PCTE による CASE ツールの統合とその応用

池田 克則・落水 浩一郎

1996 年 3 月 18 日

IS-RR-96-0010S

北陸先端科学技術大学院大学

情報科学研究科

〒 923-12 石川県能美郡辰口町旭台 15

k-ikeda@jaist.ac.jp

ochimizu@jaist.ac.jp

ISSN 0918-7553

要旨

本論文では、PCTE(Portable Common Tool Environment) を用いて、商用 CASE ツールのデータを統合する手段を論じたものである。様々な統合技術の調査及び開発を紹介しつつ、その結果を StP/OMT と ObjectCenter の統合にまとめる。上流工程と下流工程の CASE ツールをデータ統合することにより、作業の連続性の保証と変更の波及の追跡が容易に分かることが期待される。本論文ではそれを交換するためのスキーマ例の結果も示す。さらに、このツールを用いて実際のソフトウェアの開発を行ない、それに基づいて統合ツールの問題点を分析しつつ、ツール統合のありかたについて提案する。

もくじ

1 はじめに **1**

1.1 研究の背景と目的 1

1.2 本論文の構成 2

2 PCTE による CASE ツール統合技術 **3**

2.1 CASE ツール統合の考え方 3

2.2 PCTE とは 5

2.3 PCTE によるツール統合の手順 6

2.4 CASE ツールのデータインタフェースの特徴 6

2.5 PCTE によるデータ統合の方法 7

2.5.1 PCTE ネーティブな CASE ツールの作成 7

2.5.2 CASE ツールの移植 7

2.5.3 リホスティング 8

2.5.4 CASE ツールのカプセル化 8

2.5.5 Emeraude TCI の使用 11

2.6 バージョン管理の導入 12

3 StP/OMT と ObjectCenter の統合 **15**

3.1 StP/OMT との統合 15

3.1.1 StP/OMT のデータインタフェースの特徴 15

3.1.2 統合の問題点及び解決策 17

3.1.3 スキーマの作成 18

3.1.4 StP/OMT の統合方式 20

3.2 ObjectCenter の統合 22

3.2.1 ObjectCenter のデータインタフェースの特徴 22

3.2.2 統合の問題点及び解決策 23

3.2.3 スキーマの作成 23

3.2.4 ObjectCenter の統合方式 26
3.3 データ統合の指針 27
3.3.1 スキーマの作成指針 27
3.3.2 ツール作成の指針 27

4 統合ツールの評価 **30**
4.1 「家計簿シミュレーションシステム」による開発 30
4.1.1 OMT 法によるモデリング 30
4.1.2 統合ツールによる実装 34
4.2 統合ツールの効果と問題点 34
4.2.1 統合ツールの効果 34
4.2.2 統合ツールの問題点 36
4.3 ソフトウェア開発の後戻りによる問題点 39

5 おわりに **44**

謝辞 **46**

参考文献 **47**

A 「家計簿シミュレーションシステム」の問題 **49**

B Emeraude PCTE 管理&操作マニュアル **52**
B.1 PCTE の起動 52
B.1.1 server の起動 52
B.1.2 client の起動 53
B.2 PCTE の終了 54
B.3 PCTE での C プログラム実装について 54
B.3.1 ワーキングスキーマの変更 54
B.3.2 オブジェクトの作成 55
B.3.3 オブジェクトの編集 56
B.3.4 オブジェクトのコンパイル 56
B.3.5 オブジェクトの実行 57
B.4 スキーマの定義について 57
B.4.1 sds_design 起動までの準備 57
B.4.2 スキーマの定義 57

B.4.3 スキーマの描写 58
B.4.4 スキーマのコンパイル 58
B.4.5 スキーマ図のポストスクリプト変換 58

C Emeraude PCTE によるツール統合技術 **61**
C.1 はじめに 61
C.2 CASE ツール統合の考え方 61
C.3 PCTE によるデータ統合の方法 63
C.3.1 PCTE ネーティブな CASE ツールの作成 63
C.3.2 CASE ツールの移植 64
C.3.3 リホスティング 65
C.3.4 CASE ツールのカプセル化 66
C.3.5 Emeraude TCI の使用 70
C.4 PCTE のバージョン管理 70
C.5 おわりに 71

図一覧

2.1 トースタモデル 4
2.2 カプセル化ツール 5
2.3 contents_get を使用したカプセル化ツールの概念 9
2.4 PCTE 通知体の概念 10
2.5 notifier を利用したツール統合の概念 11
2.6 import メカニズムの概念 12
2.7 PCTE のバージョン管理の概念 13

3.1 StP/OMT の概観 16
3.2 StP/OMT データの流れ 17
3.3 StP/OMT のファイル構成 18
3.4 StP/OMT スキーマ 19
3.5 StP/OMT の統合方式 21
3.6 Object Center の概観 22
3.7 Object Center スキーマ 24
3.8 Object Center の統合方式 26

4.1 家計簿シミュレーションシステムのオブジェクトモデル 31
4.2 家計簿シミュレーションシステムの動的モデル 32
4.3 家計簿シミュレーションシステムの機能モデル 33
4.4 StP/OMT で作成したオブジェクトモデル図 35
4.5 StP/OMT のクラステーブルエディタ 36
4.6 StP/OMT で生成したインタフェースファイルの例 (balance_happened.h) . 37
4.7 StP/OMT で生成した実装ファイルの例 (balance_happened.c) 38
4.8 ソフトウェア開発の流れ 40
4.9 変更の影響の伝搬 41
4.10 依存関係を考慮した新しいスキーマ 43

B.1 Chenge Working Schema ダイアログ 59
B.2 Create object ダイアログ 59
B.3 sds_design で用いられるアイコン 60

C.1 トースタモデル 62
C.2 カプセル化の概念 62
C.3 リホスティングの概念 65
C.4 contents_get 関数を用いたカプセル化ツールの概念 66
C.5 カプセル化ツール evi のプログラム 67
C.6 PCTE 通知体の概念 68
C.7 StP/OMT の統合方式 69

表一覧

2.1 contents 操作関数とそれに対応する file 操作関数 8
2.2 PCTE のバージョン操作 14

3.1 StP/OMT で使用するオブジェクトの型 20
3.2 Object Center で使用するオブジェクトの型 25
3.3 Object Center で使用するキー 25
3.4 Object Center で使用する型参照子 25

C.1 contents 操作関数とそれに対応する file 操作関数 64
C.2 PCTE のバージョン操作 71

第 1 章

はじめに

1.1 研究の背景と目的

近年のソフトウェア開発では、ソフトウェアの品質向上及び開発期間の短縮・開発要員の削減[1]のためにCASE(Computer-Aided Software Engineering) ツールを導入する機会が増えてきている。しかしながら、現在一般的に使用されているCASE ツールはソフトウェア生産プロセス (仕様分析→設計→コーディング→デバッグ) 全体を支援している訳ではなく、それぞれの作業を支援しているに過ぎない。すなわち、それ自身のCASE ツールで閉鎖的な環境を構築しているものが多く、それぞれのCASE ツールで生産されるデータを、直接下流工程のCASE ツールで利用するのが困難な場合が多い[2]。下流工程のCASE ツールで使用するためには、データのフォーマット変換ツールを作成すれば良いが、ツール毎に変換ツールを作成しなければならず、すべてのCASE ツールでデータを利用することは事実上不可能である。また、ある特定のCASE ツールにおいては、現在使用している環境で上流工程または下流工程のツールを使用できるといった、ツール間の連動を考慮したものが存在するが、全てのCASE ツールが連動できるといった汎用性がある訳ではない。さらに、下流工程のCASE ツールから上流工程のCASE ツールに戻って作業を行なうといった、作業の後戻りを考慮しているCASE ツールはほとんど見受けられない[3]。

このような問題を解決するためには、データ統合・データの格納と管理・プログラムの実行・ツールとユーザ間のコミュニケーション・ツール相互間のコミュニケーションといった機能を含んだ共通の枠組 (フレームワーク) が必要になる [1]。PCTE は、開放型リポジ

[1]開発費用の削減が主な目的であると考えられる。

[2]CDIF(CASE Data Interchange Format) と呼ばれる異なるベンダから提供されるツール間でCASE 情報を交換するためのフォーマットがEIA(Electric Industries Association) から提案されているが、まだ普及していない。

