ソフトウェア工学 126−11
（2000．3．10）

ファイル処理問題におけるクラス設計の自動化システムEOS/Pの開発

南 旭瑞*1，山内 亨和*1，原田 実*2
青山学院大学理工学部

ソフトウェア開発過程の中でも中流工程であるプロセス設計の自動化は重要であるにもかかわらず非常に遅れている．そこで我々は対象をファイル処理問題に限定し，データの依存関係と構造に着目して，問題に対する非手続き仕様をプログラムに自動変換可能な仕様に変換する自動設計システムEOS/Pを開発した．EOS/Pは，プロセスに対する多段修飾された等関係式仕様からこれを満足する一連のプロセスを設計する我々のプロセス設計理論に基いて，プロセス設計を行い，結果として生成された一連のプロセスを表すクラス図とプロセス中で生成された中間ファイルのデータ仕様とプロセスの最終段階において行うメインプロセスに対する一段修飾の等関係式仕様を自動生成する．

Automated Class Design System EOS/P for file processing problems

Akimizu Minami*1, Michitaka Yamauchi*1, Minoru Harada*2

We developed the automated class design system EOS/P. EOS/P inputs the class diagram which shows the relation between the process classes and the file classes in the problem and the set of equation oriented expressions called multi_step_modified EOS specification describing the function of each process class. EOS/P does process design based on our design theory, and generates the class diagram showing a series of designed mini-processes and the main process, the data specification of the intermediate files generated in the processes, and the single_step_modified EOS specification of the main process which will be performed at the final stage of the process.

*1 青山学院大学理工学部経営工学科
Undergraduate School of Science and Technology, Aoyama Gakuin University
*2 青山学院大学理工学部
Faculty of Science and Technology, Aoyama Gakuin University

1. 研究テーマ

1.1. 研究背景

これまでに，オブジェクト指向開発のための多くの方法論が提唱され，これらの方法論に基づく設計図を作成し，その設計図からコードを生成するCASEツールが実際に利用されている．Rational Software 社の Rational Rose[11]はその代表例である．しかしこれらのオブジェクト指向 CASE ツールはあくまで設計図を作成するためのツールである．生成するコードもクラスの宣言部のみで，メソッドの実装を生成することはできない．どのようなクラスが必要か考え，クラス毎にどんな属性，メソッドが必要か，メソッドの処理仕様(実装)はどうするかなどの上流工程の作業はいまだに開発者自身が判断しなければならないのである．

一方，オブジェクト指向分析・設計の自動化の研究においても，日本語文章による処理仕様からオブジェクト図を自動生成する CAMEO/A[12]やデザインパターンの設計図を開発中の設計図に合成するシステム OOPAS[10]などがあるが，クラス設計段階の自動化，特に何らかの高水準仕様から必要なクラス群やその関連およびクラスメソッドの処理仕様を自動的に生成するシステムの研究はこれまでに報告されていない．

1.2. 研究目的

オブジェクト指向開発における設計の自動化研究として原田研究室で，図１に示すように，一段修飾の等関係仕様からメソッド群とそのロジックテーブルによる機能仕様を自動生成する EOS/M[4][6]とロジックテーブル仕様から C++プログラムを自動生成する LOLA[5][8]を開発してきた．これらは，共に SOME[7]というオブジェクト指向モデリング環境に統合されている．SOME とはオブジェクト指向設計を支援する構造化オブジェクトモデリング環境であり，視覚的な設計図（クラス図とイベントトレース図）を構築することができる．

本研究の目的は，図１に示すように，これらの上流工程を自動化するために，対象をファイル処理問題に限定し，データの依存関係と構造に着目して，問題に対する非手続き仕様をプログラムに自動変換可能な仕様に変換する自動設計システム EOS/P を開発することである．具体的には，EOS/P の入力は，問題におけるプロセスクラスとファイルクラスの関係を表すクラス図と，プロセスクラスに対する多段修飾された等関係式仕様という等式の集合である．我々は，プロセスに対する多段修飾された等関係式仕様からこれを満足する一連のプロセスを設計する西村らのプロセス設計理論[3]を改良し，これに基いて EOS/P は，プロセス設計を行い，結果として生成された一連のプロセスを表すクラス図とプロセス中で生成された中間ファイルのデータ仕様とプロセスの最終段階において行うメインプロセスに対する１パス性を持つ一段修飾の等関係式仕様を自動生成する．

