

ソフトウェアクリエーション： 交換接続制御プログラムの設計知識

大森麻理† 馬場健†† B. H. Far 河野善彌

埼玉大学工学部情報システム工学科

† 現 株式会社 東芝 †† 現 株式会社 日立製作所

〒 338 埼玉県浦和市下大久保 255

E-mail : {far,koono}@cit.ics.saitama-u.ac.jp

あらまし

電話交換のためのスイッチ網を制御するプログラムの設計の知識を報告する。まず正常動作の設計から抽出した知識として上位の呼のプロセスから発着信を分離した複数プロセスへの分割、状態の差によるタスクの発生、及びタスクの詳細化について、次に準正常動作を自動付加するための知識及び評価について述べる。設計繰り返しにともなう知識数の増加には習熟効果がみられた。さらに自動生成を支援する機能として、人手介入のためのチェック、上位情報への補完機能について報告する。現在知識抽出、及び自動生成システムがもつべき外部機能の洗いを終えており、これらを Expert System 化し自動生成させることが今後の課題である。

キーワード 設計知識, ソフトウェア自動生成, 交換制御ソフトウェア

Software Creation : Design Knowledge for Design of A Telephone Switching Control Software

Mari OHMORI, Takeshi BABA, B. H. Far, Zenya KOONO

Dept. of Information and Computer Science, Faculty of Engineering, Saitama Univ.

255 Shimo-Okubo, Urawa 338, Saitama

E-mail : {far,koono}@cit.ics.saitama-u.ac.jp

Abstract

This paper reports on acquisition of design knowledge for a switching control system. The design procedure consists of splitting a call process, adding tasks and adding events. In each phase design knowledge is acquired by comparing the documents of the two consequent phases. Next the test algorithm for finding missing or abnormal routes in the state transition map and also the partial check of the design are added on. Thus acquired design knowledge and maintenance algorithms will be implemented in an expert system for switching software design.

key words design knowledge, automatic software design, switching control software

1 はじめに

巨大な交換システム等のシステムソフトウェアも根本は機能数、規模の年率 10-20%程度の増大に原因がある。その増加部分は既存技術の繰り返しといえる。そこで我々は人の設計に倣った設計知識再利用型のソフトウェアの自動生成を研究している [1, 2, 3, 4, 5]。本報告では、状態をもつ系の代表として研究している交換機の接続制御プログラムを設計する人の知識と、その実現のアルゴリズムについて報告する。

さらに実用性を考え、人手介入による変更に対処するチェック知識をも考慮した。これについては 5 章で述べる。

2 前提

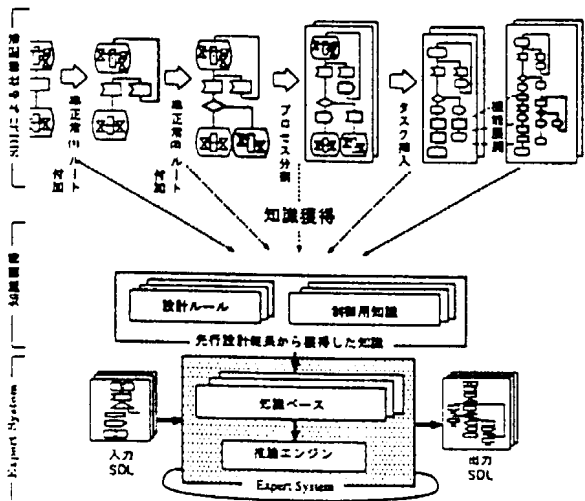


図 1: 本研究の方式

設計は図 1 上部に示すようにまず概略決定にはじまり、徐々に詳細化されていく。各工程間でどのような処理が行なわれたかは図面間の差をとることによりわかる。この差を設計知識とする。我々の研究対象とする自動化システムでは、エキスパートの設計結果から獲得された知識を図 1 下部に示すように Expert System に保存し、再利用する。

次に、仕様記述には ITU の仕様記述言語 SDL [6] を使用する。SDL は図的表現/テキスト表現が互換であるため人間、機械の両者とも親和性が高い。また、CASE Tool により C, CHILL, Ada 等へのコード変換が可能である。