[3]最近のいくつかのツールには、このような機能を持ったものも見受けられるようになってきたが、まだ少数である。

トリのための共通ツールインタフェース標準であり、CASE ツールが高度な統合性と可搬性を持つことを可能にするための、共通で標準的なサービス群を定義している。PCTE をツール統合のフレームワークに採用することにより、データ統合やデータの格納や管理といったことが容易に行なうことができるようになる。

本研究では、既存の CASE ツールのデータを PCTE を用いて統合を行なうための統合技術を開発すると共に、ツール統合の指針を提案することを目標としている。

1.2 本論文の構成

本稿では、第 2 章で PCTE による CASE ツール統合技術、第 3 章で StP/OMT と ObjectCenter のツール統合、第 4 章で統合ツールの評価について述べ、第 5 章でまとめる。

第 2 章

PCTE による CASE ツール統合技術

2.1 CASE ツール統合の考え方

CASE ツールの統合法には、3 種類の方法があり、それぞれ以下のような特徴を持つ [2]。

1. **データ統合：** ツールと環境の間での情報の共有

2. **制御統合：** 環境内の個々のツール間の協調と通信制御の共有

3. **表示・操作統合：** 環境内でのツールによるユーザインタフェースの共有

このうち表示・統合操作に関しては、UNIX で使用されている X ウインドウシステム等のユーザインタフェースを採用すれば、容易に統合することが可能である。しかしながら、残りのデータ統合や制御統合に関しては、新たなプラットフォームを採用しなければ統合することはできない。

ECMA(European Computer Manufacturers Association) と NIST(National Institute of Standards and Technoligy) は、「トースタモデル」と呼ばれる CASE 環境の枠組 (フレームワーク) を提唱した。図 2.1 にその概要を示す。このモデルは、前方に User Interface Services と呼ばれる表示・操作を統合したものが、また後方に Object management Services と呼ばれるデータを統合したものが存在しており、その間には Tool Slots と呼ばれる部分が存在する。この Tool Slots は、トースタにパンを差し込むのと同様に様々なツールを差し込むことによって、User Interface Services と Object management Services をツールが使用することができることを示している。すなわち、共通の表示・操作及びデータをツールで共有することが可能であることを示している。また、ツールの連動といったようなツール間のコミュニケーションのために、Communication Services が用意されている。これは、制御統合が可能であることを示している。

この状態では、Tool Slots で使用するツールは、CASE 環境の枠組に準じたツールを使用しなければならないが、図 2.2 に示すように、CASE 環境の枠組に準じていないアプ

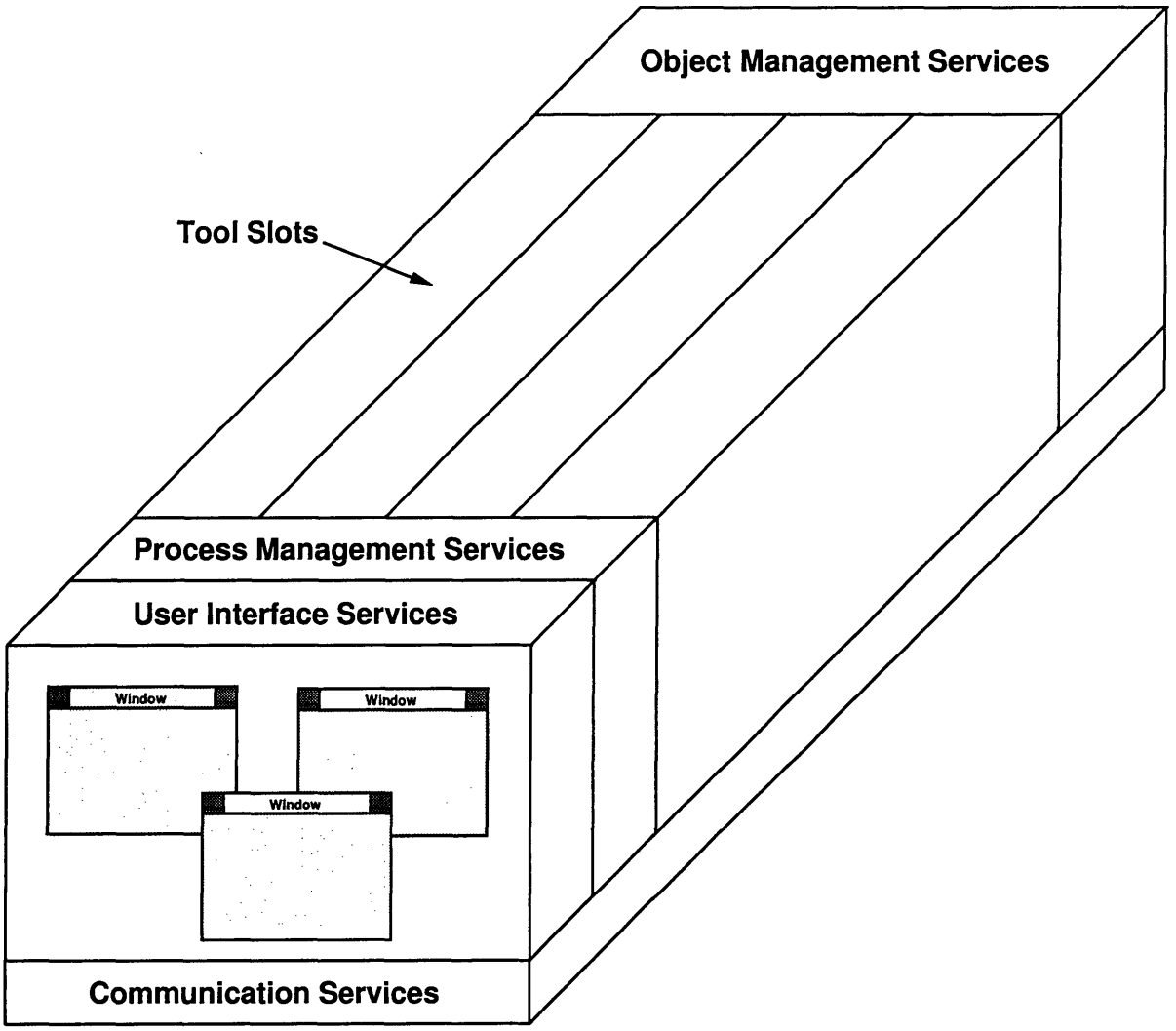
Object Management Services
Tool Slots
Process Management Services
User Interface Services
Window
Window
Window
Communication Services

図 2.1: トースタモデル

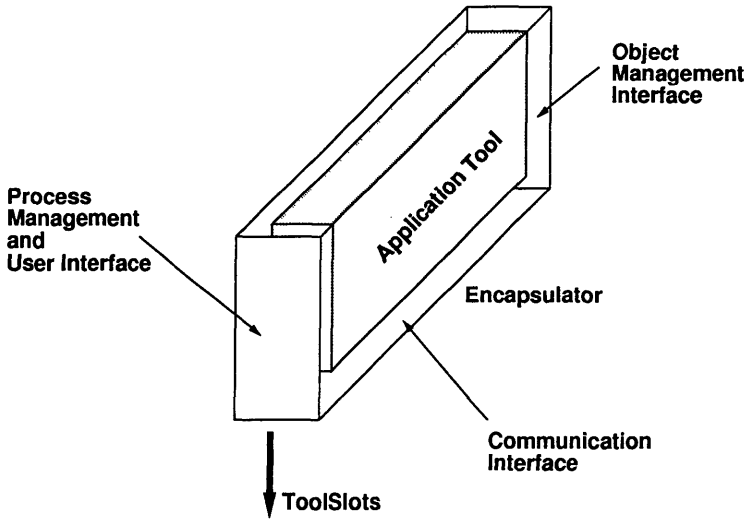

図 2.2: カプセル化ツール

リケーションツールを CASE 環境の枠組で使用できるようにするカプセル化技術を使用することで、既存のアプリケーションも同じ CASE 環境で使用することができるようになる。

このような、CASE 環境を実現するためのフレームワークの一つに PCTE が存在する。

2.2 PCTE とは

PCTE とは、可搬型共通ツール環境 (Portable Common Tool Environment) の略で、開放型リポジトリのための共通インタフェース標準であり、CASE ツールが高度な統合性と可搬性を可能にするための共通で標準的なサービス群を提供している。