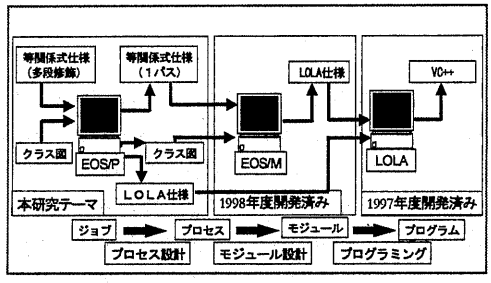

図 1　ソフトウェア開発の自動化の流れ

2. 等関係式

2.1. 等関係式とは

EOS の入力仕様である等関係式仕様はファイル処理の要求を非手続き的に記述できる仕様であり，実体や関連の属性項目の値リスト間の図２に示すような導出関係を表す数式の集合である．左辺は単一の項目からなり，右辺はそれを定義する項目からなる算術式や条件式である．そこでは，各データ項目には，その項目の値がどのクラスの値であるかを示すためのクラスの識別項目名と，そのクラスのどんな条件を満たすインスタンスレコードの値なのかを指定する条件が，"."を用いて付与されている．例えば，図２の等関係式は Salary という項目が BasicSalary と OverTimePay の和であることが定義されている．またこの等関係式はｂファイルとｏファイルの両方に入っている社員インスタンスに対してだけ成立することを表している．

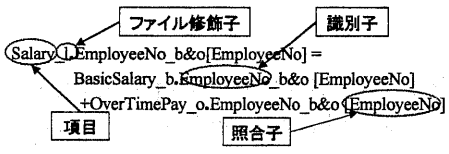

図 2　等関係式

2.2. 多段修飾の等関係式とは

等関係式において図２ように，ひとつの"."修飾からなる項目を一段修飾項目と呼ぶ．この"."修飾は関連ファイルを経由することによって多段にすることができる．多段修飾の等関係式は，複数の"."修飾を持つ項目が存在する等関係式であり，複数のファイル内のキー項目が同一値を持つレコードを経由して関係している項目間の修飾関係を表している．図３で記述された多段修飾された等関係式仕様は，ある部(ファイル d)に所属している(ファイル b)ある社員(ファイル e)の社員名を表す多段修飾された等関係式である．

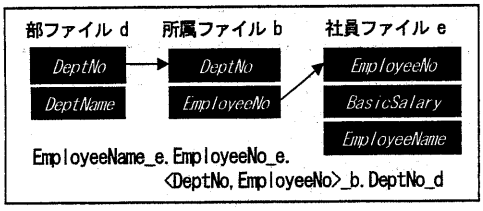

図 3　多段修飾された等関係式

3. 事例問題

本研究では事例問題として給与計算問題を取り上げた．以下にこの問題を EOS/P で自動設計する際に入力として必要なものと，EOS/P によって自動設計された結果として出力されるものを説明する．

3.1. 入力に必要なもの

EOS/P を用いるにあたり，給与計算問題におけるプロセスクラスとファイルクラスの関係を表すクラス図と，プロセスクラスに対する多段修飾された等関係式仕様とデータ構造定義を与える．

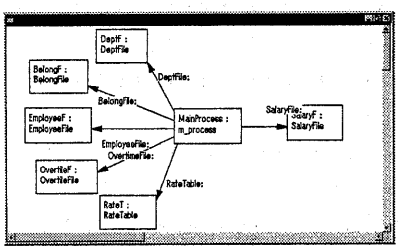

図 4　EOS/P 仕様のクラス図

3.1.1. クラス図

図４に示すクラス図は，この問題が５つの入力ファイルクラス，１つの出力ファイルクラス(SalaryF)，主たる計算を行う１つのプロセスクラス(MainProcess)からなることを表している．

3.1.2. データ構造定義(ファイル仕様)