交換システムとしてはデジタル方式を想定し、設計対象には信号プロセスに囲まれた接続制御ソ

フトウェアのみをとりあげる。また、発着信分割方式をとる。

3 正常機能の設計アルゴリズム

3.1 サービス仕様から話者毎プロセスまで (発着信分離)

仕様は、図 2 左に示すように、呼に関係する全ての情報が示される呼中心の記述で与えられる。これだと関連する全体がみえているために設計しやすい。設計時には徐々に詳細化を行なうが、その第一段階として、呼レベルの単一プロセス (P0) から話者毎の複数プロセス (P00, P01) への分離がある。

エキスパート (人間) はこの変換を一度に行なってしまうが、機械では難しい。調査・検討の結果、実は人間は三段階の変換処理を並行して行なっていることがわかった。図 2 に示すようにそれぞれの処理は単純かつ独立であり、順に適用すればよい。それぞれについて述べる。

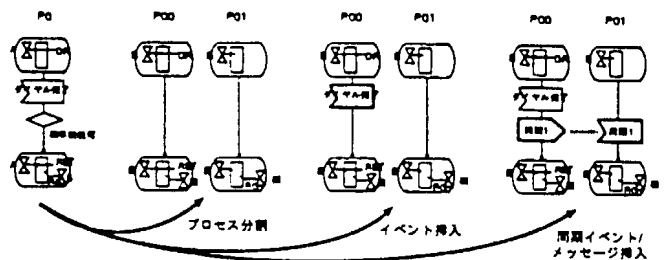


図 2: 発着信分離のながれ

3.1.1 プロセス分割

加入者毎のプロセス (P00, P01) を用意し、状態に関する情報を振り分け、分割する。単一プロセス (P0) の各状態における加入者、資源の情報を調べ、必要なもののみを該当プロセスにコピーする。分割後の状態にはそのプロセスが目指す加入者自身、及びその加入者と直接接続されている資源についての情報のみが記述される。それ以外は冗長な情報と判断し、含めない。必要ならば初期状態となるべき状態を用意する。

3.1.2 イベント挿入

単一プロセス (P0) 中の各イベントについて受け取る加入者を調べ、該当するプロセスの該当位置にそのイベントを挿入する。しかしこのままでは

他のプロセスにイベントのない遷移が残っており、プロセス間の状態遷移に同期がとれない。そこで、さらに次段階の処理が必要となる。

3.1.3 同期イベント/メッセージ挿入

同期をとるためにイベント/メッセージを挿入する。前処理でイベントが挿入された加入者プロセスから残りの各プロセスへイベント/メッセージを送信し、状態遷移の同期をとる。したがってイベント送信、イベント受信の組を対応する位置に挿入することになる。

3.2 ネットワーク状態の遷移(タスク挿入)

連続する二状態間の遷移内容を実現させるために、必要なタスクを発生させる設計工程である。例えば遷移によって信号音が“呼出音”から“話中音”に変わるのであれば、実際の設計時にはその間に“音種切替”に相当するタスクを挿入する必要がある。この工程について説明する。図3に概略を示す。各資源毎にルールを表形式で用意することができる。この表から遷移前、遷移後の資源の変化により対応するタスクを求め、挿入する。

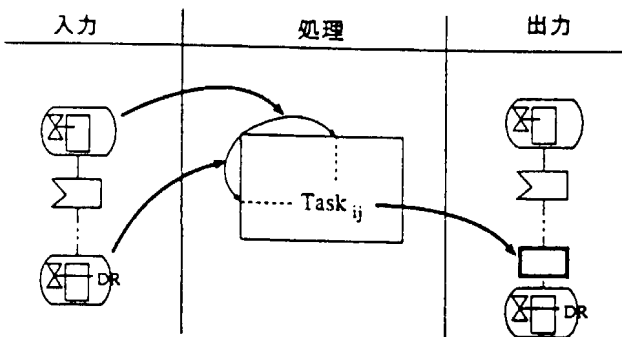


図3: タスク挿入

3.3 機能展開ルール

抽象度の高いタスクをより具体的なタスク群に変換する詳細化過程である。図4にルール例を示すように、これは単純な機能概念からより具体化された複雑な機能への展開である。極力単位的な知識として汎用性、柔軟性に富ませるためには、設計の小さな進行毎に設計知識をとることにするとよい。

