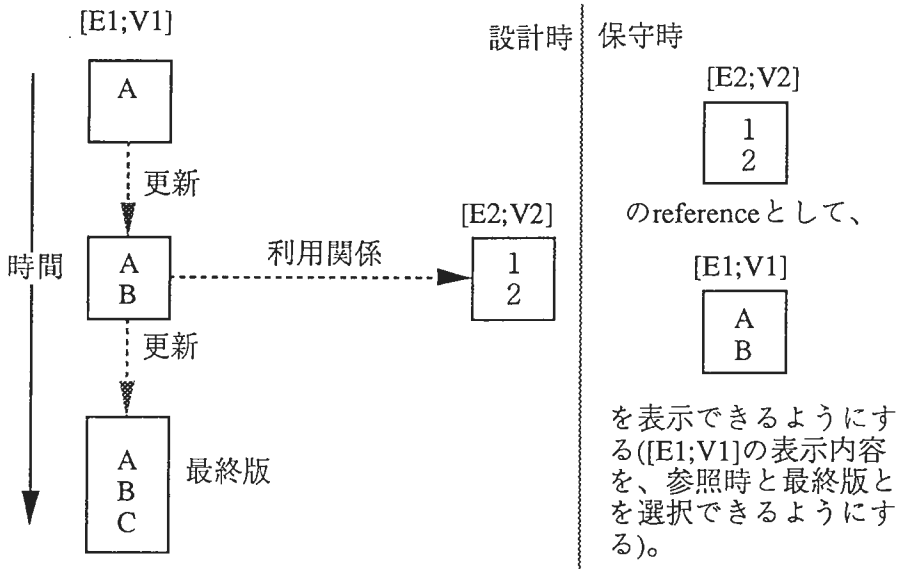
以下は，研究の項目の計画，進捗および残された課題をまとめたものである．

項目		計画	進捗	残された課題 (以下で解説する章番号)
研究の思想，位置付け等		100%	100%	
設計プロセス記録ツール群				
	プロトタイプ	90%	80%	1, 2, 4, 6
	既存CASE toolとの結合方式	0%	0%	8
設計プロセス記録支援方式				
	設計プロセス再利用	70%	60%	9, 10, 試作, 評価
	視点利用規則	80%	70%	3
修正支援機能		80%	70%	5, 7, 11

以下，これらの残された課題について述べる(章は重要度の高い順に配置).

1. 差分の厳密な管理

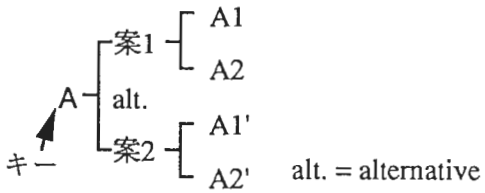
設計ビューがその更新の途中で他の設計ビューの設計で参照されている場合が考えられる．その際，参照時点での設計ビューの内容を見せる機能が必要ではないか？



2. Alternativeの扱い

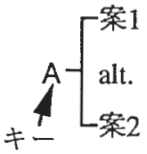
設計における解のAlternativeは，現在，設計エンティティ間の関係として記述することができる．しかし，alternativeについては，今までの検討から，多少特殊な扱いが必要であることが分かっている．例えば，機能項目Aの機能項目としてA1, A2から構成する方法と，A1', A2'から構成する方法の2種類の案を考え付いた場合，本来なら，

視点：機能項目



と記述したいところである．ところが，現在のDIGでは，このような記法は2段階に分けて記述する必要がある．しかし，まず，設計ビューとして，2つのalternativeがあることを

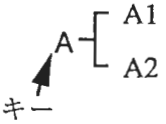
視点：処理方式



のように記述するまでは良いが，Aに対する機能項目を設計する設計ビューが記述できない．即ち，キー，視点以外に，どのalternativeに対する設計かというインデックスが必要になる(下の例)．

視点：機能項目

処理方式=案1



この処理は実現上の難しさはない．但し，大幅な変更が必要なのは，今まで設計ビューの識別子は，キーと視点のペアーであったが，この方式だと，キー，視点，alternativeのリストの組が識別子となる．

