ソフトウェア工学　60－4
（１９８８．６．２２）

設計標準化に基づく部品合成システムの考察

加地　浩一　　松村　一夫
（株）東芝　システム・ソフトウェア技術研究所

ソフトウェア生産における、生産性と信頼性の向上を目的に、部品合成の研究を進めている。そのため我々は、部品化の考え／方針により開発された現場のアプリケーション・ソフトウェアに対し、部品合成の観点から分析を行った。その結果、このアプリケーションにおいては、状態遷移による設計の標準化が有効であることが確認できた。本稿においては、アプリケーション・ソフトウェアの分析をもとに、設計標準化を前提とするソフトウェア開発について述べ、さらに状態遷移をベースとした部品合成の自動化の可能性について考察する。

Consideration of a program synthesis system based on the standardization of design

Koichi Kaji and Kazuo Matsumura

Systems and Software Engineering Laboratory, Toshiba Corporation
70 Yanagi-cyo Saiwai-ku Kawasaki, 210 Japan

We are researching a program sysnthesis system based on program parts for the purpose of improving the productivity as well as the reliability in software products. For this purpose, we analized an application software, which is developed by selectively combining the parts prepared in advance. Consequently, as the software architecture of this application,we extracted a state-transition matrix. This paper describes a software development method based on the standardization of design, and the possibility of an automatic program synthesis which supports the state-transition matrix.

1．はじめに

ソフトウェア生産の工業化を目指すＩＭＡＰシステム[1] 開発の一環として、我々は部品化・再利用を推進している。部品化の目的は、アプリケーション・ソフトウェアの生産性と信頼性の向上にある。現在、自動化を最終目標に部品合成の研究を進めている。

この研究において先ず、医用画像処理装置の１つ超音波診断装置の一部分を成す画像計測ソフトウェアに対して、部品化の試行を行った。さらに今回、この実現されているアプリケーションに対して、部品合成の観点から考察を行った。その結果、状態遷移による設計の標準化が有効であるとの結論を得たので報告する。

２章においては、背景について、３章においては、部品合成の考え方について述べる。４章においては、アプリケーション“超音波計測”の特徴を、５章ではその分析について述べ、６章においては、分析に基づき設計標準化を指向した開発案を述べる。７章においては、部品合成システムの考察として自動化の可能性などについて述べる。

ＩＭＡＰ：Integrated software MAnagement and Production support system

2．背景

部品合成は事務処理分野（バッチ処理）においてかなり進んでおり、それらの多くは汎用性の高い“標準処理パターン”を予め準備することにより、合成を支援している[2] 。我々は、画像計測ソフトウェアに対して、ＩＭＡＰの部品化の考えを適用した。つまり“最初に部品ありき”の考えに基づき、アプリケーション開発に先立ち部品の作成・整備を行い、その後アプリケーション開発を行う、というものである。今回このアプリケーション（タスクレベルを想定）に対して、部品合成の観点から分析を行った。その方法としては、事務処理系の考えが制御系に応用できないか、つまり事務処理系でいうところの“処理パターン”に該当するものが制御系において存在しないか、という調査・分析である。実際分析を行ったところ、タスクレベルを覆う汎用的な処理パターンの検出は困難であった。しかし汎用的な設計の枠組みとして、状態遷移による方法が有効であることが確認できた。そこで、この分析をもとに考察を行う。従って本稿の目的としては、設計標準化を前提とするソフトウェア開発について述べ、さらに自動化の可能性について考察することである。なお、実現されたアプリケーションにおいても、状態遷移の考えに基づき設計されていたが、本稿ではこの考えをさらに高めた。

3．部品合成の考え方

我々の最終目標は、アプリケーション・ソフトウェアの部品自動合成であるが、その本質は“設計標準化”にあると考えている。設計過程において用いた設計情報や部品は全て再利用対象と考えられるが、再利用を活性化あるいは合成を自動化するためには、設計情報や部品を蓄積する枠組みを標準化・形式化する必要がある。そこで我々は次の枠組みをベースとし、さらに設計情報を蓄積する枠組みを考慮した部品合成方式を考える。

i) 設計の枠組み（ソフトウェア・アーキテクチャ）
ii) トップダウン設計法
iii)データ抽象設計法

まず“設計の枠組み”であるが、これは設計の標準化を狙うもので、例えば状態遷移によるものがある。これは、例えば対象分野のアプリケーションが複数の類似のタスク群から構成される場合、それらタスク群の具体化において同一のアーキテクチャを適用することにより実現方式を統一する、といったような設計の標準化に用いる。ソフトウェア開発の作業過程は、一般に“分解”と“統合”からなると考えられるが、この作業過程の自動化において、標準化された設計の枠組みは有効である。