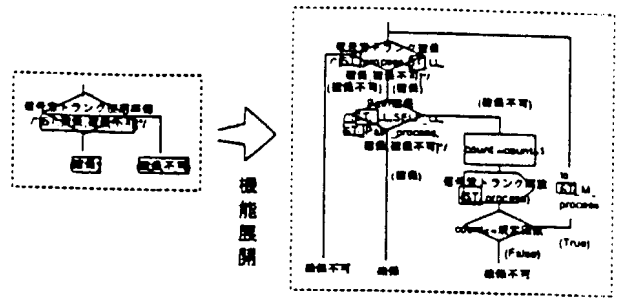


図4: 機能展開ルール例

3.4 設計知識の評価

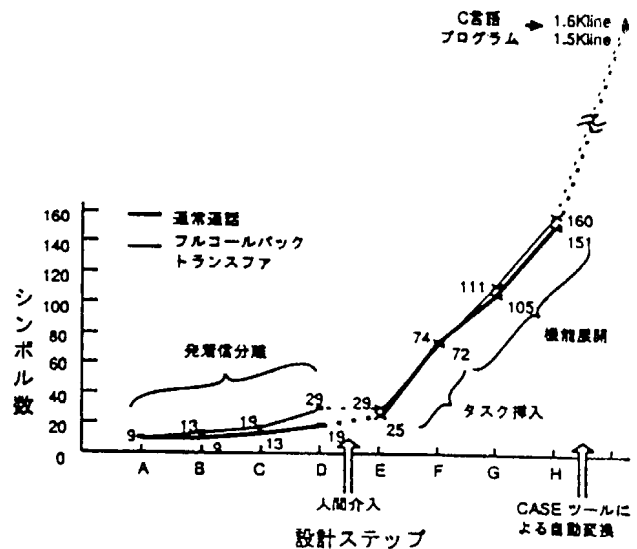


図5: 自動生成予測値

代表的な交換サービス5種を対象にルール抽出及び評価を行なった。

図5に、正常ルートの処理におけるSDLシンボル数の変化、及びコード化した場合のコード行数(予測値)を交換サービス2種の例で示す。ステップAは最初に与えられる仕様である。まずステップB,C,Dにより発着信を分離する。ここで分岐条件等を設計者が指定する(ステップE)。その後各設計知識の適用により仕様が詳細化され、SDLシンボル数が増加していく。

詳細化されたSDL記述はCASE Toolの自動変換機能によりC言語等への変換が可能である。

簡単なルールに分割することで、複雑な工程を誤りなく行なえるようにした。今回の試行では、タスク挿入については4種、機能展開の二段階についてはそれぞれ10, 6種のルールを抽出、使用した。

続いてサービス数が増えるときの設計ルール数の増加について考察する。図6に、設計サービス

数と使用ルール数の関係を示す。横軸は設計サービス数を、縦軸は使用したルールのサービス数を表す。ルールは資源毎に細かく区分して考えているため前述の数より多くなっている。また“内線自動転送準備”のようにあるサービスに特有なものを含めず、ルールとして有効であろう汎用的なもののみを集計している。K1~K5は、サービスを設計していく順序を変えた場合のそれぞれを意味している。

このグラフから次のことがわかる。サービス設計を進めるにつれ、はじめルール数は急激に増加するものの、やがて緩やかになる。両対数軸で示すと直線上になる。これは他の交換用運用プログラムの例 [3, 7] でも観測されている傾向であり、人の知識の習熟効果を示すと考えられる。