ここでリポジトリとは、仕様書やソースコード、CASE ツール生産物等「オブジェクト」と呼ばれるソフトウェア開発で必要なデータを格納しておく場所である。これらのオブジェクトは、「スキーマ」と呼ばれるオブジェクト情報とオブジェクト間の関連情報を記述したものに従いリポジトリに格納される。そのため、PCTE で CASE ツールを統合するためには、目標の CASE ツールのオブジェクトの関係を表現したスキーマを作成する必要がある。PCTE で用いられるスキーマは、スキーマ定義セット (SDS:Schema Definition Set) と呼ばれており、アプリケーション毎に定義し使用する。この時使用している SDS を作業スキーマと呼ぶ[1]。PCTE では、作業スキーマに含まれていないオブジェクトは参照できないという特徴があり、これは作業に不要な情報を隠す (リポジトリのフィルタリ

[1]PCTE 自体もスキーマで表現されており、このスキーマは必ず作業スキーマに加えられていなければならない。

ング) という点で優れている機能である。

また、CASE ツールを統合するための共通で標準的なサービス群は、Ada・C 言語 [4]・C++でプログラムできるように定義されており、今回用いた Emeraude PCTE[2]では C 言語もしくは esh[3]を用いて CASE ツールの統合プログラムを作成することができる。

2.3 PCTE によるツール統合の手順

PCTE をフレームワークにしたツール統合を行なうための手順の概要を以下に示す。

1. スキーマの作成
   ツールが用いているデータモデルをスキーマで表現する。PCTE では、sds_design というグラフィカルなツールが用意されており、これを用いてスキーマの描写を行なう。その後、作成したスキーマのコンパイルを行ない、PCTE の SDS として使用できるようにする。

2. 統合ツールの作成
   CASE ツールと PCTE のデータを統合するためのツールを作成する。

次の節以降では、2 の CASE ツールと PCTE のデータを統合するツールの実現方法について述べる。

2.4 CASE ツールのデータインタフェースの特徴

PCTE を用いて CASE ツールのデータ統合を行なうには、統合する CASE ツールのデータインタフェースを知る必要がある。すなわち、どのような方法でデータを蓄え、またそのデータをどのようにアクセスしているかについてである。このことが分からないと、CASE ツールのデータを統合することはできない。

1. UNIX ファイルシステムの使用
   データを蓄える部分に、UNIX ファイルシステムのディレクトリとファイルの構成をそのまま利用している。システム構成が簡単な分、ツールのデータ管理は複雑なことを行なうことはできないが、ユーザのデータアクセスは容易に行なうことができる。

[2]フランスの Emeraude 社が開発した PCTE で PCTE 1.5 に基づいている。

[3]Emeraude PCTE で使用されるシェル。

2. データベースの使用

データを蓄える部分に、独自のデータベースを使用している。データベースにはデータベース管理システム (DBMS) を使用していることが多い。その理由として、1. 複数のユーザが同時にデータベースにアクセスでき、共用できる。2.SQL 言語等のアプリケーションプログラムを使用してデータの管理ができる。3.(閉じられた中で[4]) データの共用ができる。という点があげられる [3]。そのため、ユーザがデータにアクセスするためには、SQL 等の言語を使ってデータベースに問い合わせる形になる。

3. 1 と 2 の複合

1 の UNIX ファイルシステムと 2 のデータベースの双方を用いている。先にあげたそれぞれの利点を生かして、個人のデータは 1 の UNIX のファイルシステムを利用して、またグループのデータは 2 のデータベースを利用して管理するような CASE ツールも存在する。

2.5 PCTE によるデータ統合の方法

PCTE によるデータ統合の実現には、統合する CASE ツールの制約のためにさまざまな方法が考えられる。以下に統合方針とその長所・短所を示す。

2.5.1 PCTE ネーティブな CASE ツールの作成

PCTE では、ツール作成のためのプログラミング環境[5]が用意されている。それらを用いて新たに PCTE 上で動く CASE ツールを作成する。この方法は実行速度や効率の面から、最も良い統合方法である。しかしながら、新たにツールを作成するのと同等の労力が必要であり、また現在使用しているツールを放棄するといった面からはあまり現実的な方法ではない。

2.5.2 CASE ツールの移植

PCTE では、リポジトリ操作のために表 2.1 の contents 操作関数が用意されている。これらの関数はすべて、unix の file 操作関数に対応した物が用意されており、統合する CASE ツールのファイル操作に関する部分を PCTE のコンテンツ操作関数に置き換えて使用することができる。すなわち、ファイル操作関数を PCTE のコンテンツ操作関数に置き換

[4]ここで言う「閉じられた中」とは、ツールの連動を考慮して開発されたアプリケーションのことである。

[5]Emeraude PCTE では esh と C 言語の 2 つの環境が用意されている。

表 2.1: contents 操作関数とそれに対応する file 操作関数

contents 操作関数	file 操作関数
contents_close	fclose
contents_copy_from_foreign_system	-
contents_copy_to_foreign_system	-
contents_get_position	ftell
contents_open	fopen
contents_read	fread
contents_seek	fseek
contents_set_position	fseek
contents_write	fwrite

えて再コンパイルを行なうことによって PCTE のリポジトリを直接操作することができるようになる。しかしながら、この方法では CASE ツールのソースコードが必要であり、ソースコードがほとんど公開されていない商用の CASE ツールでは適応することができない。

2.5.3 リホスティング

通常 unix 上で動作する実行ファイルは、実行時に必要な全ての資源を持っているのではなく、必要な libraly を実行時にリンクしている。そこで、実行時にファイル入出力に関する UNIX の shared library を PCTE の contents 入出力に関する shared library に置き換えて実行することによって、PCTE のリポジトリにアクセスすることができるようになる。しかしながら、この方法では、Dynamic に library とリンクされているツールのみで使用でき、Static にリンクされているツールでは使用できない。この場合は、実行ファイルを作る際のリンク時に、PCTE のライブラリに置き換えてリンクする必要があるが、元のツールのオブジェクトコードが必要になる。

2.5.4 CASE ツールのカプセル化

既存の unix ツールが、PCTE のリポジトリを操作できるように用意をした後に unix ツールを実行する方法をカプセル化 (encapsulation) という。この方法は、今までの方法に比べ効率は悪いが、もとのツールの実行ファイルのみ存在すれば PCTE との統合が可能である。そのため、このカプセル化にはいくつかの方法がある。

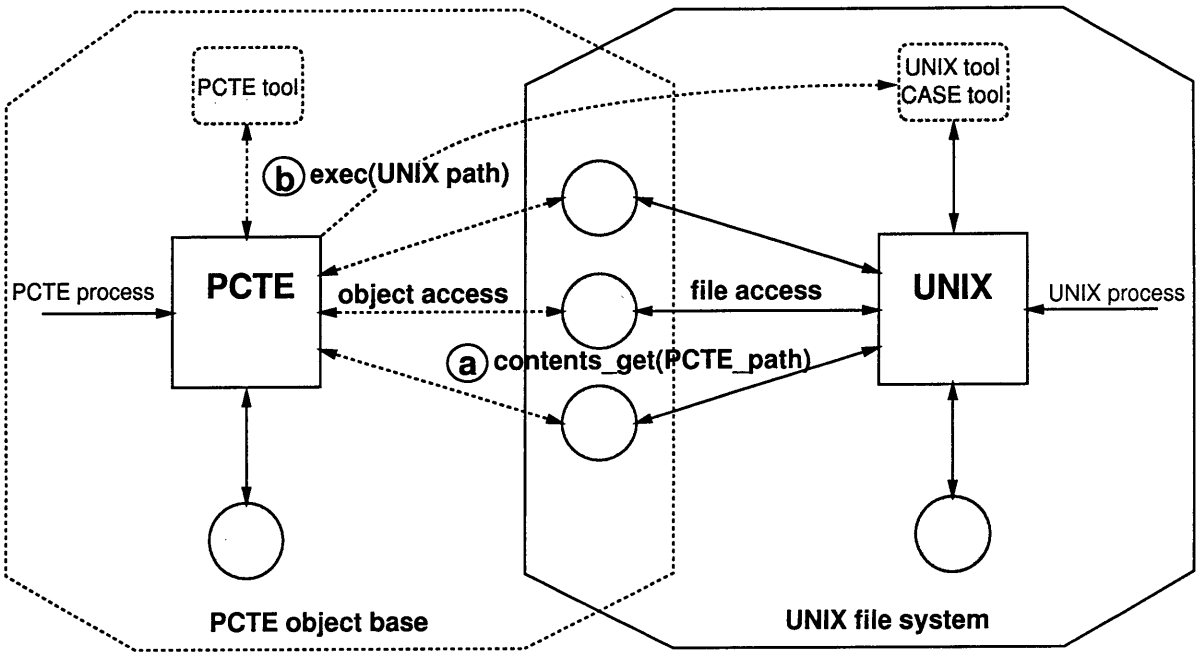

図 2.3: contents_get を使用したカプセル化ツールの概念

1. **contents_get 関数の使用**

   PCTE で用意されている contents_get 関数を用いることによって統合を行なうことができる。図 2.3に contents_get 関数を用いたカプセル化ツールの概念を示す。contents_get 関数は、PCTE のパス名を UNIX のパス名に変換することができる[6]。カプセル化ツールは、