現在のDIGにはこの処理を行う機能はインプリメントされていない．この機能を設計した結果については，1991年2月に試作としてJIPより納入されたDIGの設計書に記述されている．本設計は，symbolics上に試作することを前提に行われた．その後，SPARC＋Smalltalk80に試作環境が変更された際にこの機能が積み残しされたままとなっている．

3. 視点利用規則の評価

現在検討している視点利用規則の評価を行う必要がある．本評価は，設計プロセスの記録の質を安定化させるのが目的であるが，その評価は難しいため，以下の項目について評価する．

記述性

- 幾つかの設計法を記述し，記述できることを確認する．
- 記述の規模(ルール数)．
- 記述の容易さについての定性的評価．

目標提示機能の評価

- 計算された設計目標数の平均値
- 計算のスピード
- 設計目標の理解の容易性(設計者は示された場合，すぐに理解できるか)

- 設計プロセス記録が改善されたかということに対する設計者の感想

その他

- なんでも

4. 領域知識からの利用関係の記録

本文でも述べたように，DIGでは，現在領域知識からの利用関係は記録していない．Symbolicsに作成されているCOSMOSは，技術ドキュメントをオンライン化するためのツールであり，これをDIGに取り込めば，領域知識からの利用関係を記録するための機能を作成することができる．

手順としては，まず，通信プロトコルデータベースとその参照機能をSPARC+st80環境へ移植する．次に，機能の面での結合について検討する．まず，COSMOSの設計中の参照行為から利用関係を自動獲得する機能，および，原要求だけでなくCOSMOSから影響波及解析をスタートする機能，設計進展をCOSMOSから行う機能の追加を行う．さらに高度な結合として，COSMOSからキーや視点に関する情報を抽出したり，COSMOSの内容をDNAにコピーする機能などを検討する．

5. 利用関係の厳密化

例えば，[出庫処理; 機能項目]の設計で，[出庫依頼; 媒体]を参照しなければいけないのは，出庫依頼が出庫処理の入力データだからである．しかし，入力データであることを書いた設計ビューが利用関係として記録されることは希と考えられる．

現在の設計ガイドや影響波及解析では，出庫依頼は出庫処理の入力データであることが在庫管理プログラムの常識として扱えているため成立している．

従って，あるキー設計エンティティの設計ビュー設計において，違うキー設計エンティティの設計ビューを利用した場合は，どこかにそのキー設計エンティティどうしの関係に関する情報が必要と成る．

これは，自動的に判別が可能である．従って，異なるキー設計エンティティを持つ設計ビュー間の参照が生じた場合，参照設計ビューの中にそれらのキー設計エンティティ間の関係を記述したものがあるか調べる．無かった場合の処理として以下の2通りが考えられる．

1) ユーザに警告を出して，参照設計ビューを追加させる．

2) システムが裏でそれらのキー設計エンティティ間の関係を記述したものを検索する．検索した結果は，利用関係に自動追加するか，または利用関係とは別のものとして保存する．どちらが良いかは検討が必要．