データ構造定義は入力ファイル（ファイルタイプ I）と出力ファイル（ファイルタイプ O）の形式を定義したものである．形式としてファイル名（ロール名）；ファイル修飾子;ファイルタイプ;レコード名;*識別子;項目;;;といった順で以下のように表される．

[1]SalaryFile;sl;O;SalaryRecord;**DeptNo;*EmployeeNo;EmployeeName;Salary;TotalSalary;;;
[2]EmployeeFile;e;I;EmployeeRecord;*EmployeeNo;EmployeeName;Unit;Salary;;;
[3]DeptFile;d;I;DeptRecord;*DeptNo;DeptName;;;
[4]BelongFile;b;I;BelongRecord;**DeptNo;*EmployeeNo;;;
[5]OvertimeFile;o;I;OvertimeRecord;**-EmployeeNo;**-Rank;**-Date;Time;;;
[6]RateTable;r;I;RateRecord;*Rank;Rate;;;

[1]を例にとって説明すると，このファイル仕様はファイル名（ロール名）が SalaryFile で，ファイル修飾子は sl，ファイルタイプが０であるから出力ファイルであり，レコード名は SalaryRecord，識別子(*付き項目)は DeptNo，EmployeeNo の２つを持ち，EmployeeName，Salary，TotalSalary の３つの項目から成り立つことを表している．なお，このファイルは図４のクラス図では SaralyF というクラスで表現されている．

3.1.3. 等関係式仕様(プロセス仕様)

給与計算問題に対する先のクラス図における MainProces に対する要求仕様を非手続き的に表現するために，以下の多段修飾された等関係式仕様を指定する．

[1]PRINT(SalaryRecord_sl. EmployeeNo_e. <DeptNo, EmployeeNo>_b. DeptNo_d" %d %d %s %d %d¥n");
[2]DeptNo_sl. DeptNo_d=DeptNo_d. DeptNo_d;
[3]EmployeeNo_sl. EmployeeNo_e=EmployeeNo_e. EmployeeNo_e;
[4]EmployeeName_sl. EmployeeNo_e=EmployeeName_e. EmployeeNo_e;
[5]TotalSalary_sl. EmployeeNo_e=TotalSalary_e. EmployeeNo_e;

[6]OvertimeSalary_w.EmployeeNo_e&o[EmployeeNo]
=SUM(Rate_r.Rank_r.<EmployeeNo,Rank,Date>_o.Empl
oyeeNo_e*Time_o.<EmployeeNo,Rank,Date>_o.Employe
eNo_e*Unit_e.EmployeeNo_e,EmployeeNo_e&o[Employe
eNo]);

[7]OvertimeSalary_w.EmployeeNo_e&!o[EmployeeNo]=
0;

[8]TotalSalary_sl.EmployeeNo_e=Salary_e.Employee
No_e+OvertimeSalary_w.EmployeeNo_e;

[1]は出力式であり，出力すべき Salary Record とその出力書式を引数として持つ．[6]は OverTimeSalary が Rate，Time，Unit を掛けたものの和である事を表している．またこの式が成り立つのは，ファイルｅとｏの両方において識別子 EmployeeNo の値が同じ値を持つクラスインスタンスに対してのみである．これに対して[7]は，識別子 EmployeeNo のある値に対してファイルｅには存在するがファイルｏには存在しないクラスインスタンスに対してのみ成立する事を表している．[8]は TotalSalary は Salary と OverTimeSalary の和である事を表している．

3.2. 出力として生成されるもの

EOS/P を実行することによって，中間ファイルのデータ構造定義，メインプロセスに対する１パス性を持つ新しい等関係式仕様，中間ファイルを生成するためのミニプロセスからなるクラス図，各ミニプロセスに対する LOLA 仕様を自動生成する．