   (a) 使用するファイル名を PCTE のオブジェクトのパス名から UNIX のファイルのパス名に変換する。

   (b) カプセル化ツールの内部から UNIX のファイル名を引数にして実行する。

   という手順で作業を行なうことになる。このような作業手順から contents_get 関数を用いたカプセル化ツールは、エディター等の単一のファイルを操作し、また実行時ファイル名を引数にして実行できるツールが最も適している。しかしながら、PCTE のパス名が予め明らかになっているツール以外では使用するのが困難である。

2. **CASE ツールのインタフェースプログラミング言語の使用**

   CASE ツールのデータベースを操作するためのインタフェースプログラミング言語

[6]PCTE のオブジェクトは Emeraude PCTE の実装上 UNIX のディレクトリとファイルで実現されている。そのため、UNIX 側から PCTE のオブジェクトを操作することが可能となっている。

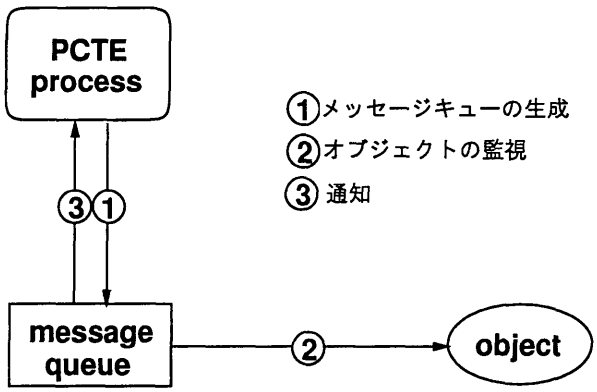

図 2.4: PCTE 通知体の概念

が CASE ツールに用意されている場合がある[7]。これを利用して PCTE リポジトリと CASE ツールデータベース間のデータのやりとりを行なう。この方法は、使用するインタフェースプログラミング言語の操作範囲の限界で制限を受けることになる。

3. **notifier の使用**

   PCTE では、特定のオブジェクトの監視のために通知体 (notifier) を生成することができる。図 2.4に PCTE 通知体の概念を示す。PCTE プロセスが通知体用のメッセージキューを生成することによって、特定のオブジェクトの変更や移動・削除等の監視することが可能になる。これをリポジトリ更新のトリガーにする。同様の機能が CASE ツール側にも存在すれば、notifier を利用したツール統合が可能である。図 2.5に notifier を利用したツール統合の概念を示す。カプセル化ツールでは、notifier を生成してオブジェクトを監視する操作を行なった後に CASE ツールを実行することとなる。

4. **シンボリックリンクの使用 (export/import mechanism)**[8]

   既に存在する PCTE のオブジェクトを操作する場合、contents_get 関数等を使用して、UNIX 側から PCTE のオブジェクトにシンボリックリンクを張ると、PCTE のオブジェクトを直接操作することができる。図 2.6に import メカニズムの概念を示す。カプセル化ツールでは、UNIX 側から PCTE 側へ CASE ツールが使用するオブジェクトに対しシンボリックリンクを作成した後、CASE ツールを実行するという操作を行なう。先に紹介した notifier を利用したもののように、PCTE 側・UNIX

[7]今回使用した StP/OMT には SQL 言語が、ObjectCenter には CLIPC(CenterLine Inter-Process Communication) が用意されている。

[8]この方法は、PCTE 上のデバック環境を使用する際に用いられている方法である。

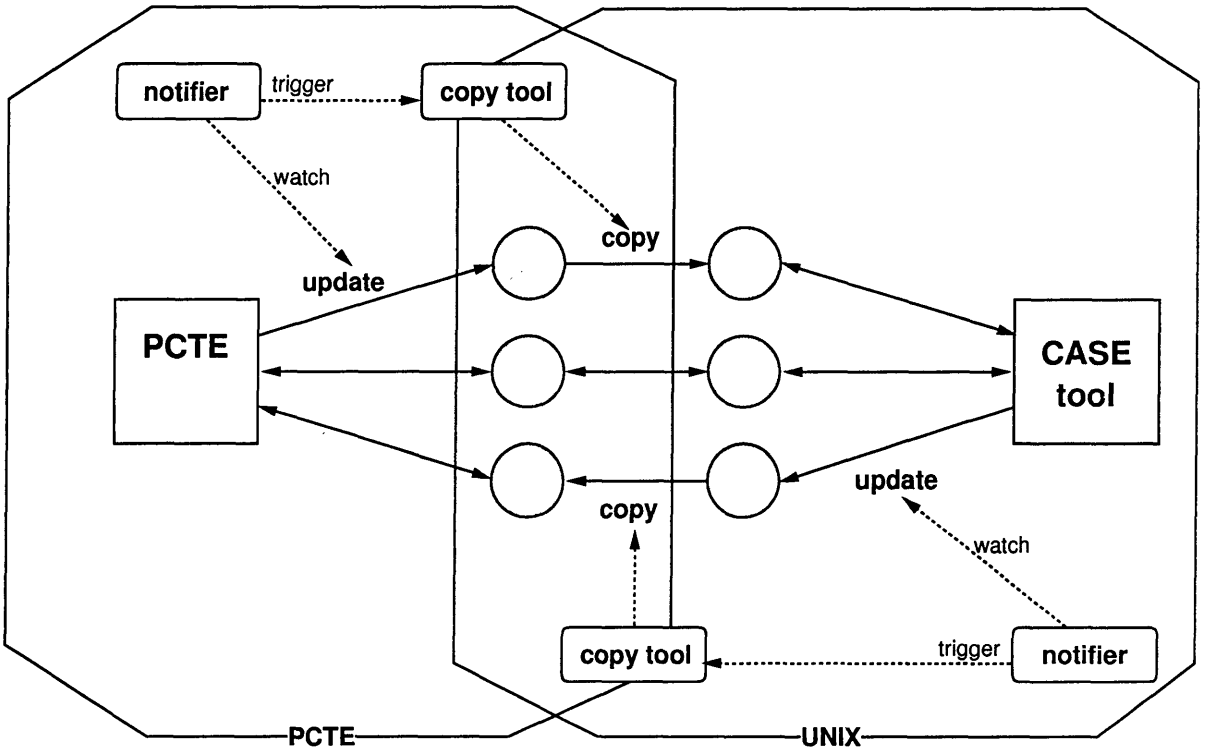

図 2.5: notifier を利用したツール統合の概念

側双方にデータがあるわけではなく、PCTE のリポジトリを直接利用しているので通常と同様に使用することができる。しかしながら、新たにオブジェクトが生成される場合は、PCTE 側にオブジェクトを生成することができない。

export mechanism はこれと反対に、UNIX のリポジトリにアクセスするために PCTE 側から UNIX 側にシンボリックリンクを張る方法である。

2.5.5 Emeraude TCI の使用

TCI サーバーを利用して、メッセージの交換を行ないツールの統合を行なう。基本的に notifier を利用した統合と考え方は一緒であるが、TCI サーバーというメッセージ交換サーバーを介して行なう。ツール連動といったような制御統合を行なう際に利用すると、高度な統合が可能である。このような概念を拡張したものに CORBA(Common Object Request Broker Architecture) がある。