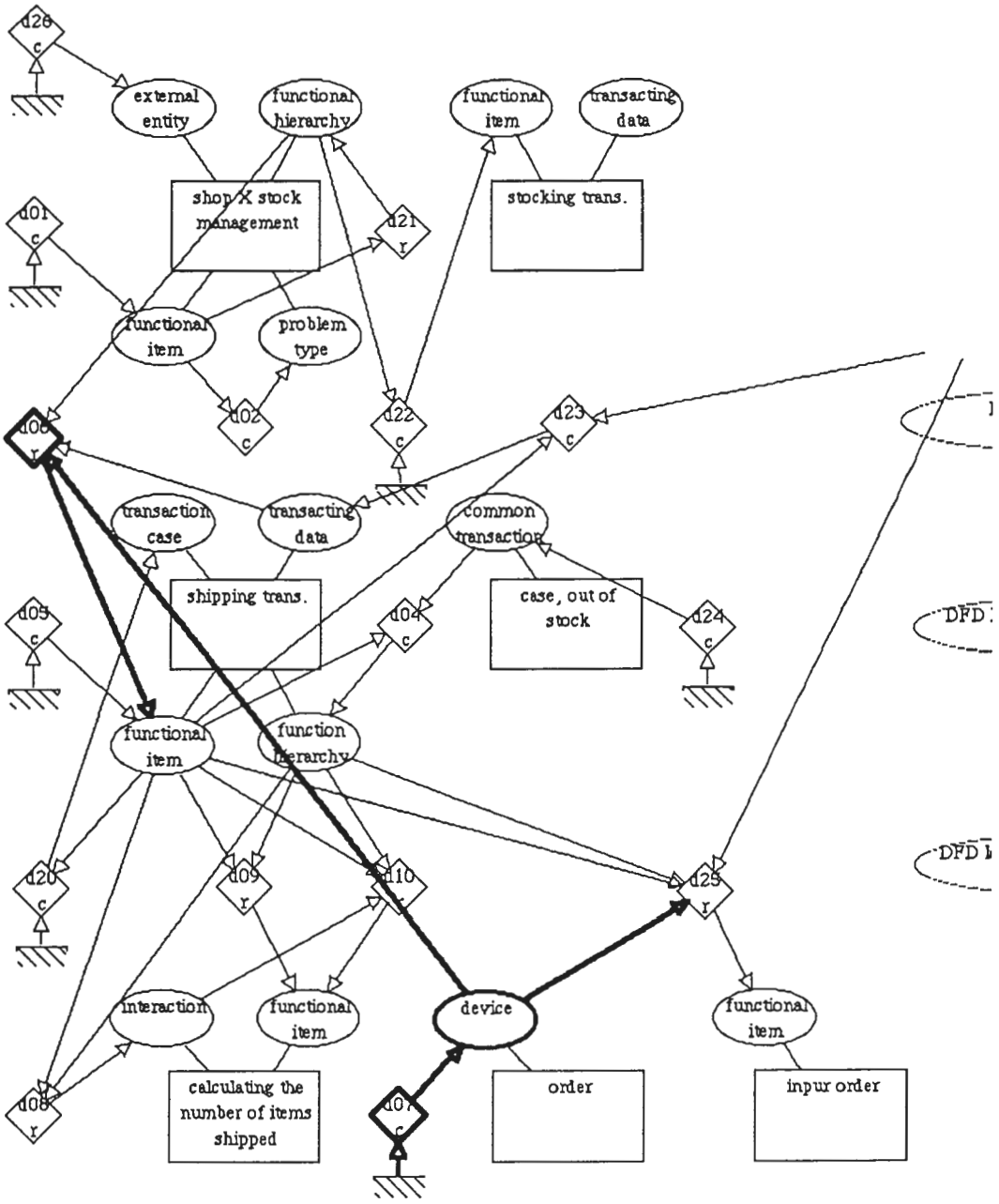
6. オブジェクト指向データベース

DIGにおいて，現在，設計プロセスデータは，smalltalk80の仮想メモリに配置されている．したがって，記録できる設計プロセスの大きさには限界が有る．この問題を解決するには，オブジェクト指向データベース(OODB)を用いた設計プロセスの管理機能を実現するのが望ましい．将来的な設計プロセスの再利用機能も考え，OODBによる設計プロセスのの集中，分散管理などを行うことが必要である．

7. グラフィカルノーテーション

設計時の進捗把握や，影響波及解析時の進捗把握には，設計プロセスのグラフィカル表示が必要に成る．

現在，E-R Diagramに似せた表記法を検討している(以下の例は，11で述べるdecisionを記録する場合の例である)．



四角：キー設計エンティティ，楕円：視点，菱形：Decision，アーク：利用関係

8. CASE toolとの関係

提案している手法は，設計法とは無関係である．ユーザが使用しているCASE systemに対して設計プロセス記録機能が簡単に付加できる機構が実現できれば大変有効である．現在，設計プロセスを記録する機能を有するレイヤをウィンドウシステムの上に被せ，その上に従来のCASE toolを載せる方法を検討している．CASE toolから見て，下のレイヤは，ウィンドウシステムと上位コンパチの機能を有する．設計プロセスのレイヤは，ユーザの発するイベントがウィンドウシステムからCASE toolに渡される前に刈り取り，必要な処理をする．また，CASE toolからウィンドウシステムに対するwindow操作に関する処理要求は素通りさせれば良い．

9. 設計事例データベースの整理機能

省略．

10. 領域知識の自動生成

設計ガイドで利用する領域知識を以下の方法で自動生成できる可能性が有る.