次に“トップダウン設計法”であるが、これはソフトウェア設計において汎用性の高い技法である。これを用いる理由の一つは、前記“設計の枠組み”適用により分割詳細化された機能をさらに分解してモジュールレベルまで詳細化するのに利用する、という点にある。また一般に前記“設計の枠組み”として適当なものが存在しない場合も考えられるが、その場合においてもトップダウン設計法が適用できると考えられる。さらにもう一つの理由として、アプリケーションのそれなりの性能をキープする、という目的がある。部品再利用を行う場合ボトムアップ支援は必要であるが、これを強調すると性能の劣化を起こす場合が考えられる。

最後に“データ抽象設計法”について述べる。抽象化により、部品の機能仕様部とその実現方式を分離することができる。またデータの型とそれに対する操作の一体化や、構造化によりデータ型を組み合わせた新しいデータ型構成の可能性がある。これらの性質により整理の指針が与えられ、再利用に有利な部品データベースの構築が可能となる。

以上の考えに基づき、アプリケーション“超音波計測”をもとに考察を行う。ここでは特に、設計の枠組みについての考察を中心に行う。

4．アプリケーションの概要

分析対象のソフトウェアは、超音波診断装置の一部を成す画像計測ソフトウェアである。この画像計測ソフトウェアは複数の計測機能単位（タスク）から構成されており、これらのタスク群は状態遷移による考えで標準化されている。我々の目標は、さらに形式化を図ることである。以下、問題を分かり易くするため、対象アプリケーション（正規のもの）を単純化して概要のみ説明を行う。

計測機能（タスク）としては、例えば次のようなものがある。

・面積トレース　：トレースROI の囲む面積・周囲長を求める。
ＲＯＩ…Region Of Interest（関心領域）

患者の疾患部の検査などに使う。
・ヒストグラム　：面積と同様の領域(ROI) 内の値の度数分布を求める。
・ＣＦＭ速度　　：プローブの向きを補正し血液などの流速を計算する。
ＣＦＭ…Color Flow Map

計測の各タスクから見た環境は次のように考えられる（図１参照）。

・実行単位としてのタスクは装置としての機能単位に分割されており、それらは並列に上位のタスク（計測コントローラ）の下に並ぶ。
・計測の各タスクが相手にするハードウェアはフレームメモリ（画像・文字・グラフ）のみと考えられる。

次に、計測タスクの１つ“面積トレース”における機能を簡単に説明する。即ち、機能、操作キー、会話イメージについて示す。

機能

ＲＯＩ（関心領域）の設定を行い、指定されたＲＯＩの面積と周囲長を計測する。計測は同時にｎ個（ＣＨ１～ＣＨｎ、但しここではｎ＝２としておく）まで可能である。

操作キー

面積トレースで用いるキーは次の通りである。
・AREA･TRACE　：面積トレースの指定
・CH1 ～CH2　：チャンネル１～２の指定
・COPY　　：ＲＯＩのコピーに使用
・ERASE･TRACE　：面積トレースの部分消去

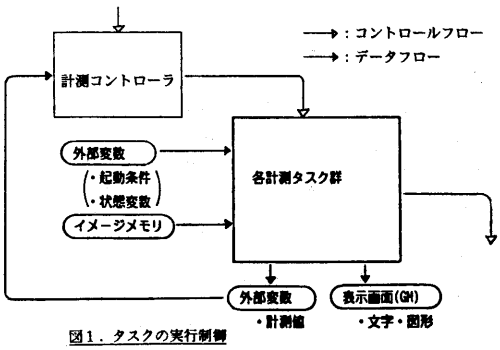


図１．タスクの実行制御

・ﾄﾗｯｸ･ﾎﾞｰﾙ　：マークの移動などに使用
・ﾛｰﾀﾘ･ｴﾝｺｰﾀﾞ　：チャンネル識別文字の回転に使用

会話イメージ

図２に、面積トレース計測の標準的な操作による会話イメージを示す。概要は次の通りである。

i) 計測すべきCHn の選択を行う。これに対応する処理として移動マーク“＋”を所定の位置に表示し、CHn に対応する計測表示エリアを消去する。
ii) ﾄﾗｯｸ･ﾎﾞｰﾙ によりマークをトレースの開始位置に移動する。
iii)CHn によりトレースの開始位置を指定する。
iv) ﾄﾗｯｸ･ﾎﾞｰﾙ によりトレースを行う。
v) ＲＯＩが閉曲すると計測結果を表示し、ＣＨ識別文字を表示する。ﾛｰﾀﾘ･ｴﾝｺｰﾀﾞ によりＣＨ識別文字の表示位置を指定する。
vi) COPYキーによりＲＯＩのコピー指定を行う。
vii)ﾄﾗｯｸ･ﾎﾞｰﾙ によりマークをＲＯＩのコピー先へ移動する。