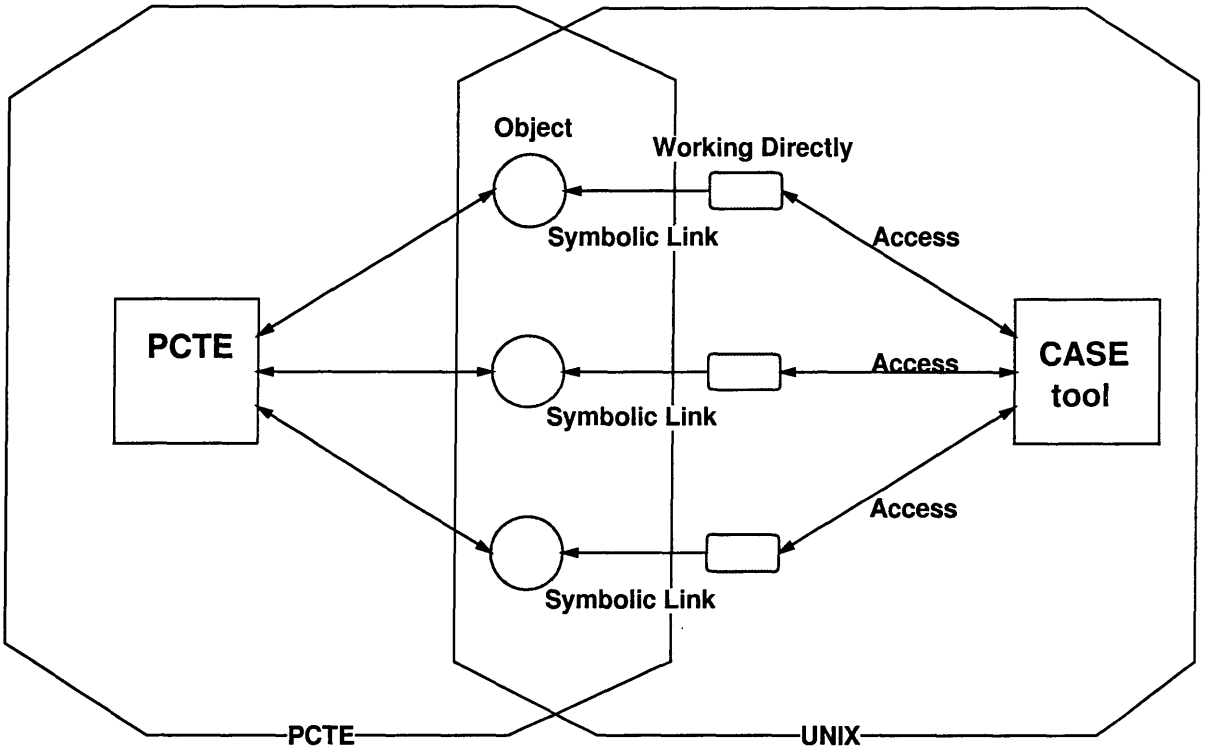

図 2.6: import メカニズムの概念

2.6 バージョン管理の導入

ソフトウェア開発では継続的な変化を伴うため、常にオブジェクトの変化を追跡することが必要になる。すなわち、常に最新のバージョンを特定する、あるいは特定のバージョンまで戻って作業をやり直すといったことが必要となってくる。また、そのオブジェクトはただ一つではなく、例えばソースファイルやドキュメントといった複数のオブジェクトから成り立っており、それらがどのオブジェクトに所属しているかといったような構成を管理する必要もある。このような版管理・構成管理機能が、使用する CASE ツールで提供されている場合は良いが、提供されていない場合は、CASE ツールを使用するユーザが SCCS(Source Code Control System) や RCS(Revision Control Sytem) といったツールを用いて管理しなければならない。

そのため、PCTE ではオブジェクト管理システムの一部としてバージョン管理が用意されている。PCTE のバージョン管理は、様々な段階において作成された複数のコピーからなり、それらは「先行 (predecessor)・後続 (successor) バージョン」という関係で結ばれている。「先行バージョン」は「後続バージョン」より前で作成されたバージョンであり、安定化 (stable) された変更不可能なバージョンである。これらのバージョンは「バージョ

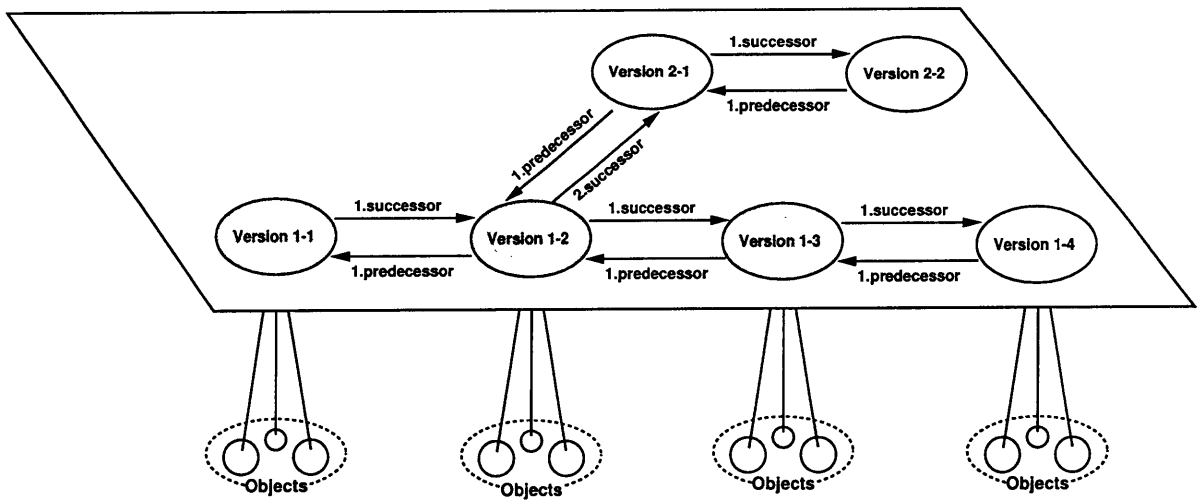

図 2.7: PCTE のバージョン管理の概念

ングラフ」を構成しており、このグラフの中で先行・後続関係を表すリンクで結ばれている[9]。これらをたどっていくことで、オブジェクトの履歴を定めることができる (図 2.7参照)。そこで、構成管理をするオブジェクトとプログラムソースやドキュメントといった複数のオブジェクトの間でリンクを生成することで、PCTE では版・構成管理を行なうことが可能となる。

ところで、PCTE のオブジェクト管理システムでは、リンク名の自動的な名前付けの機能を有している。これは、+という特別な値をリンク名に使用することで、最大の整数値を暗示するものである。すなわち、version.1.object と version.2.object という 2 つのオブジェクトが存在していた場合、version.+.object というのは version.2.object を指示している。また、++という値も存在し、これは最大の整数値に 1 を足した整数値を暗示している。すなわち、先ほどの例で version.++.object は version.3.object を指示していることになる。これらの機能は、バージョン管理を行なう際に最新のオブジェクトにアクセスする (+の場合)、新しいバージョンを生成する (++の場合) などの操作に用いることができる。

表 2.2に PCTE が提供するバージョン管理の機能を示す。このうち、version_snapshot と version_revise は新しいバージョンのコピーを作成する操作であり、これらの操作が行なわれることによって初めてバージョン付けが行なわれる。これらの機能の違いは、

[9] 1.predecessor,2.successor というようなリンク名で結ばれている。

表 2.2: PCTE のバージョン操作

version_add_predecesor	オブジェクトに新たな先行バージョンを付け加える
version_remove_predecesor	バージョングラフから先行バージョンを取り除く
version_revise	オブジェクトの変更可能なバージョンを生成する
version_snapshot	オブジェクトの安定なバージョンを生成する
version_test_ancestry	あるオブジェクトに対して別のオブジェクトが先祖バージョンであるか調べる
version_test_descent	あるオブジェクトに対して別のオブジェクトが子孫バージョンであるか調べる

- **version_snapshot**
  一つのオブジェクトの安定な先行バージョンを作成する。作成されるバージョンは、スナップショットを行なった時点のオブジェクトのコピーである。原版のオブジェクトは変更可能なまま残され、スナップショットをとったオブジェクトの方が安定化される。これらのオブジェクトの間に、先行・後続バージョンの関係が生成する。

- **version_revise**
  一つのオブジェクトの変更可能な後続バージョンを作成する。作成されるバージョンは改訂 (revise) を行なった時点のオブジェクトのコピーである。原版のオブジェクトは安定化され、改訂を行なったオブジェクトの方は変更可能である。これらのオブジェクトの間に、先行・後続バージョンの関係が生成する。

であり、図 2.7のバージョンの関係は、改訂で行なったものである。

バージョン管理の機能を持っていない CASE ツールを統合する場合、バージョン管理をカプセル化ツールの中で行なうことで、ユーザはバージョン管理に対し最小の注意を払えば良いようになる。すなわち、

- **最新のバージョンで作業ができる**
  PCTE の方でバージョンを管理しており、ユーザは特に意識しなくても最新のバージョンで作業ができる。

- **容易に以前のバージョンに戻ることができる**
  作業の節目で改訂をとっておくことで、その状態まで戻ることが容易にできる。また、途中から複数のバージョンで作業を行なうことも可能である。