(1) 同じ視点からの設計結果において，ある特定の設計エンティティと持つ関係の等しさから類推する.

例えば，ある事例では，　[出庫処理;機能項目¦出庫依頼を受け取る]

別の事例では，　[出庫処理;機能項目¦出庫依頼のメールを解析する]

但し，[キー;視点¦内容を構成する設計エンティティ].

この場合，"出庫依頼を受け取る"，"出庫依頼のメールを解析する"は概念的に近いと考えられる.

本手法の問題点は，特定のエンティティを特定する方法である．この例では，出庫処理が供に存在したから簡単であるが，もし，違う場合，別の設計ビュー等から得られた概念的な近さ情報を利用する方法が考えられる．しかし，概念的に近いものの組み合せとして可能なものの数が膨大になり，情報量がほとんどない可能性がある.

(2) 設計ビューにおけるis-a関係を用いる.

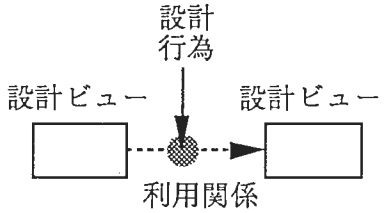
これは，必ず記述されるとは限らない．従って，is-a関係の記述を義務づける方法が考えられる．例えば，事例ベースにない設計エンティティを定義した場合，領域知識にその設計エンティティを登録させるか，または，その問題種別，機能種別などの視点からの設計ビューの作成を要請するなどの方法が考えられる.

何れの方法も要検討.

11. 読解性向上

DNAの読解性を向上させるための方策として以下の案がある.

現在，DNAとして設計の戦略は記録していない．これは，下図に示す通り，今利用関係として記録している情報には，本当は設計行為が介在している(つまり入力設計ビューから出力設計ビューを設計するアルゴリズム＝設計戦略).

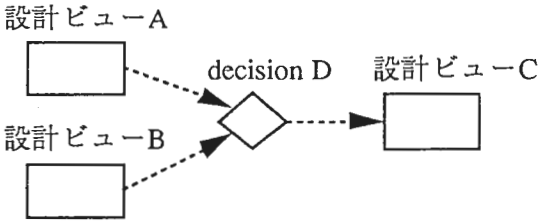


これは，関連する研究の所でも述べたように，以下の理由による.

記録情報が多ければ，記録した設計プロセスの読解性は向上するが，記録のコストが増える．記録する情報は選択されるべきである．ソフトウェアの保守で大事な情報は，すべてのソフトウェア設計で共通する一般的な情報ではなく，そのソフトウェア固有の情報である．また，保守者は，すべてのソフトウェア設計で共通する一般的な情報，即ち設計戦略的な知識は知っていると考えられる．よって，設計行為の詳細についての情報は記録して残す必要はない.

しかし，設計の内容が特殊であり，後で設計プロセスの記録を見る人は分かりにくいだろうと設計者が考える場合がある．このような場合を想定して，設計者が，分かりにくいことが予想される箇所に解説を付ける機能が考えられる．以下は，ハイパーテキスト技術を用いた設計のdecisionのメモ機能について概説する(本機能は，研究としての関心度が低いため検討を中断した).

Decisionを記録する方法



設計ビューA, Bに基づく設計判断(Decision D)により，設計ビューCを設計した.

(1) 新しい 記録項目

今までDNAで記録していた情報の他に以下の項目からなる情報を記録することにする.