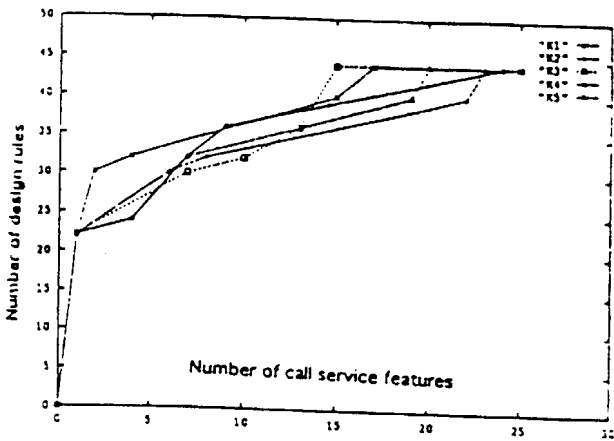


図 6: 設計サービス数と使用ルール数

4 準正常機能の設計アルゴリズム

4.1 準正常動作とは

交換システムのように対人インタフェースをもつ系では、前章までで対象としてきた正常動作以外に人やシステムの各種異常、正常でない状態に対処する機能が不可欠である。

図 7 は受話器を上げてから通話を終えるまでの正常動作のシーケンスを示す。実際のサービスでは、正常動作以外にも考慮すべき遷移がある。我々はこれを以下の二種類に分類した。

1. 加入者が途中で受話器を置く等、加入者自身に原因があるもの。準正常(1)動作と呼ぶ。
2. 相手が話中である等、内外部の資源捕捉失敗

によるもの。これを準正常(2)動作と呼ぶ。交換機は多くの加入者で資源を共有する方式をとるため、このように必要な資源がとれず失敗する場合もある。

これら準正常ルートは設計者の経験による知識にもとづき設計しているのが実態であるが、我々は機械による自動設計を目的としてこの知識がどのようなものであるか調べ、設計を簡単化する方法を工夫した。その結果、全体のみえる設計上流の状態図設計工程において準正常ルートを設計すれば、設計上流工程で全ての遷移を洗いだすことができる。したがって、下流の設計工程においては遷移内容の仕様は確定しているため準正常ルートについても正常ルートと同様に設計を進めることができるようになる。すなわち前節まで述べてきた知識を用いて自動的に詳細化することを可能とする。

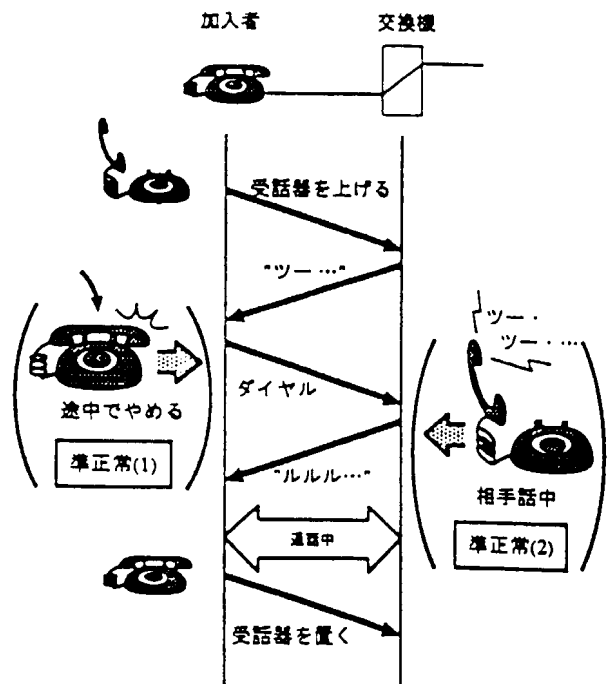


図 7: 準正常動作とは

4.2 基本的な処理方式

準正常ルートの洗いを準正常(1)、準正常(2)の順に二段階に分けて行なう。これは準正常(1)ルートの方が仕様の抽象度が高いからであり、正常及び洗いだされた準正常(1)ルート全てに対して同様に準正常(2)ルート洗いを適用することができるからである。

これら準正常ルート(遷移)の設計における問題

点を図 8 に示す、一般に接続制御においては全部で数十～数百の状態が存在するが、設計者はそれぞれに対して起こりうる全ての場合(数～二十)を考え尽くさねばならず、膨大な数になる。また従来は設計情報が複数の文書に分散して陰に記述されているため、経験にもとづく類推が必須となり、結果として準正常ルートが存在の見逃しが多発する。問題点をまとめると、

1. 設計の全段階において準正常ルートの付加の可能性がある。情報を陰に示す文書から確実に洗い出すための工数は莫大である。
2. 見逃しが発生しやすい。
3. 最終段階でしか見逃しに気づかず、その時には変更にも多大な工数が必要となる。