といった機能を追加することが可能になる。

第 3 章

StP/OMT と ObjectCenter の統合

3.1 StP/OMT との統合

StP/OMT(Software through Pictures / Object Modeling Technique) は、グラフィカルなユーザインタフェースを通じてオブジェクトのモデルを分析・定義することをサポートする上流工程の CASE ツールである。StP/OMT は、オブジェクトモデル図からプログラムコード[1]の一部を自動生成することが可能である。

3.1.1 StP/OMT のデータインタフェースの特徴

第2章で述べたように、PCTE を用いてツール統合を行なうには、統合する CASE ツールのデータインタフェースを知る必要がある。図 3.1に StP/OMT の概観を示す。StP/OMT は、リポジトリブラウザ、オブジェクト・動的・機能モデルエディタ、クラス・状態テーブルエディタ等から成り立っており、これらは StP デスクトップというグラフィカルなブラウザで統合されている。また StP/OMT で使用されるツールは、UNIX のファイルシステムと StP/OMT が用意しているリポジトリである StP リポジトリ[2]にデータを格納するようになっている。以下に、StP/OMT がどのようにデータを保存しているかを示す。

- ファイルシステムへの保存方法
  UNIX ファイルシステムのファイルという形で保存され、その内容は各モデルに関する情報が ASCII 形式で保存されている。各ファイルには、オブジェクト・動的・機能モデルエディタで作成されるダイアグラムあるいはクラス・状態テーブルエディタで作成されるテーブルについての情報が含まれている。ダイアグラムファイルでは、シンボルに関する情報を保存し、テーブルファイルではテーブルセルに関する情報を保存している。これらのファイルは、個人の開発環境で使用されている。

[1] C++,Smalltalk,Ada のプログラムコードを生成することができる。

[2] Sybase という RDBMS を使用している。

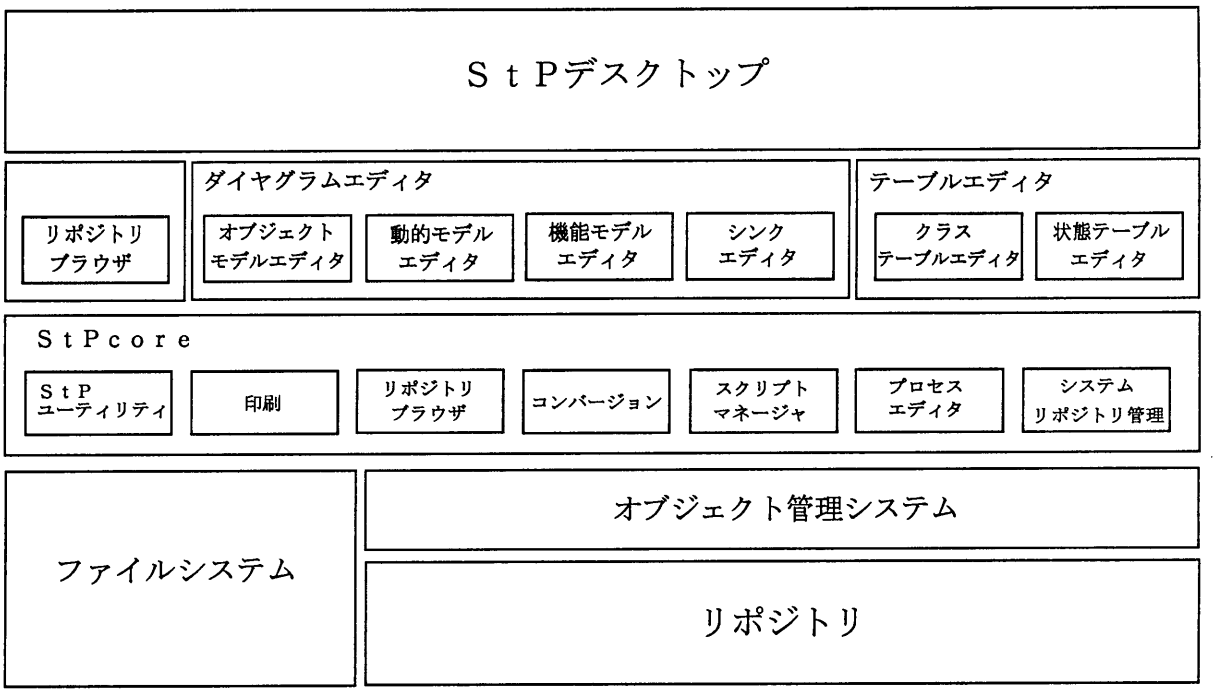

図 3.1: StP/OMT の概観

- **StP リポジトリへの保存方法**
  StP リポジトリデータベースのデータという形で保存され、その内容はダイアグラムとテーブルに関する情報がデータベースの形で保存されている。データベースの更新は、ファイルの状態のデータから行なわれる。このデータベースは、複数のユーザが使用する開発環境で使用される。また、プログラムコードを生成する際に QRL(Query and Reporting Language) という StP の言語を用いて、StP リポジトリから生成するためにも使用される。

また、StP/OMT は、オブジェクトモデル図とクラステーブルからプログラムコードを生成することができる。プログラムコードは、インタフェース (ヘッダ) と実装 (ボディ) に分かれてクラス毎にファイルの形で生成する。

図 3.2に今回統合を行なう StP/OMT のデータの流れを示す。StP/OMT で使用されるオブジェクトモデル・動的モデル・機能モデルとクラステーブルのデータ、そしてそれらから生成されるプログラムコード (C++のソースコードでヘッダファイル.h とボディファイル.cc) を下流の CASE ツールが利用できるように PCTE のリポジトリに格納することが目標である。

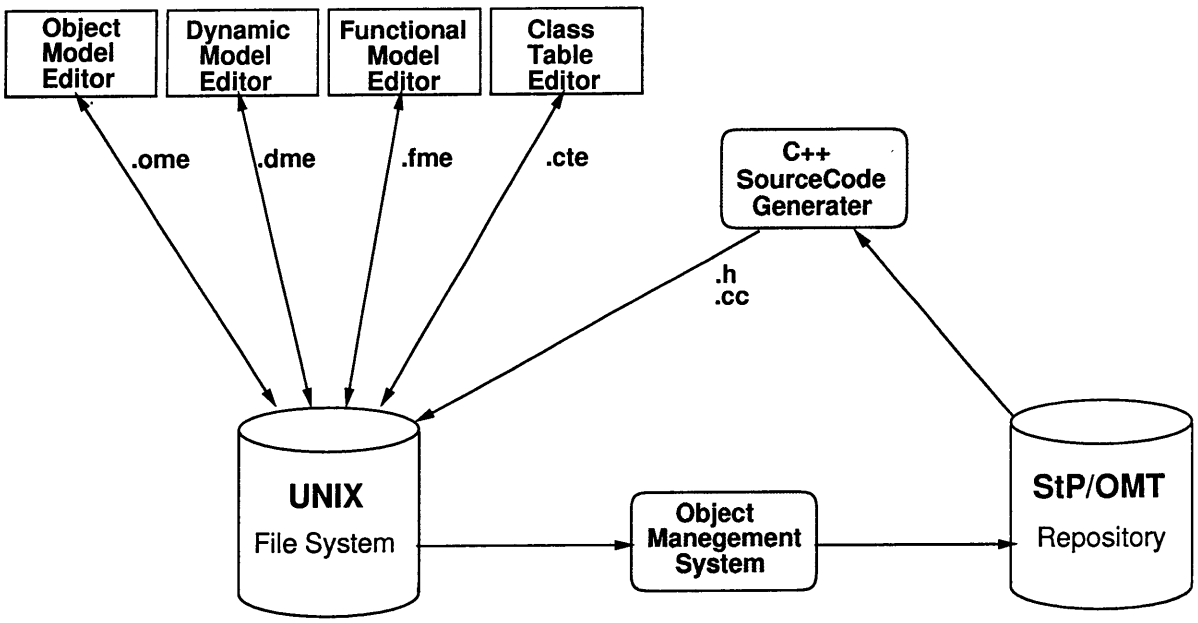

図 3.2: StP/OMT データの流れ

3.1.2 統合の問題点及び解決策

これまでに判明した StP/OMT の特徴から、統合上の問題点及びその解決策を示す。

1. **StP/OMT はソースコードの公開されていない CASE ツールである。**
   StP/OMT は市販の CASE ツールであり、プログラムのソースコードは公開されていない。そのため、今回の統合では、CASE ツールの移植という方法では統合が行なえないのでカプセル化で統合を行なう。