- decision --- ある 設計対象の 性質から，ある設計上の判断decisionに基づき， 別の 設計 対象の 性質を設計するというモデルにおけるdecisionを記録する.
- used relationship type --- decisionが、 新しい 設計対象の 性質の 設計に対して 果たす 役割を 表す。以下の 2 種類が 考えられる。
  - 設計 エンティティの 存在 理由
  - 既存 エンティティ 間の 関係の 存在 理由

(2) used relationship type の 判別

- 設計 エンティティの 存在 理由

ある 設計 ビューの 設計に おいて、 今まで 作成 されて いない 設計 エンティティが 入力 された 場合、 その 設計エンティティの設計で利用した設計ビューは、 その 新しい 設計 エンティティの 存在理由を表していると考えられる。 1 つの ソフトウェアの 設計に おいて、各エンティティは その 存在理由を 1 組だけ 持つ。 ここで、 1 つの 設計 ビューを 設計する 際に 同時に 参照した 設計 ビュー全てを 1 組の 設計 ビューと 呼ぶ。

- 既存 エンティティ 間の 関係の 存在 理由

設計 エンティティの 存在 理由以外の利用関係は，すべて設計エンティティ間の関係が存在する理由を表していると考えることができる.

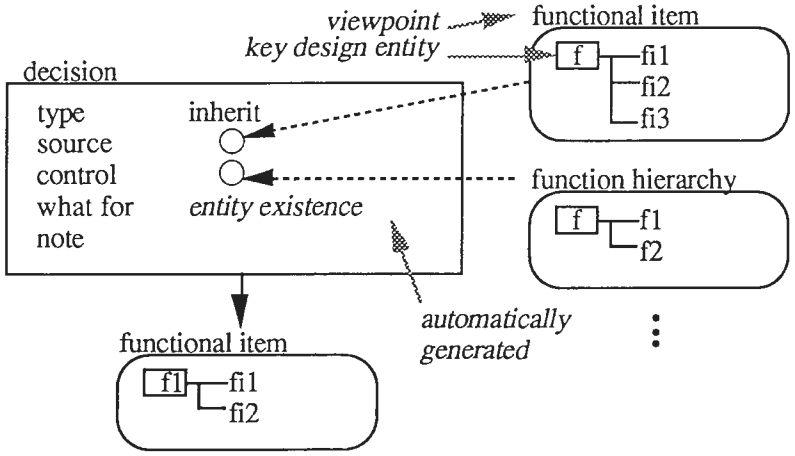
(3) Decision

- 設計ビュー の decision に おける 役割 (要検討)

Decisionは，設計ビュー に基づき行われる．その際の，設計ビューがdecisionに対してどのような 役割を果たすかを検討する。

- 継承 (割り当て) [継承 対象 : source, そのための 判断 材料 : control]
- 要求 -> 実現 [要求 項目, そのための 制約 事項]
- 必然性 (帰結) [因]
- 評価 [評価 項目]
- その他 (free format)

(4) 具体例



図中 decision の タイプとして 継承 (inherit) を 設計者が 指定すると、 inherit型decisionの テンプレートが 開く。 テンプレートには source, condition を 入力する フィールドが ある。 本例では、 source として、 f という 機能の 機能 項目を 、 その 機能の 構成 要素である f1 が 引き継いだ 設計を 表している。 その 引き継ぎの 条件として 利用した 設計対象の 性質として f の 機能構成 等が ある。 what for というのは、 前述の used relationship type を 表し、 設計 された 設計 ビューの 中の 設計 エンティティが 新規性より 自動的に 判断する。

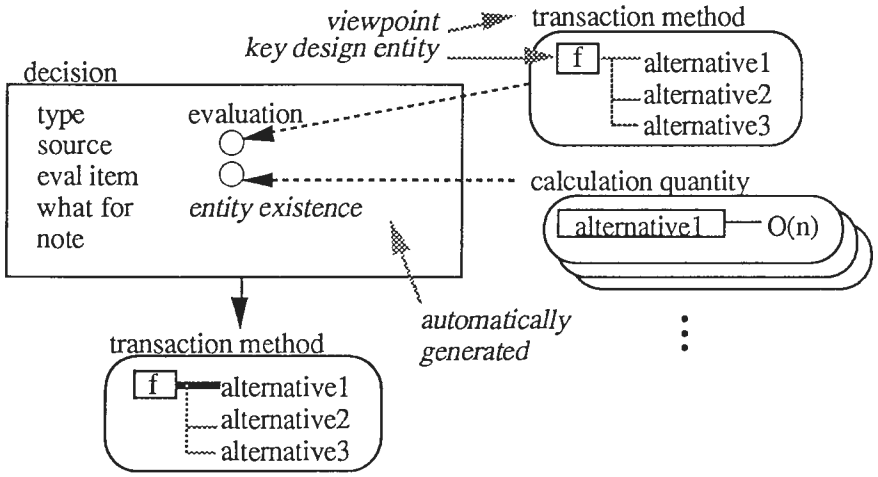
(5) 新しい 情報を 利用した 影響 波及 解析 (未検討)