これらの解決策として、特定段階での全ルートの洗いだしが求められる。この実現のために設計者のもつ知識を分析しアルゴリズムを抽出する。

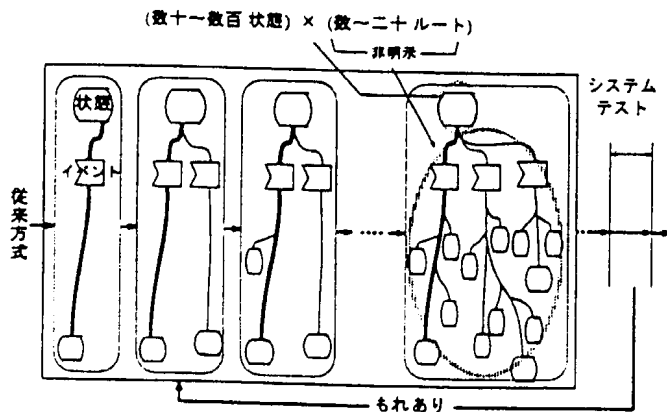


図 8: 準正常ルート設計の困難さ

4.3 準正常(1)ルート付加

準正常(1)ルート付加の概要を図 9 に示す。この処理は 3 段階からなる。それぞれについて述べる。

4.3.1 イベント付加

ある状態に対して考えられるイベントを全て付加する設計工程である。表を設け、加入者の状態毎にありうるイベントを全て記録しておく。設計時にはその時点の情報をもとにこの表をひく。結果として得られるイベント群のうちまだ記述されていないものが準正常動作を起こすイベントであるから、受信されるイベント群に追加する。

4.3.2 次状態作成

状態及び受信するイベントの結果として遷移する次状態を設計する工程である。ここで使うルールは表形式で用意される。表の要素として前状態がどんな状況にあるか、イベントを受け取る加入者の前状態における立場、受け取るイベントの内容が挙げられる。これらをもとに表をひくと前状態に対する遷移内容を得ることができる。その遷移内容を反映させた状態を遷移先状態とすればよい。

4.3.3 接続先候補の提示

前記により設計された遷移先状態と一致する既存状態をさがし、存在すれば一候補として提示する。以後既存状態に遷移させるか新たな状態として後続設計を行なうかの判断は設計者が行なう。どちらを選択した場合でも、後の詳細化は正常ルートと区別することなく、自動的に行なえる。

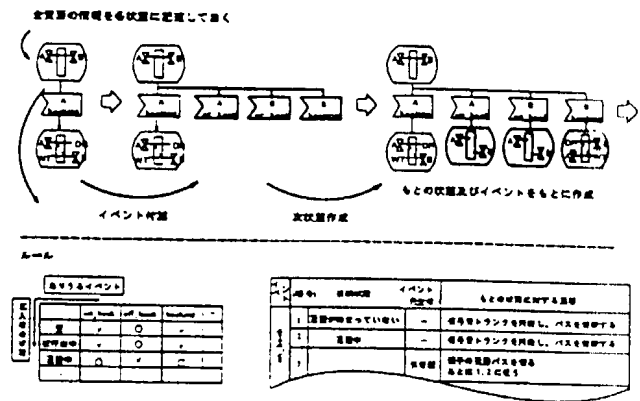


図 9: 準正常(1)ルート付加

4.4 準正常(2)ルート付加

続いて準正常(2)ルートの付加について述べる。これはある遷移の実現に必要な資源の捕捉に失敗する可能性のある遷移を洗いだし、失敗した場合の遷移を付加することである。この処理は 3 段階からなる。大まかななれを図 10 に示す。

4.4.1 資源塞がりルートの存在の明示

ある状態から次の状態への遷移の中に塞がりの可能性があるかないかを明示する。両状態について、記述されている各資源の遷移前後における資源状態の変化を調べる。新たに資源を捕捉する必要がある場合には塞がりが発生しうるため、設計