2. **カプセル化ツールで統合する際にファイル名を引数にして実行することはできない。**
   StP/OMT では、複数のファイルで作業を行なうので、起動時に contents_get 関数で PCTE のパス名を UNIX のパス名に変換し、UNIX のパス名を引数にして実行することができない。そのため、今回の統合では、StP/OMT を起動する際に project のあるディレクトリのパス名と system 名を引数にして起動し、StP/OMT によって生成されるファイルを notifier で監視して PCTE のリポジトリに格納するようなカプセル化ツールを作成する。

3. **カプセル化ツールで統合する際に StP/OMT に notifier の機能がない。**
   StP/OMT には、ファイルの生成や更新を監視する notifier の機能がない。今回の統合では、StP/OMT が生成するファイルやファイルの更新を監視する notifier を作成する。

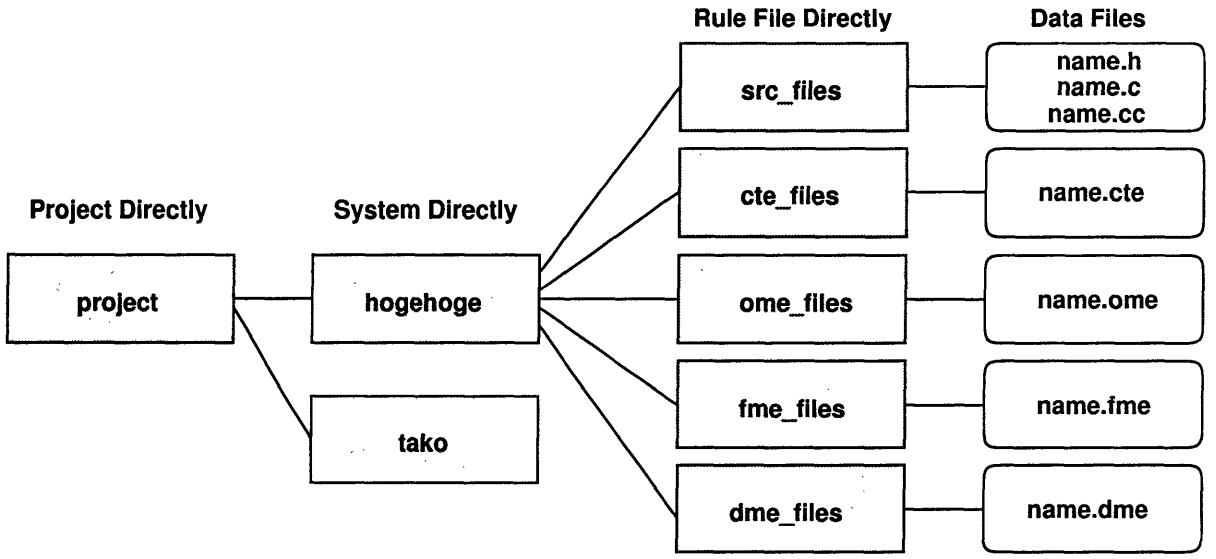

図 3.3: StP/OMT のファイル構成

4. **StP/OMT にバージョン管理の機能がない。**
   StP/OMT はソフトウェア開発で使用する CASE ツールであるにもかかわらず、生産物のバージョン管理の機能がない[3]。今回の統合では、PCTE のバージョン管理の機能を用いて、StP/OMT にバージョン管理の機能を追加する。

3.1.3 スキーマの作成

StP/OMT を統合するためのさまざまな制約を考慮したうえ、PCTE で使用するスキーマの作成を行なう。図 3.3に StP/OMT が生成するファイルの構成を示す。StP/OMT で生成されるファイルは、それぞれの rule file のディレクトリの下に生成する。rule file ディレクトリは system ディレクトリの下に作られ、system ディレクトリは project ディレクトリの下に作られている。

StP/OMT のこのようなファイル構成を考慮して作成したスキーマを図 3.4に示す。この中で、object は sys-object から、file は sys-file から輸入 (import) している。同様に StPOMT_file は file から輸入している。また、user と group はそれぞれ env-user、env-group から輸入している。それ以外のものは、表 3.1に示す。

またこの中で、project_object から system_object へのリンクのキーに k1,k2 を使用しているが、これはバージョン管理のための整数型のキーを用いているためである[4]。それ以外のキーに関しては文字列型のキーを用いている。

[3] オプションで SCCS を導入することができる。

[4] gen-k1,gen-k2 から輸入している。

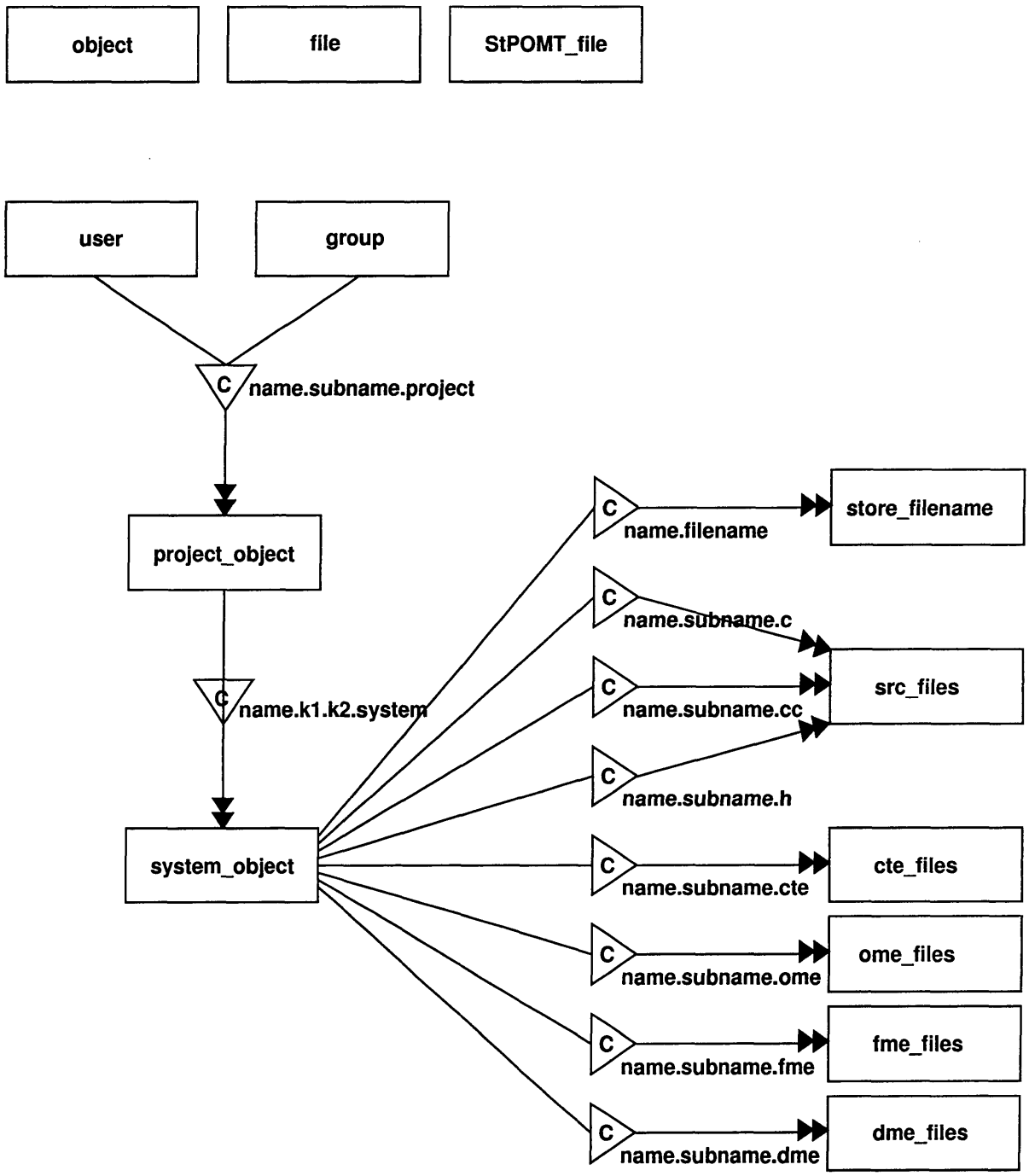

図 3.4: StP/OMT スキーマ

表 3.1: StP/OMT で使用するオブジェクトの型

オブジェクト型名	オブジェクト型
user	env-user
group	env-group
project_object	sys-object
system_object	sys-object
store_filename	StPOMT_file (sys-file)
src_files	StPOMT_file (sys-file)
cte_files	StPOMT_file (sys-file)
ome_files	StPOMT_file (sys-file)
fme_files	StPOMT_file (sys-file)
dme_files	StPOMT_file (sys-file)