3.2.1. クラス図

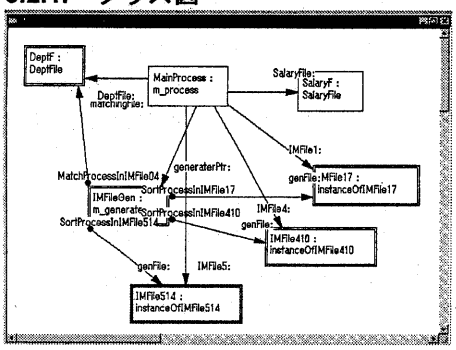

図 5 EOS/M 仕様のクラス図

EOS/P が事例に対して設計した結果を表す新しいクラス図を図５に示す．ここでは先の図４のクラス図に比べると，新しい中間ファイルクラス IMFile17，IMFile410，IMFile514 が追加され，これを MainProcess クラスがそれぞれ IMFile1，IMFile4，IMFile5 という名前(ロール名)で入力ファイルとしている．

3.2.2. プロセスフロー

プロセスフローとは，多段修飾された等関係式群を一段修飾の等関係式群に変換する過程において，必要となった一群の中間ファイルを生成するためのソートミニプロセスクラスと照合ミニプロセスクラスからなる処理を表したものであり，図６に示すようなクラス図として生成される．実際ここでは，図５のクラス図で MainProcess がアクセスしている IMFile17，IMFile410，IMFile514 が生成されている．

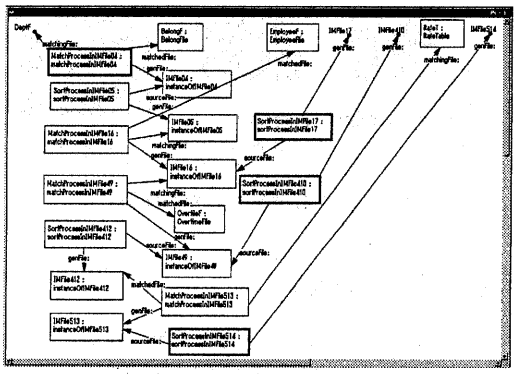

図 6 プロセスフローを表したクラス図

3.2.3. データ構造定義(ファイル仕様)

EOS/P によって自動生成された中間ファイルの内 MainProcess がアクセスしている中間ファイルと出力ファイルのファイル仕様は，以下のように自動生成される．

[1]DeptFile;d;I;DeptRecord;*DeptNo;DeptName;;;

[2]IMFile1;x1;I;IMRecord1;*DeptNo;*EmployeeNo;Em
ployeeName;Salary;Unit;;;

[3]IMFile4;x3;I;IMRecord3;*DeptNo;*EmployeeNo;*R
ank;*Date;Time;;;

[4]IMFile5;x4;I;IMRecord4;*DeptNo;*EmployeeNo;*R
ank;*Date;Rate;;;

[5]SalaryFile;sl;O;SalaryRecord;*DeptNo;*Employe
eNo;EmployeeName;Salary;TotalSalary;;;

ここで，例えば[2]は中間ファイル IMFile1（ファイルクラス IMFile17）を示しており，図６のクラス図からこれは(図中最上段中央右側の黒丸で表現されている)，入力ファイル d(DeptF)，b(BelongF)，

e(EmployeeF)を照合した結果生成された中間ファイルであることが分かる．

3.2.4. 等関係式仕様(プロセス仕様)

EOS/P は 3.1.3 に示した多段修飾の等関係式仕様を以下に示す 1 パス性（一段修飾条件，順序条件，構造条件を満たす）の等関係式仕様に変換する．この生成された仕様を，図５の新しいクラス図の MainProcess の EOS 仕様として指定する．なお，ここまで生成されたクラス図やファイル仕様やプロセス仕様を基に、既に開発の EOS/M がロジックテーブルを自動設計し，これを SOME が C++プログラムに自動プログラミングする．

[1]EmployeeName_s1.EmployeeNo_x1=EmployeeName_x1.EmployeeNo_x1;

[2]EmployeeNo_s1.EmployeeNo_x1=EmployeeNo_x1.EmployeeNo_x1;

[3]TotalSalary_s1.EmployeeNo_x1=TotalSalary_x1.EmployeeNo_x1;

[4]DeptNo_s1.DeptNo_d=DeptNo_d.DeptNo_d;