者に警告を発する必要がある。具体的には、表を資源の種類毎に用意して発生しうる塞がり内容を記述しておき、設計時にこの表をひく。

4.4.2 塞がり原因の列挙

熟練した設計者ならば、この段階でその後の処理となるべき設計の全てを考え尽くすことができる。ここで実際に発生する塞がりの原因をどのようにして知るかを考えると、前処理で用いた表に記されている塞がり内容を列挙していけば洩れなく洗いだせることになる。

4.4.3 その後の詳細設計

この後、洗いだされた各塞がり原因についての対処方法となるべき設計が必要である。そのためには各原因の評価順、塞がりに対する処理といった要素に対して重みづけを行い、最適なものを採らねばならない。これはシステムの外部条件や設計者の経験といったものに左右される。

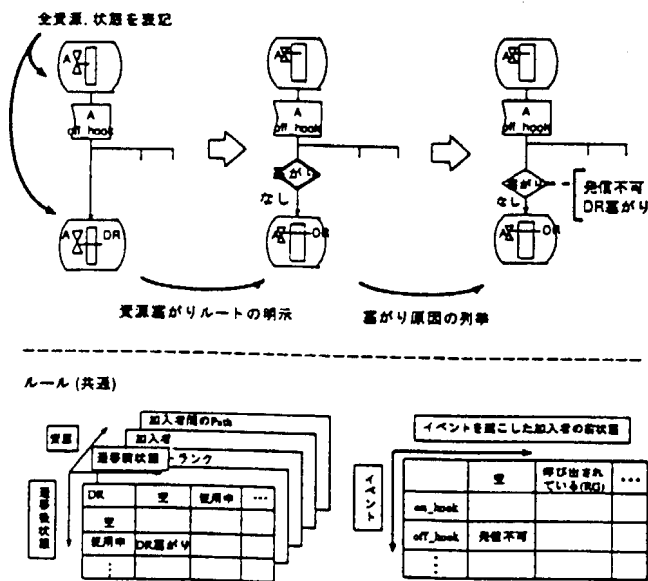
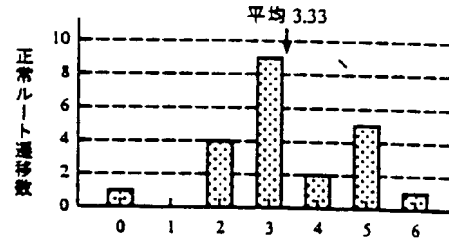


図 10: 準正常 (2) ルート付加

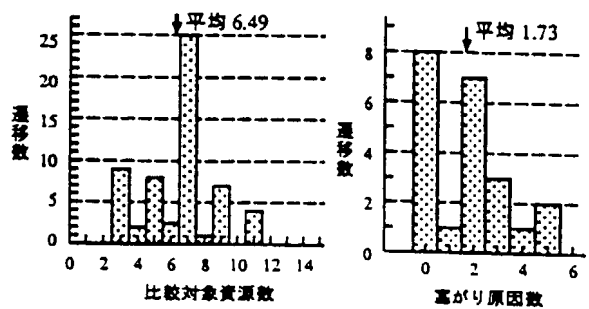
4.5 準正常ルート付加の評価

準正常 (1) ルート付加フェーズに対する評価を述べる。5種の交換サービスに対する設計試行を行ない前述の自動化がどの程度有効であるかを調べたところ、正常ルートにおける状態間の1遷移に対して、平均3.33の準正常 (1) ルートへの分岐が存在した。内訳を図11上部に示す。この処理を人

間が行なう場合、分散した文書中の情報を集めながらの処理となることもあり見落としが避けられない。それに対して、全情報を記述してある機械で網羅的に行なう今回の自動化法は、十分に有効である。



(a) 発生する準正常 (1) ルート数



(b) 準正常 (2) ルート付加処理の評価

図 11: 準正常ルート付加 評価

次に準正常 (2) フェーズの有効性を述べる。準正常 (2) ルートが存在するかどうか調べるためには、1遷移に対して平均6.49回の資源毎の比較が必要であった。これは単調作業の繰り返しであり機械処理に適している。人間が同様の処理を行なう場合は、誤りやすいためレビューが必須となり工数はこの倍以上となる。ところが自動化すれば、機械による単純作業であるから誤りは生じない。したがってこの工程は自動化が有効である。