設計対象の 性質が 修正 された 場合、 影響を 受ける decision の 検出が 可能となる。

例えば、 inherit 型の decision において、 condition となっている 設計対象の 性質が 変更された 場合、 source から inherit されている 項目を 見直す 修正を 行えば 良いことが 分かる。

(6) 応用

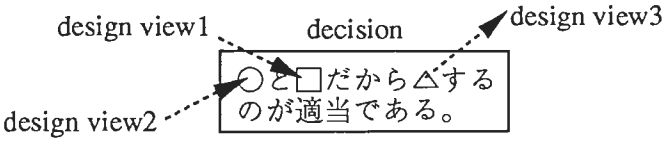
- alternative における選択の理由を記述する手法としてdecisionを用いることが考えられる.



(7) その後の検討

Decisionを記録するか否かは記録者の判断に委せる。つまり、ノートとして残すことができる仕掛を提供するという立場を採る。

その際、設計対象の性質は、Decisionを構成する要素となる。ハイパーテキストで考えると、Decisionを構成する文章に登場する1つのtagとなる(下図)。


design view1
decision
design view3
○と□だから△する
のが適当である。
design view2

6. 今後の課題

1. 当面

研究のテーマは，ソフトウェア設計における事例ベース推論技術の確立，即ち，設計知識の

1) 獲得方法

2) 利用目的を考慮した体系化，汎化方法

3) 利用方法

からなる．テーマ自身が技術的な視点からのものであり，対象とする問題領域に依存しない一般的な技術の確立を目指している．一方，適用領域を絞った具体的な利用技術として，

4) 通信ソフトウェア開発，保守支援における適用法

が存在する．

当面の研究の方向としては，以下の2通りが考えられる．

a) 今まで確立した手法をベースに4)を進める．

b) 1), 2), 3)それぞれの技術について更に発展させる(事例ベース推論技術を用いたソフトウェア設計支援).

a) 通信ソフトウェア開発，保守支援における適用法

一般に行われている方法は，設計やサービス要求に対する機能的な要求をどう書くかという技術である．しかし，現実にはそれ以外の要因が多い．通信ソフトウェアで言えば，サービスの振舞い以外の要求をどう扱うかという問題に相当する．このような研究はソフトウェア工学などでは一般にnon-functional requirement と呼ばれている．しかし，研究はほとんどされていない．サービスの振舞いは状態遷移モデルで記述でき，自動化できる部分も比較的多い．しかし，非機能的な要求についてはまだ未整理であり，今後の課題となっている．事例ベース推論技術を，非機能的な要求について適用するのが研究の進め方の案の1つとして考えられる(注: ATRでは．以前非機能的な要求について研究動向の調査を行っている---テクニカルレポートTR-C-0069, Software Design and its Automation Final Report, T, Gleeson, 3.10.19).

我々の手法でいえば，設計者が設計中に着目した設計対象の性質は，設計上考慮する必要があったものであるから，これは，非機能的な要求との関連が深いと考えられる．

特定な問題領域への利用技術の確立は，ドメイン分析を行うことにほかならない．通信ソフトウェア設計について，非機能的要求の分類，体系化や，非機能的要求と機能的要求の関係のモデル化を行う．具体的には，応用先として，要求分析(サービス要求を獲得する)工程か，次のSDLへの展開する工程が考えられる．

- 要求分析では，STRを再利用する際(要求理解等)，非機能的要求を制約として用いて候補の絞り込みに利用する等.
- SDL設計工程では，サービスの振舞いだけから決らない箇所(内部信号や物理的な装置の割り当て等)の設計における知識の構造化，獲得，利用の各方法の確立が考えられる．