[5]OvertimeSalary_w.EmployeeNo_x1&x3[EmployeeNo]=SUM(Rate_x4.Rank_x4*Time_x3.<EmployeeNo,Rank,Date>_x3*Unit_x1.EmployeeNo_x1,EmployeeNo_x1&x2[EmployeeNo]);

[6]OvertimeSalary_w.EmployeeNo_x1&!x3[EmployeeNo]=0;

[7]TotalSalary_s1.EmployeeNo_x1=Salary_x1.EmployeeNo_x1+OvertimeSalary_w.EmployeeNo_x1;

[8]PRINT(SalaryRecord_s1.EmployeeNo_x1" %d %d %s %d %d¥n");

以上の式は全て明らかに一段修飾である．なお，[1]の等関係式におけるファイル x1，x3，x4 は，図６のクラス図が示すプロセスフローによって生成された中間ファイルであり，そのデータ仕様は3.2.3の[2]，[3]，[4]に生成されている．

4. プロセス設計の自動化

4.1. プロセス設計の流れ

EOS/P によるプロセス設計とは，等関係式仕様で与えられた要求仕様をいくつかのプロセス仕様に分割することである．この分割の基本は，まず与えられた要求仕様を閉じたプロセス仕様群に分割し，次に一連の中間ファイルを作成するプロセスを新たに生成することである．このプロセス設計では図７に示すように，要求仕様に含まれる多段修飾を取り除き，順序条件と構造条件を満足するようにソートミニプロセスや照合ミニプロセスを追加し，生成された中間ファイルに無駄がないように中間ファイルの統合を行い，最後に要求仕様と等価な一段修飾のプロセス仕様を生成する．以下に各設計プロセスを説明する．

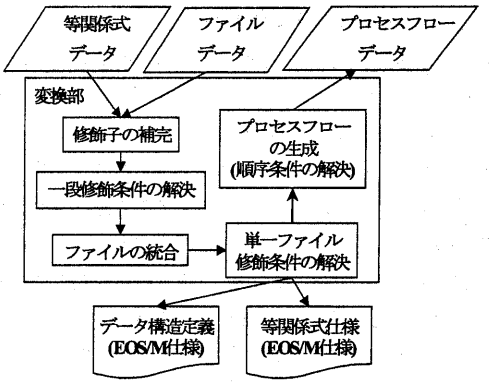

図 7 プロセス設計の流れ

4.1.1. 修飾子の補完

仕様中の各等関係式の表わす処理要求が，実際はどのクラスに対する処理要求なのかを仕様中の要求項目から入力項目へ至る経路を見つけることにより確定し，それを表わすために必要な修飾子をもとの等関係式に付け加える．事例中の 3.1.3 の等関係式［1］は給与明細リストの内の SalaryRecord に関する出力式である．この事例は給与明細リストを出力する要求であるので，この出力式は PRINT 文となっている．この PRINT 文の引数は，「クラス『部』に所属するクラス『社員』の SalaryRecord を出力する」という処理要求を表わすために，要求単位項 DeptNo_d から始まり，関連項 <DeptNo,EmployeeNo>_b を経由して要求対象項 EmployeeNo_e に至る修飾子 EmployeeNo_e.<DeptNo,EmployeeNo>_b.DeptNo_d によって修飾されたレコード SalaryRecord_s1.EmployeeNo_e.<DeptNo,EmployeeNo>_b.DeptNo_d である．一方，この明細行の属性項目である EmployeeName や TotalSalary に関する値は式[４]，式[５]で与えられている．この２式の各項目の修飾子は，どの対象の項目なのかを指定するために，本来ならば式[1]と同一の EmployeeNo_e.<DeptNo,EmployeeNo>_b.DeptNo_d でな

ければならない．しかし，記述を簡略化するために，実際には，式[４]，式[５]のように修飾子中の要求対象項以外の項を省くことができる．したがって，プロセス設計の際には，これらの省かれた項を補っておかなければならない．これを修飾子の補完という．修飾子の補完を行うことにより，構造条件を解決することができる．