また、この比較結果から導出される準正常 (2) ルートへの分岐数 (塞がり原因数) は、1遷移に対して平均1.73であった。

5 チェック機能の充実

人の設計には誤りが付きものである。よって人間介入により誤りが作り込まれる可能性がある。これを放置すると後続段階に拡大し発見/摘出が困難となる。そこでチェック機能も重要となる。

従来の方式の問題点を挙げる。従来のチェックは図12左に示すように“チェック工程”を用意して、

- このプロセス本体部に変更を加えたものを新設計情報とする。システム、ブロックレベルはどちらも同じものである。新旧両方の設計情報を本機能への入力とする。
- まず両プロセス本体情報を比較してどのシンボルが追加/変更されたのかを抽出し、それに対する処理を上位へ波及させる。

7 結論

設計上流から下流工程までの一連の作業における知識を抽出、整理し、適用できる見込みを得た。これまでの研究結果をまとめ、図 15 に概略を示す。

- 状態遷移図は徐々に詳細化されていく。
- 各機能はそれぞれ独立とする。
- 各機能終了毎に、独立なチェック機能を用意する。
- 上位階層への補完機能は各工程で共通のものを使う。
- 設計者から与えられる正常ルートの仕様に対し、まず
 - 準正常ルートの洗いだし、次に
 - 各ルートの詳細設計を自動処理で行なう。

各段階で用いる設計知識について述べる。

- 仕様が確定したルートに対する詳細設計を行なうアルゴリズムの確立、ルール抽出を行なった。このルールは正常/準正常の区別なく適用することができるものである。
- 準正常ルートを自動付加するためのアルゴリズムを確立し、必要なルールを抽出した。準正常(1),(2)ルート付加フェーズをそれぞれ試行し、この知識の有効性を評価した。
- 上位階層への補完機能について、主要アルゴリズムを確立し確認を行なった。

今後の長期的な課題として、並行する他の研究と併せた「知的 CASE Tool の実現」が挙げられる。

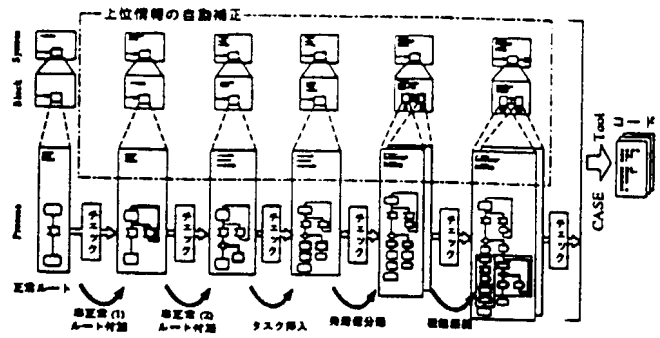


図 15: 自動生成システム概略

参考文献

- [1] Z.Koono, T.Baba and Y.Yabuuchi, "Software Creation: A Trial for Switching Software," 1992 Joint Const. on Communications, Networks, Switching Systems and satellite Communications. (1992,7)
- [2] 大森, ファー, 河野: "ソフトウェアクリエーション 交換接続プログラムの設計ルール", 1993年電子情報通信学会春季大会, B-525
- [3] Z.Koono, B.H.Far, T.Takizawa, M.Ohmori, K.Hatae and T.Baba, "Software Creation: Implementation and Application of Design Process Knowledge in Automatic Software Design", Software Engineering and Knowledge Eng. Conference (1993.6)
- [4] 河野, 馬場, 大森: "交換制御ソフト設計における人間による情報交換の研究 (通信ソフト開発エキスパートシステム化の基礎研究)", 電気通信普及財団研究調査報告書 No.8 pp.558-572 (1994.1)
- [5] 大森, ファー, 河野: "ソフトウェアクリエーション: 交換接続プログラムの準正常ルートの設計検討", 1994年電子情報通信学会春季大会, B-631
- [6] CCITT Recommendation Z.100, "Specification and Description Language", (1988)
- [7] H.Chen, B.H.Far and Z.Koono, "Software Creation: Reuse of Design Knowledge of Switching Software," Int. Conf. on Communication Technology, ICCT '94 (1994.6)