５．アプリケーションの分析

対象ソフトウェアの設計仕様書や、部品化の試行により作成された部品に対して分析を行う。

以下、超音波計測の特徴、パターン化、（実際に具体化されている）モジュールの階層構造に対する考察を行う。

(1) 超音波計測の特徴

部品化の試行により作成された部品とアプリケーションに対する考察を行い、特徴を説明する。

i)　部品の特徴

部品化の試行により作成された部品について少し考察しておく。アプリケーションは、図２のような会話処理の仕様を明確に規定する部分と捉えることができる。部品は、各アプリケーションを実現するための計測処理や画面表示処理の機能群である。その他、フレームメモリ処理や基本部品からなる。例えば、次のような機能部品がある。

計測処理……トレースの周囲長を求める
　　流速値の平均値を求める
画面表示処理……トレースを描く
　　マークを移動する
フレームメモリ…フレームメモリの１点読み出し

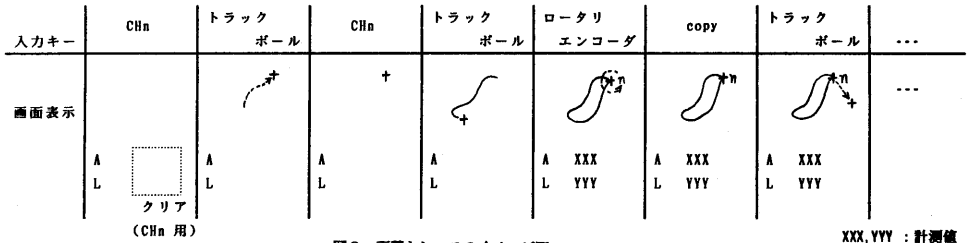


図２．面積トレースのイメージ図

基本部品……ASCII 文字列を表示する

ここでこれらの部品群に対し、再利用性・汎用性の観点で少し説明しておく。まず使用頻度の高い部品であるが、例えば“マークを移動する”とか“ASCII 文字列を表示する”がある。“マークを移動する”は、アプリケーションの性質上（アプリケーションの特徴のところで述べる）使用頻度が高く、汎用性を考慮して実現されている。“ASCII 文字列を表示する”は基本機能と考えられ、やはり使用頻度が高い。一方、使用頻度が低い部品としては、例えば“トレースの周囲長を求める”とか“流速値の平均値を求める”がある。これらは今回の開発においては１度しか使用されなかったが、今後のシステムのバージョンアップや部分的な仕様変更などが生じた場合に、有効となる場合が考えられる。また、これらを部品として準備した理由として、（例え１度であっても）使用されることが分かっていたことや、部品化により理解性・保守性が向上する、などがある。つまり、ここでの部品化の試行においては、汎用性のありそうな部品については汎用性を考慮し、また汎用性はなくても理解性・保守性を向上させるために独立な機能単位と認められるものについては、殆ど全てにわたって準備した、と考えてよい。

ii) アプリケーションの特徴

アプリケーション側の作業は、予め準備された機能部品を各タスクの操作仕様を満足する形に繋げることで、例えば図２のような仕様を実現することである。この場合、これらのタスクの会話処理にはある手順があって、入力情報をもとにその手順に応じ段階的に処理が進められる。各タスクにおいては、状態フラグを用いて段階的な手順を制御している。即ちアプリケーションは、“常に、ある状態を持っていて、キーによる情報が入ってくると、各状態に応じた処理（計測および表示）を行い次状態を設定する”、という作業の集合体と捉えることができる。アプリケーション（タスク）毎に、計測内容そのものは異なるが、計測の範囲指定に用いるマークの使用方法（マークの形状などはタスクにより様々である）や、計測が２チャンネル可能であるため、その区別としてのチャンネル識別文字の使用方法については同じ思想で統一（標準化）されており、部品の共通化がなされている。