(b) 事例ベース推論技術を用いたソフトウェア設計支援

研究テーマの進捗は以下の通りである．

設計ガイド方式における

① 視点利用規則に基づく設計目標の提示，管理

思想，基本技術は完了，詳細技術はほぼ完了．残るはインプリメントと評価である．

② 設計プロセスの再利用

思想，基本技術は完了．詳細技術，インプリメント，評価が残っている．

今までは，設計プロセスを再利用し，設計プロセス記録コストの削減および質の安定化を図ることを目的として行ってきた．今後は，更に設計支援としての側面を強めていく方向が考えられる．設計支援としての側面を強めるには，設計プロセスの最適化という重要な課題を扱わなければならない．一般に，設計のプロセスを再利用するからには，新しい設計状況に最適なものに修正するための技術が必要となる．我々のアプローチの考え方では，再利用する設計プロセス情報は，設計対象の性質に関するものだけしかなく，設計の手順に関する情報は入っていない．従って，設計のプロセスが最適であるか否かといった問題は生じ得ない(これは，影響波及解析を目的としていたためにである)．しかし，逆に，最適化コントロールが必要な，設計のプロセスに関することは，何も情報がないので設計者が考えなければならない．設計を支援するためには，今まで扱ってきた次元と別の次元の知識である，設計戦略(設計のコントロールに関する情報)が必要である．一般に行われている設計プロセスや設計知識関連の研究は，設計対象の性質の依存性を取り除いた汎用的な設計戦略知識を表現することを目的としたものが多い．しかし，これらの研究成果を見る限り，設計対象の性質への依存性を取り除けるレベルでの設計戦略知識は具体性がなく，あまり利用価値がない．従って，この両次元の知識を供に扱うことができる設計知識の技術が必要である．また，知識のモジュラリティを高めるため，これらの知識は有る程度の独立性を持ち，依存性はお互いの制約という形で表現できることが望ましい．従って，研究の方向として，設計支援を目指すため，今のDNA(設計プロセスのデータモデル)に対して，設計戦略的な知識情報をどう位置付け，表現，コントロールするかについて研究を進める必要がある．設計対象の性質と設計戦略知識がお互い制約として働くような表現が可能になれば，大変有益な研究になると考える．

もう一つの方向として，リバースエンジニアリングとの結び付きが考えられる．これは，関連する研究でも述べたように，既存のリバースエンジニアリングの技術は，プログラムの抽象化技術，即ち，ソフトウェアの全体を見るために必要な情報をソースコードより生成するための技術でしかなく，リバースエンジニアリングという言葉本来のニュアンスである，「設計技術上のノウハウを暴く」というところまで行っていない．これに対して，設計プロセス情報を用いて，設計ノウハウ的な情報を暴く技術を確立するという方向が考えられる．実際には，設計プロセスを記録しながら設計を行うという方法論との関係から，

「設計の飛躍を設計プロセスのデータベースを用いて補完する」

技術の確立として一般化することができる．設計プロセスを記録しないで設計したソフトウェアに対してはリバースエンジニアリングの技術として，設計に影響を与えた要因などをコンピュータが推測し，保守などを支援する．また，設計プロセスを記録しながら設計する場合は，設計プロセスを細かいステップで記録しなくても，コンピュータが抜けた部分を推測して補ってくれるという技術である．