4.1.2. 一段修飾条件の解決

EOS/P の後を受ける EOS/M で扱える等関係式仕様は，"."修飾が１回だけのものでなければならない．そのため，多段修飾を含む等関係式仕様をそれと等価な一段修飾の等関係式仕様に変換する必要がある．多段のドット修飾というのは幾つかのファイルを経由した修飾関係なので，これを一段修飾に変換するには，それらの修飾によって制限された情報のみを含む最終的な中間ファイルを作る必要がある．この最終的な中間ファイルを求めるには，多段修飾の中に連続している２つの"."修飾があるとき，これらの"."修飾のファイル修飾子になっているファイルをこれらが共有する識別子によって照合した新しい中間ファイルをファイル修飾子に持つ１つの"."修飾に取り替えることによって，１つずつ"."修飾を減らしていく変換を行えばよい．形式的には我々のプロセス設計理論にある以下の定理１と定理２を用いてこの中間ファイルを求める．なお，ここで，c=a&b[I]はファイル a とファイル b を識別子 I で照合した結果のファイル c を表している．この際，ファイル c のデータ項目は，ファイル a のデータ項目と同じものとなる．

[定理 1]　t = r&s[I1]として，
X_f.<I1,I0>_r.I1_s = X_f.<I1,I0>_t である．

[定理 2]　a = o&q[I0]，　b = p&q[I0]　として，
X_f.I0_o&p[I0].<I1,I0>_q
= X_f.　<I1,I0>_a&b[<I1,I0>]である．

図８はシステム内部で一段修飾条件の解決を行う際のデータの変換について示している．修飾子 I0_e.<I1,I0>_b.I1_d を例に説明する．図中，OM(OriginalModifier)は原要求となる多段修飾中の１つの"."修飾を表し，TM(TranslatedModifier)は中間ファイルを用いた等価な一段修飾になった"."修飾を表しており，具体的な値はその右下に示されている．図に示されているように，外部ファイル b と d を I1 で照合してファイル x0 を作成し，これとファイル e を I0 で照合してファイル x1 を作成し，この x1 のみを用いた一段修飾子<I1,I0>_x1 が生成されている．

4.1.3. 順序条件の解決

等関係式仕様中の出力式における識別子のソート順位と入力ファイル中の識別子のソート順位が異なるときは，EOS/M で処理を行うことができない(順序条件)ので，出力式のソート順位に合わせて入力ファイルをソートしなければならない．

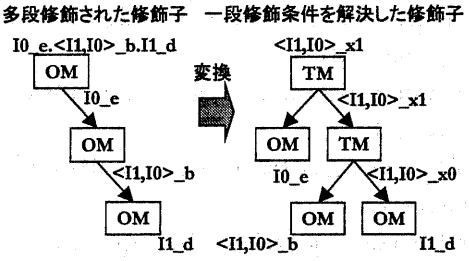

図 8 一段修飾条件の解決

4.1.4. 中間ファイルの統合

等関係式中の各項目に対してプロセス設計を行い中間ファイルを生成すると，データが冗長に保持されていることが多い．この冗長性を排除し，プロセス数を減らすことはジョブの処理時間の短縮につながると同時に，中間ファイルのための物理的な資源の節約にもつながる．そこで中間ファイルの統合を行う．統合を行う際に次の２点に注意しなければならない．第一に，統合によってファイルの構造は変化しないか，第二に，統合されるファイル中のデータが統合によって欠落や重複しないかということである．このような不都合を排除するためには，①上位内包性，②識別子ごとの多重度が同一，③レコードを構成する項目が同一であるという３つの条件が成立しなければならないことがわかっている．以下でこれらについて説明する．

[上位内包性]　ファイル g をファイル f に統合する(置きかえる)ためには，ファイル g 内における識別子のソート順位が，ファイル f 内における識別子のソート順位と同じでなければならない（構造条件）．言い換えれば，ファイル g の部分ソート項目がファイル f の部分ソート項目になっていなければならない．この

ときファイルｆはファイルｇを上位に内包しているという．

[識別子ごとの多重度が同一]　ファイルの識別子ごとの多重度（レコード数）という概念を導入する．識別子の各値に対して必ず１件のレコードが存在する場合を多重度１と定める．他の場合も表１に示すように多重度の値を定める．ファイルｇをファイルｆに統合するためには，この統合によってレコードの漏れや重複が起きてはいけないので，両ファイルの多重度が共通する全ての識別子において同じでなければならない．