(2) パターン化

アプリケーションを局所的に捉えるか大局的に捉えるかによって、パターンの形態は異なる。事務処理系でいうところの処理パターンに該当あるいはそれに近いものは、局所的に捉えれば存在する。一方、大局的に捉えると処理パターンの抽出は困難であるが、設計の枠組みレベルのものとして状態遷移による方法が存在する。

以下、局所的パターン化と状態遷移表の適用について分析を行う。

i) 局所的パターン化

アプリケーションの局所的なパターン化は可能である。例えば、各タスクの初期化の処理に着目してみる。初期化の内容としては、計測結果のタイトルやマークを所定の位置に表示し、画面の表示エリアをクリアし、各種フラグや変数の初期情報をセットする、などである。ここで、初期化の内容は個々のタスクにより異なる（マークの形状や変数の初期情報などが異なる）が、何をどんな順番で初期化するのかという処理の流れについては、共通化が可能である。この処理の流れを処理パターンとすることにより汎用に使用できる。一種のホワイト・ボックス部品である。

しかし、本アプリケーションの主要部分は図２に示した会話処理部にある。局所的パターンの蓄積も重要であるが、主要処理部を規定する枠組みを準備しない限り、大幅の改善は望めない。

次に、状態遷移表に基づく分析を示す。

ii) 状態遷移表の適用

アプリケーションの特徴のところで述べたが、各タスクは“目的の計測を、段階的な操作（会話処理）により行う”という性質上、各操作段階を保持するための情報を持っている。この情報は、一般に状態とかモードと言われる制御情報で、この情報がアプリケーションの操作仕様を決定する上で重要な役割を果たす。つまり、“特徴的な状態を準備し、各状態に対し入力キーを対応付け、それぞれの対応において出力処理および次状態を定義する”ための枠組みを設定することを考える。

状態遷移表[3] の適用イメージを表１により説明する。表の横軸に入力情報であるキーの列を並べている。縦軸には状態を並べている。それぞれの交点は実行すべき内容を記述する。例えばstate2の状態のときkey2の入力があった場合、action22を実行するという具合いである。ここでaction部に記述される内容としては、計測処理と表示処理および次状態の設定である。なお、この状態遷移表を駆動する制御用モジュールが上位に存在する。

(3) モジュールの階層構造に対する考察

部品化の試行によって具体化されている本アプリケーション“面積トレース”の階層構造図（図３）に対して、設計の標準化と部品化の観点で考察する。

図３の説明を行う。このタスクの場合、階層は大きく４つに分けることができる。即ち、最上段はタスクの制御モジュールで、その内容としてはイニシャル処理とエンド処理とユージャルの制御処理、およびデータ宣言部などを含む。二段目は状態遷移表により分割された機能群である。最下段は予め準備された部品群で、三段目は

表１．状態遷移表

状態＼入力	key1	key2	...
state1	action11	action12	...
state2	action21	action22	...
⋮	⋮	⋮	...

二段目の各機能群と最下段の部品群との“つなぎ”のモジュール群である。

まず、タスク構成の標準化の観点で考察する。最上段と二段目との関係は、ユージャル処理を状態遷移表を用いて分割詳細化を行っているため、すっきりした分割が実現されている。三段目は最下段の部品群を用いて、二段目の機能を実現するために準備するモジュール群であるが、標準化の困難な部分であると思われる。機能の分割併合や合併などのトップダウン的作業と、部品利用の立場としてのボトムアップ的作業からなると考えられ、発見的に実現されている部分である。この部分においても何等かの支援、例えば用語の標準化や部品への展開の指針を確立する必要がある。

次に部品化の観点で考察する。一般にモジュールは下位にある程、再利用の頻度が高い。部品の基準の一つの観点として、下位参照モジュールが存在しないモジュールは部品と捉えることができる。例えば階層構造図のtrachg、tracopy は、

trachg　：トラックボールの移動量を座標系に変換する
tracopy ：現在使用中のチャンネルＣＨｎのＲＯＩを次のチャンネルＣＨｎへコピーする

という機能モジュールで、これは本アプリケーションの実現方式に関わる内容であるが、本アプリケーションの部品として準備すべき機能と考えることができる。これらの機能モジュールは、予め準備した部品群の中に、それらの機能を意図した部品が用意されていたが、アプリケーション作成時において準備した部品は使用されず、改めて作成したものである。部品の再利用性を高めるためには、部品標準化の基準を明確にすると共に、設計の標準化が重要になると考えられる。