2. 将来

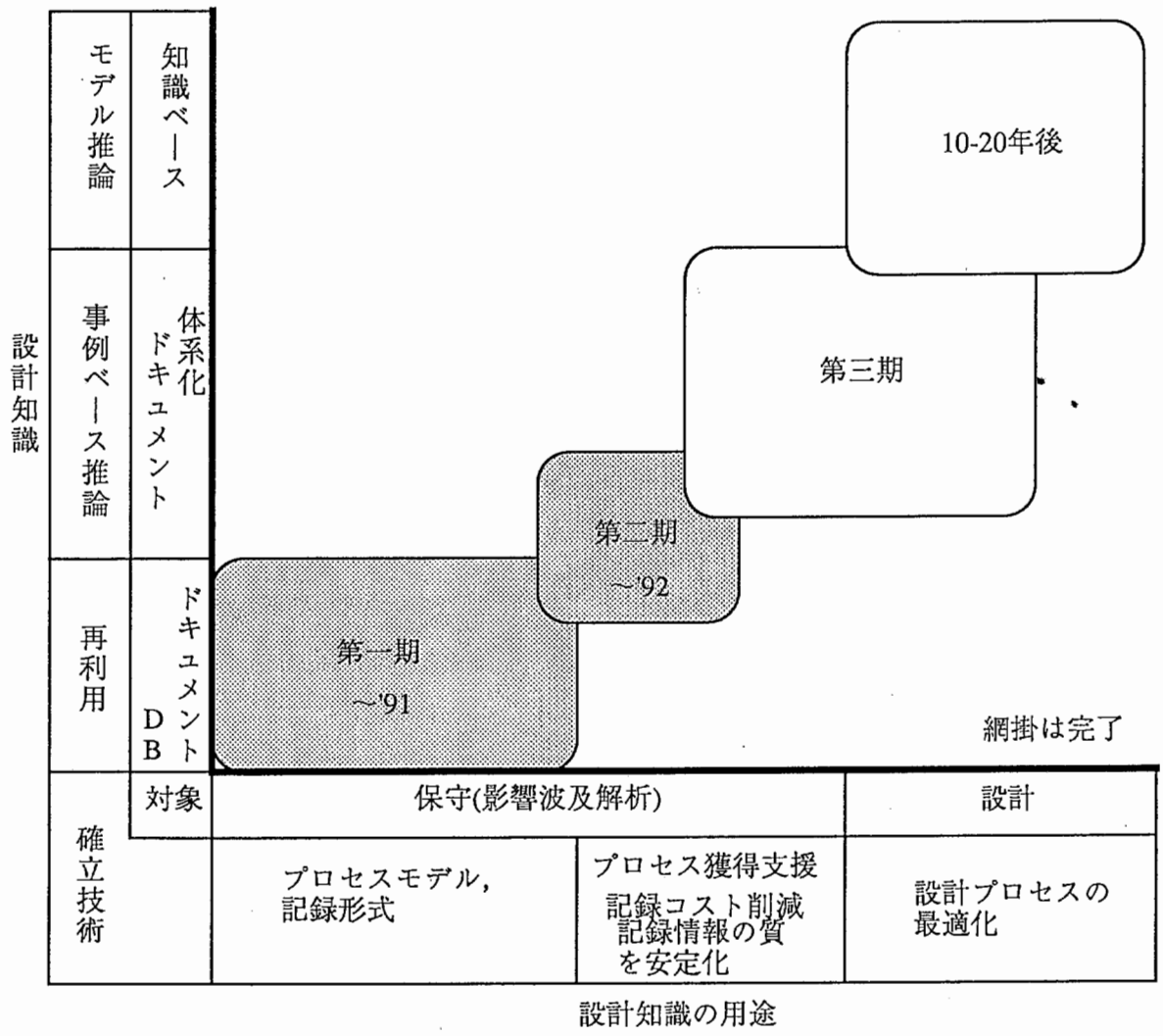


図1　研究の方向

研究の目的は，1) 設計知識の獲得，表現方法と2)その応用方法の確立にある．1)は図1の縦軸に，2)は横軸に記述した．図中の角の丸い四角が，研究の領域を示している．現在，網掛をした部分までが完了している．

1) 設計知識の表現の段階は大きく分けて3段階有る．一つは，ドキュメントDB段階である．これは，人間が読まないと内容が分からない．コンピュータの支援内容は，ドキュメントの格納，検索である．これに対してコンピュータが理解可能な知識ベースの段階が考えられる．コンピュータは知識を利用して演繹が可能になる．これは，設計や保守の自動化を意味する．また，これの中間的な段階として，体系的でかつ整理されたドキュメントである．変数化(テンプレート化)などにより，一部計算機による計算が可能であること，また，蓄積情報の重複がないなどの点がドキュメントDBに対して進んでいる所であるが，以前，人間が読まないと分からない情報が中心である点が知識ベースに対して劣る所である．

2) 設計知識の利用対象としては，今まで研究を進めてきた保守(影響波及解析)およびそのための設計プロセス記録支援と，新しいアプリケーションとして設計支援がある．設計支援については，当面の課題で述べた．

今後は，図中に示した第三期さらには10-20年後を目指した研究が考えられる．