定義	多重度	データ構造定義での表記法
識別子の各値に対して必ず1件のレコードが存在する	1	* EmployeeNo
識別子の各値に対して何件かのレコードが欠けている	1-	*-EmployeeNo
識別子の各値に対して必ず1件から複数件のレコードがある	N	** EmployeeNo
識別子の各値に対して0件から複数件レコードがある	N-	**-EmployeeNo

表 1多重度の定義をまとめた表

[レコードを構成する項目が同一]　ファイルｇをファイルｆに統合する（置きかえる）ためには，ファイルｆとファイルｇのデータ項目が同じものでなければならない．中間ファイルを生成するには4.1.2で示した一連の照合処理を行うが，生成された中間ファイルのデータ項目は，これらの照合プロセスにおける最左ファイルのデータ項目なので，結局中間ファイル ｇ,ｆを生成するのに用いられた最左ファイルが同一であればよい．

4.1.5.　単一ファイル修飾条件の解決

EOS/M の制約条件の１つである単一ファイル修飾条件には，「等関係式の右辺に現れる各値リストの項目識別子のファイル修飾子は等しくなければならない」とある．そこでこの制約を解決するために，それらが等しくない場合は各項目識別子のファイル修飾子をそこに現れる全てのファイルを照合した照合ファイルからなるファイル修飾子に取り替えることで，この制約条件を満足させる．

4.1.6.　プロセスフローの生成

多段修飾された等関係式仕様を一段修飾の等関係式仕様に変換する過程についての情報から，3.2.2で述べたプロセスフローを生成する．このプロセスフローから

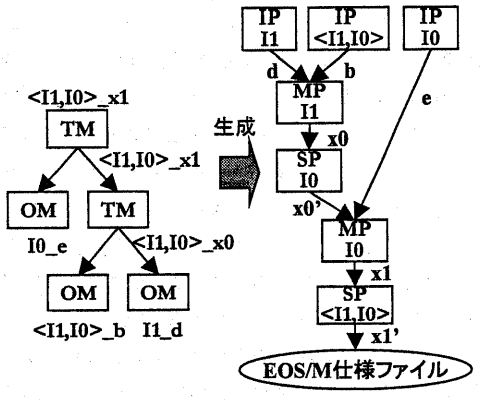

図 9 プロセスフローの生成

EOS/M 仕様のクラス図と LOLA 仕様を生成する．図９でプロセスフロー生成の手順を説明する．図中の左側の木はこの木を基に，図８で一段修飾条件を解決する過程で得られたものである．まず，OM から入力プロセス(IP)を生成し，TM から照合ミニプロセス(MP)を生成する．この時，ソート順位が合わなければソートミニプロセス(SP)を生成した後，照合ミニプロセスを生成する．最後に順序条件を解決すために，出力レコードの識別子順にソートするソートミニプロセスを生成する．

4.1.7.　クラスの生成

メインプロセスクラス，中間ファイル生成プロセスクラスの順でクラスを生成し，それぞれのコンストラクタ，デストラクタ，関連も同時に生成する．その後，プロセスフローからミニプロセスクラスおよび中間ファイルクラスを生成する．次に出力ファイルクラスを生成し，最後に EOS/P 仕様クラス図(図４)に存在していたメインプロセスクラス，ファイルクラス以外のクラスを追加して EOS/M 仕様のクラス図（図５と図６）の生成を終了する．

5.　おわりに

5.1.　評価

本研究で開発した EOS/P によって，多段修飾された等関係式仕様を１パス性の等関係式仕様に変換し，その過程において中間ファイルとプロセスフローを自動設計することができた．このことにより，利用者が一般的な（多段修飾された）等関係式仕様を EOS/P に与えるだけで，EOS/P が１パス性の等関係式仕様に変換し，その過程において必要な中間ファイルとこれらの生成