6．設計標準化を指向した開発手順（案）

３章で述べたように、自動化を指向した部品合成の前提として“設計標準化”を考えている。つまり設計の枠組み（ソフトウェア・アーキテクチャ）を準備することにより、アプリケーション開発の標準化・自動化を図る。設計の枠組みの適用方法としては、大きく２通り考えられる。つまり対象アプリケーションが新規開発で、既存ソフトウェアに再利用可能な部品が殆ど存在しない場合、もう一つは既存システムに類似のものがあり再利用可能な部品が存在する場合である。ソフトウェア・アーキテクチャの選択において、再利用可能な部品が存在する場合はそれらの部品群に合わせた選択が必要である。新規開発の場合はアプリケーションの将来性などの考慮が必要ではあるが、基本的にその問題に合ったアーキテクチャを選択することになる。

超音波計測の分析結果をもとに、設計標準化の枠組みとして状態遷移を用いたソフトウェア開発手順（案）を述べる。なお本開発手順の適用範囲としては、現在のところタスクレベル（構造設計レベル）を考えている。つまり、タスクを実現するにあたり、より下位のモジュール（サブルーチン、関数など）をどのような構成で組み上げるか、という部分を対象として考えている。大まかな手順は、次の５ステップからなると考えている。

step1)用語の標準化
step2)ソフトウェア・アーキテクチャの決定と、
　　機能概要記述
step3)トップダウンによる機能の分割詳細化

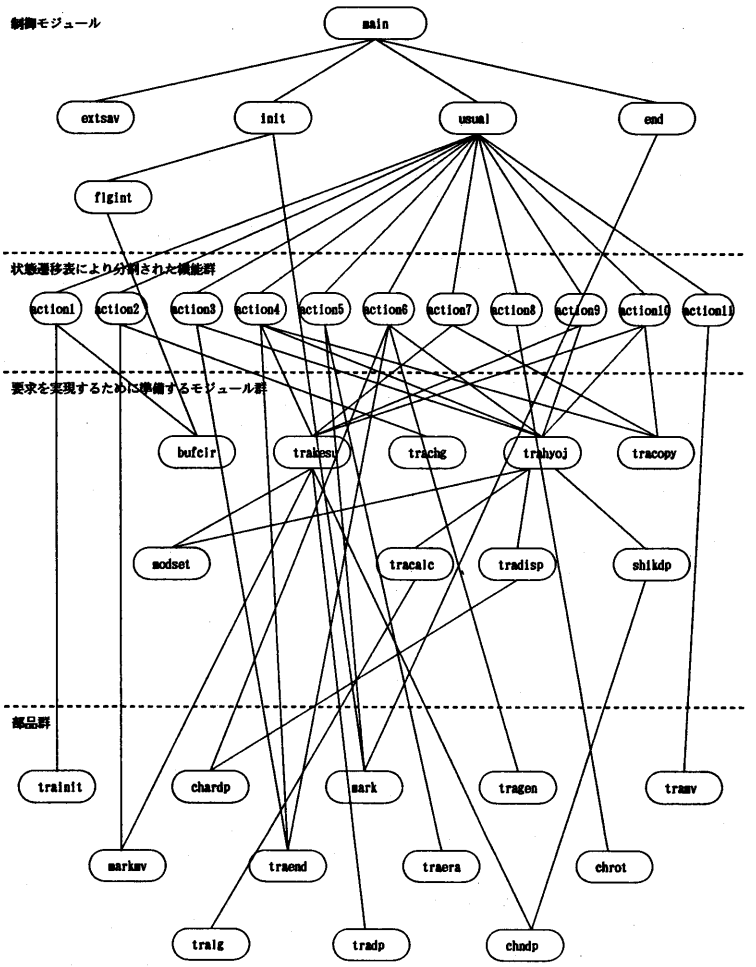


図３．タスク内のモジュール階層の例

step4)部品化の検討
step5)データ抽象化

以下、各ステップを超音波計測の“面積トレース”について、考察を行う。

step1)用語の標準化

対象分野において使用される用語の標準化を行う。このステップで標準化する用語は、設計の仕様記述、部品の機能仕様記述、データ名などに使用されるものであり、それらの記述に適切な用語の選択が必要である。

同じ用語であっても、対象分野が違えば解釈が異なる場合が考えられる。標準化は、分野を制限して行うべきかも知れない。

step2)ソフトウェア・アーキテクチャの決定と、機能概要記述

要求仕様を分析し、対象分野のソフトウェア・アーキテクチャの決定と、標準化された用語による機能概要の記述を行う。

本アプリケーションのソフトウェア・アーキテクチャとして状態遷移表を用いるが、その様子は表２の通りである。この場合、会話により処理を段階的に進めて行く過程において、特徴的な“表示状態”がいくつか存在する。そこで、表２のような入力キーと表示状態との対応からなる状態遷移表を準備し、標準化された用語を用いて機能概要を記述する。表２の各ボックス（入力キーと表示状態の交点のボックスのこと）には、各表示状態において各キー入力があったときに成すべき処理と次に遷移すべき状態設定の記述をしてある。ここでの記述に際しては、標準化した用語は意識するが、自由な記述でタスクの概要（機能および実現方式の詳細化の過程）が一目で分かることを目標とする。

この枠組みの効果としては、各ボックスに注目して機能記述が行えるため複雑さの分散や記入漏れを軽減することができる。また、表示状態に応じた入力キーという対応なので、実現システムでの会話をイメージしながら記述ができ、設計の思考を助ける。

step3)トップダウンによる機能の分割詳細化

表２の各ボックスを、標準化された用語をもとに必要機能を列挙する。

ここでは、step2)で得たタスクの概要をより詳細化するステップで、トップダウン設計法を適用する。即ち、表２の各ボックスの概要を分析し機能の切り出しを行い、切り出された機能の制御を明確に記述する。このとき、機能の切り出しにおいては機能の目的を明確にし、モジュールの独立性を向上させなければならない。図４に、表２のボックス（初期画面表示，ＣＨｎ）の内容を詳細化し、ＴＦＰ[4] により記述した例を示す。ここでの記述の指針としては、機能の目的を明確にするために操作名と対象名により表現する、を意識した。

ＴＦＰ: Technical description Formula for Procedure design

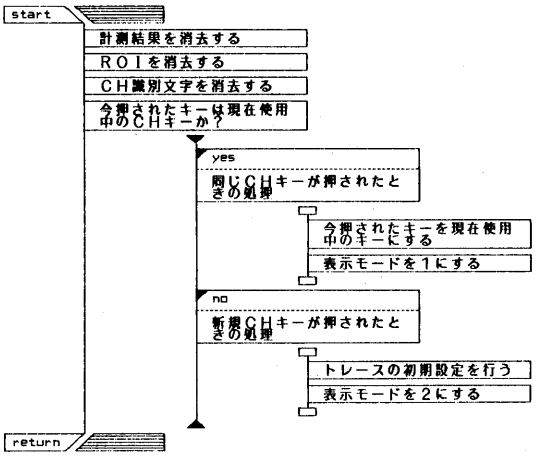


図４．ＴＦＰによる機能の詳細記述例

表２．面積トレースの状態遷移表

表示状態＼入力キー	ＣＨｎ	ＣＯＰＹ	ＥＲＡＳＥ
状態1.初期画面表示	計測結果，ＲＯＩ，ＣＨ文字の消去。 ＣＨキーは使用中か？ yes→状態２ no →トレース初期設定 状態１	dummy	dummy
状態2.トレース状態	トレースの強制終了と表示。 ＣＨキーは使用中か？ yes→状態３ no →表示消去 状態１	トレース強制終了と表示。 ＣＨの方向リストのコピー。 状態４	マークを消去する。 トレースした結果を一部削除する。 現在点にマーク表示する。 状態２
状態3.トレース終了状態	（状態１，ＣＨｎ）	今押されたキーの結果の消去。 ＣＨの方向リストのコピー。 状態４	
状態4.ＲＯＩの移動中			

・ＣＨｎ　　：ＣＨ１またはＣＨ２
・（　，　）：（表示状態，入力キー）に対応するボックスを表わす
・dummy　　：何もしない

step4)部品化の検討

再利用性を向上させるための部品化の検討を行う。

基本的には、step3)で得られた機能が部品と考えられるが、さらに再利用性を考慮した部品標準化の検討が成されなければならない。つまり、部品の分割併合や合併など、設計の発見的手法により機能やデータ、さらに部品のインタフェースの標準化の検討が必要である。

step5)データ抽象化

部品のデータ抽象化を検討する。

step4)で整理された部品群に対し、データ抽象化を基本とした部品の階層化を行い、メンテナンスや検索の容易な枠組みを検討する。このステップはstep4)と密接に絡み合っており、並行して検討する必要もある。