を行うプロセスフロー，EOS/M 仕様のクラス図，LOLA 仕様を自動生成することができる．EOS/P を事例(給与計算問題)に適用したところ 8 個の一般的な等関係式，6 個の外部ファイル，EOS/P 仕様のクラス図から 8 個の 1 パス性の等関係式，3 個の中間ファイル，2 個の外部ファイル，EOS/M 仕様のクラス図，中間ファイルを生成するミニプロセスに対する LOLA 仕様を自動生成することに成功した．更にこの生成された 1 パス性の等関係式とクラス図を EOS/M に与えると，EOS/M がロジックテーブル仕様のメソッド群を自動生成し，これらのメソッドとクラス図から LOLA が事例問題を処理する C++のプログラムを自動生成できた．この生成されたプログラムを Visiual C++5.0 を使用して，コンパイルし実行したところ正しい実行結果を得ることができた．このことから，EOS/P が自動生成した中間ファイルとプロセスフローが正しいものであることが分かった．

5.2. 今後の課題

今後の課題としては次のものが挙げられる．

第一に，等関係式を作成するための CASE ツールを作成することである．等関係式は一般的な利用者が記述するには煩雑であるので，容易に等関係式を記述できるようなエディタの開発が必要である．更には日本語要求仕様書から等関係式仕様を自動生成するシステムなどの研究が望まれる．

第二に，現在は等関係式の記述が間違っていてもこの間違いを検出することができず，誤った結果を出力するか，システム自体が停止してしまうので，等関係式のチェック機能の追加が必要である．

第三に，構文上のエラーチェックは行っているが，制約条件を満たしているかどうかのチェックは十分でないので，これらエラーメッセージを出力するための処理が必要である．

6. 参考文献

[1]原田実，中村義幸：〝プログラムの構造と論理の自動設計システム EOS/M〟，情報処理学会論文誌，Vol.34，No.9，pp.2013-2024(1993.9)．

[2]原田実，西村淳一，中村義幸：〝非手続き仕様からのプロセス設計の自動化〟，電子情報通信学会論文誌，Vol.J77-D-I，No.2，pp.196-206(1994.2)．

[3]原田実，西村淳一：“非手続き仕様からのプロセス設計の自動化システム EOS/P〟，情報処理学会ソフトウェア工学研報，94-SE-101，pp.79-88(1994.11)．

[4]原田実，田口志郎：“等関係式仕様から決定表形式のモジュール仕様への変換理論〟，電子情報通信学会春季大会論文集，D-3-7，pp.65(1997.3)．

[5]原田実，水野高宏：“決定表を用いたメソッド機能の高水準記述言語 LOLA〟，情報処理学会ＯＯ'98 シンポジウム論文集，朝倉書店，pp.86-94(1998.9)．

[6]原田実，川端崇央，田口志郎:“クラス機能の非手続き的仕様から手続き的なメソッド処理仕様の自動生成システム SOME/EOS"，情報処理学会ＯＯ'99 シンポジウム論文集，情報処理学会，pp11-19(1999.7).

[7]原田実，北本和宏，岩田隆志: "制御構造とイベント送信を図示できる構造化オブジェクトモデリング環境 SOME"，情報処理学会ＯＯ'97 シンポジウム論文集，朝倉書店，pp136-144(1997.7).

[8]原田実，水野高宏:“決定表を用いたメソッド機能の高水準記述言語 LOLA"，情報処理学会ＯＯ'98 シンポジウム論文集，朝倉書店,pp.86-94(1998.9).

[9]Minoru Harada and Takahiro Mizuno: " Executable C++ Program generation from the structured object-oriented design diagrams ", Proceedings of APSEC'99(Asia Pacific Software Engineering Conference), Takamatsu. pp.630-636 (1999. 12).

[10]原田実，永山英嗣:“設計図の融合機能を持つデザインパターン適用支援ツール OOPAS"，情報処理学会ＯＯ'98 シンポジウム論文集,朝倉書店，pp104-111(1998.9).

[11]Rational 社ホームページ;http://www.rational.com/

[12]原田実，田村浩樹，野村佳秀: "オブジェクト指向分析システム CAMEO/A と帰納推論によるルールの学習"，情報処理学会ＯＯ'96 シンポジウム論文集，朝倉書店，pp57-64(1996.7).