以上が各ステップの概要であるが、基本的にはstep1)からstep5)へ段階的に進めると考える。

7．部品合成システムの考察

前章で提案した開発手順に沿って作成・蓄積された部品および設計情報を用いて、再利用と自動化の可能性について検討を行った。

以下、自動化と設計情報の再利用について考察を行う。

(1) 自動化の考察

前章において、設計の枠組みとして状態遷移表による事例について考察した。ここでは状態遷移表を用いて設計を行った場合の、自動化の可能性について考察を行う。

表２のような枠組みを使用して詳細化を行った場合、その表を制御する制御モジュールを準備すれば、表２の下位において詳細化された情報（例えば、図４のような情報）との自動リンクが可能である。ここで制御モジュールの典型例は、ＴＦＰを用いて示すと図５の形をしている。即ち、前処理と後処理とがあって、その間に状態遷移表を駆動する部分として、入力イベントをチェックし、現在の状態と入力イベントから実行すべきアクション部の判定を行い、各アクション部に分岐する、を行う処理部からなる。

この図５の状態遷移を制御するモジュールの骨格を、もう少し詳しく検討してみる。

i) 状態遷移データの自動作成

表１または表２の状態遷移表の記述形式を定めることにより、状態遷移データを自動作成し、図５の“アクション部の判定”の処理は、その状態遷移データを参照して行うようにできる。この部分は一種のインタプリタとなる。

状態遷移の記述形式としては、入力イベント、状態、アクション部の処理、次状態設定などが書ける必要がある。

ii) 状態遷移の階層的な適用

システムの動きを設計する上で、状態遷移という切り口または観点が人間の思考を良く助けることは前に述べた通りである。しかし、どんな状態を設定するかは人間の仕事であり、それによっては設計がうまく整理されたり、複雑なものになったりする。一般的に、状態の抽象レベルがバラバラだったり、あるいはあまりに多くの状態を作ったりしてしまうと、理解がしにくくなり、設計として質の悪いものになる。そこでレベルを揃える必要がある。

例えば、状態遷移は、今回分析したアプリケーション“超音波計測”のようにマンマシン系を作成するのに適していると思われるので、これを例にとって状態遷移のレベルについて考察してみる。

まず、このアプリケーションのマンマシン系を設計するときに最初に重要なのは、イベント制御である。例えば、表２の作成作業である。つまり、大枠として、アプリケーションを構成する処理単位（例えば、表２のアクション部である各処理ボックス）は、どんな状態でどんなイベントが生起したときに成されるべきかが決定される。このとき、この状態遷移表の中で、例えばキャラクタキーを入力している途中で、別のイベント発生を促すキー入力があるというような場合、この詳細なことまで全てを一度にこの状態遷移表に書くと、非常に複雑になるであろう。そこで、ある状態から発生しうるイベント集合の決定までの入力列の制御を、別の状態遷移表に書くことを別途行うことが良いと考える。これは、一つ一つの入力キー毎に新しいより詳細な状態へ遷移し、前に決定したイベントの発生が起こるまでを記述することになる。

このように、イベント制御のための状態遷移と、入力列制御のための状態遷移に分けることで、設計を段階的に行えるし、また自動作成の観点からは、２レベル目の入力列制御の部分を自動作成したプログラムを、図５の“入力イベントチェック”という部分に埋め込むことにより、階層的にプログラムを作り上げて行くことも可能

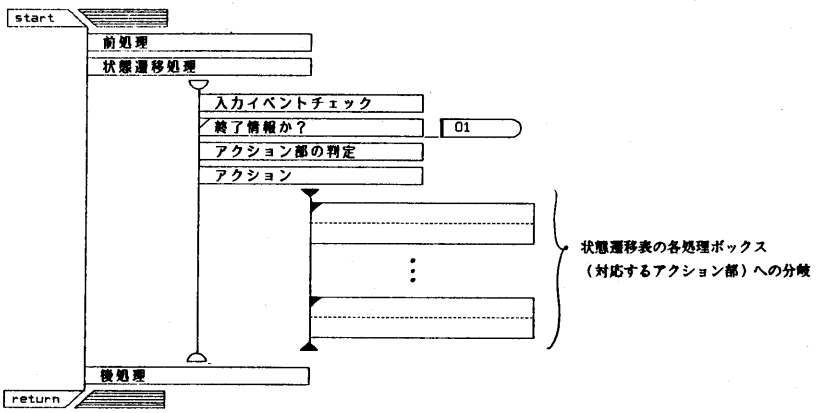


図５．状態遷移表を制御するモジュール

になる。ここで説明した例が、必ずしも適切かどうかは検討の余地は残るが、ある程度、汎用な部品合成ツールとしては、不可欠なものと考える。

iii)アクション部の部品作成

アクション部に入るべき部品については、各アプリケーション分野またはソフトウェア部品系列毎に事前に作っておいたものを、設計者が選んで状態遷移表に記述することになる。部品の作り方や部品化の観点における我々の検討については、文献[5]を参照されたい。また、部品名で記述するか、あるいは、より日本語文に近い形式で記述するかについては、文献[6]で考察しているのでこれに譲ることにする。

iv) プログラムの最適化ルール

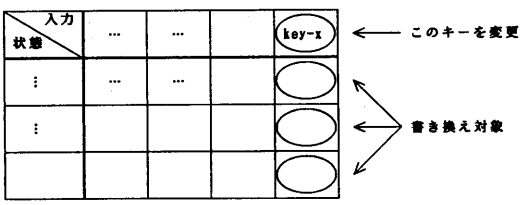
設計者は、状態遷移表を用いることにより、漏れなく全ての場合を検討し整理して設計することができる。しかし、実際にプログラムの処理フローを作るときには、さらに整理して単純な形にまとめていることも多い。例えば、図5において全ての条件分岐を作るのではなく、同じアクションをする条件だけをまとめて、条件の論理和により分岐数を減らしプログラムを短くするなどである。この種の最適化については、自動化ツールの中で、ルールとして蓄える仕組みを持つ必要がある。

(2) 設計情報の再利用についての考察

状態遷移表を用いて作成されたアプリケーションにおいて、仕様変更が生じた場合の修正について考えてみる。例えば、表3のように入力キーkey-xの仕様に変更が生じた場合を考える。修正の基本的な考えとしては、トップダウンで行う。即ち、まず表3のようにソフトウェア・アーキテクチャのレベルから修正を検討し、書き換えの対象となる部分を明確にし修正を行う。続いて修正箇所に対応する各ボックス部分の機能の修正を行う。その後部品化の検討を行い、足りない部品については作成する。最後に修正したボックス部分のモジュールの作成を行い、部品との対応付けの修正を行う。

一般にトップレベルから見直すと修正の波及範囲は広がるが、総作業量が少なくなると考えられる。一見、トップダウンを強いると手間がかかり作業効率が劣化しそうであるが、そうならないがための設計標準化による分かり易さの保存であり、対応付け変更容易なための部品化である。また性能の確保についても、トップダウンは有効である。修正はトップダウンで検討し、常に分かり易いソフトウェア構造を保存すべきである、と考える。

表3．状態遷移表の仕様変更イメージ



8．おわりに

設計標準化を前提とした部品合成システムについて、具体的なアプリケーションをもとに考察した結果、一つの指針を得ることができた。しかし一方において、種々の課題が出た。その一つとしては、ソフトウェア・アーキテクチャの適用により詳細化された機能に対して、さらに分割詳細化を行う場合の標準化・形式化がある（図3の三段目の支援に相当する）。そのためには、用語の標準化や機能部品の切り出しの指針の明確化、また検索を考慮した部品のデータ抽象化について検討して行かなければならない。また、再利用支援を考慮した、設計情報および部品の蓄積方法を検討する必要がある。今後、これらの課題を含めて、状態遷移をベースとしたプロトタイプの作成を行う予定である。

［謝辞］

本稿の作成に際し、数度のディスカッションを行い有益な助言をしてくださった（株）東芝那須工場の井上課長に感謝いたします。

［参考文献］

[1] 高橋，他：ＩＭＡＰシステム (1)～(10)，情報処理学会第31回全国大会予稿集，pp.489～508 (1985)

[2] 原田　：事務処理分野における自動プログラミング，情報処理，Vol.28,No.10，pp.1378～1397 (1987)

[3] J.マーチン，他（国友，他訳）：ソフトウェア構造化技法，近代科学社

[4] 松村，他：ソフトウェア設計記述技法，東芝レビュー，Vol.41,No.8 (1986)

[5] 山城，他：ソフトウェア生産における再利用環境，ソフトウェア・シンポジウム’８８ (1988)

[6] 蓮田，他：ソフトウェア部品再利用に於ける部品仕様記述，情報処理学会第34回全国大会予稿集，pp.895～896 (1987)