また、図 3.3のファイル構成には存在しないオブジェクト store_filename が図 3.4のスキーマ図にはあるが、これは Emeraude PCTE の制限[5]によるもので、ここには PCTE リポジトリに格納されているオブジェクトのパス名 (UNIX のファイル名) を記録している。

3.1.4 StP/OMT の統合方式

図 3.5に、今回作成したツールの概要を示す。StP/OMT のオブジェクトモデルエディタ・動的モデルエディタ・機能モデルエディタ・クラステーブルエディタで作成した作業成果物は、それぞれのツールで「保存 (save)」をすると UNIX のファイルシステムに書き込まれる。また、C++ソースコードジェネレータは、オブジェクトモデルやクラステーブルからヘッダファイル・ボディーファイルを自動生成する。notifier はその書き込みを監視しており[6]、書き込みがあった場合 UNIX ファイルシステムから PCTE リポジトリにコピーを行なう。さらに、新たに生成したファイルも、生成した時点から監視するようになっている。

また、これら生成物は PCTE のバージョン管理機構を用いて管理されており、ユーザが PCTE で統合された StP/OMT を起動する際に、オプションを付けて起動することで

[5]Emeraude PCTE V.12.5.1 まではリンクキーのサイズが 32 文字に制限されていた。V.12.6.1 からは 256 文字に拡張されたが、V.12.5.1 の時にプログラムを作成したのでこのような方法をとっている。

[6]ファイルの書き込み時間を比較している。

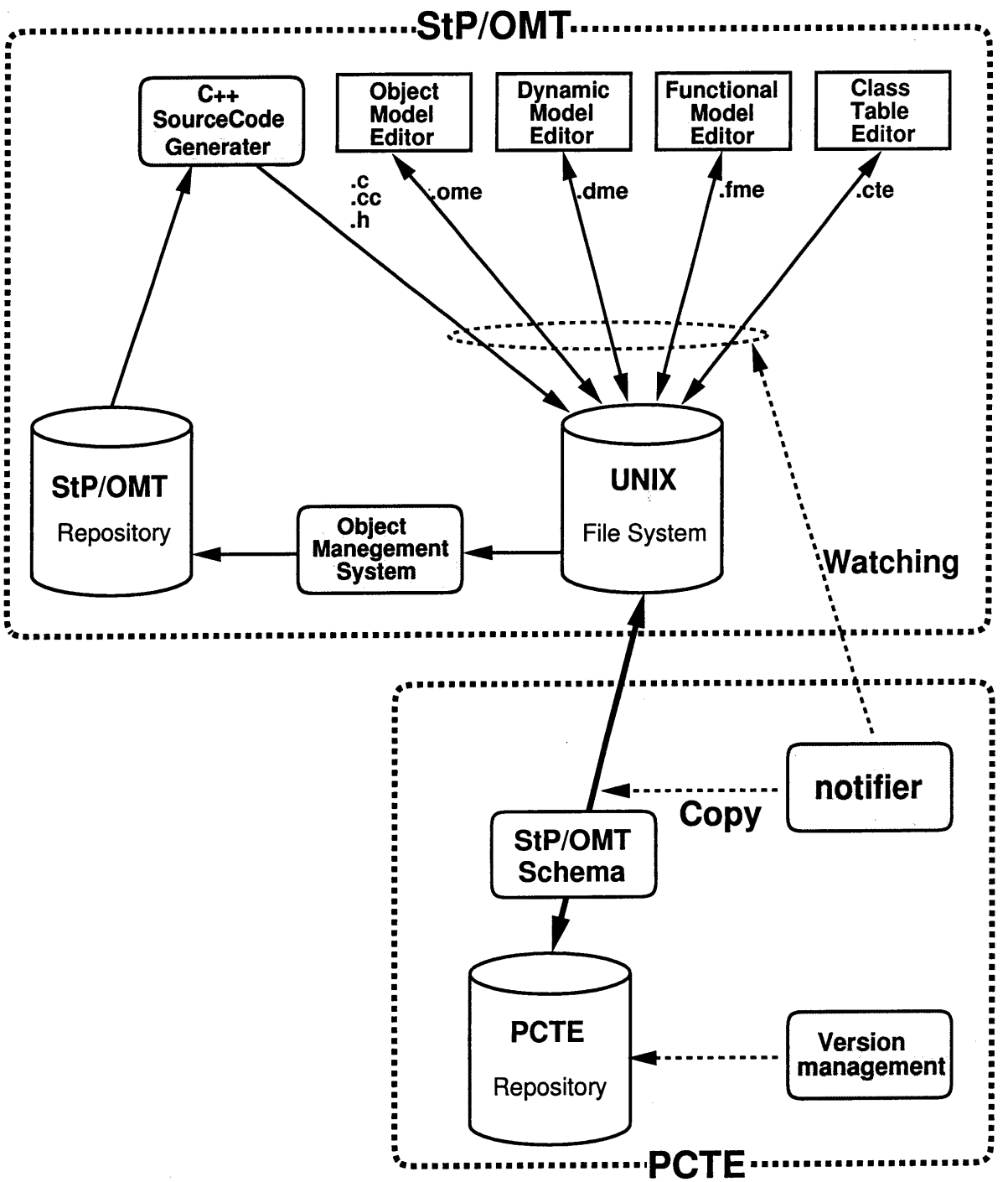

図 3.5: StP/OMT の統合方式

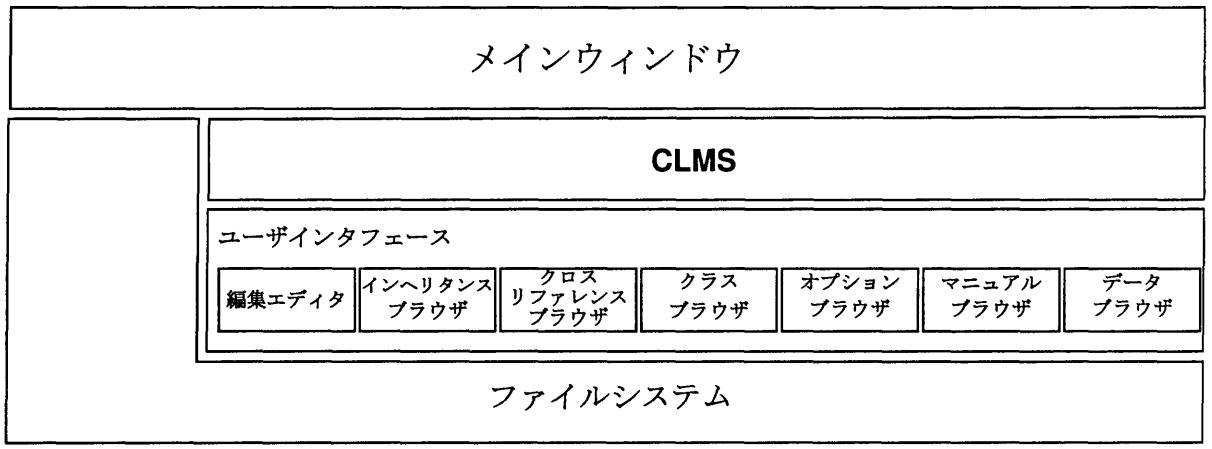

図 3.6: Object Center の概観

新しいバージョンを生成することができる[7]。また、以前のバージョンに戻る際も、オプションを付けて起動することで、以前のバージョンから新しいバージョンを生成することができる[8]。バージョン管理のオプションを特に指定しない場合には、最新のバージョンで起動する。

3.2 ObjectCenter の統合

Object Center は、C++のコード開発・テスト・デバッグを行なうために必要な機能を提供している下流工程の CASE ツールである。Object Center は通常コンパイルして使用する C++のソースコードをインタプリタで扱うことができ、デバッガとあわせて強力な開発環境を提供している。

3.2.1 ObjectCenter のデータインタフェースの特徴

図 3.6に Object Center の概観を示す。Object Center は、編集エディタ、インヘリタンスブラウザ、クロスリファレンスブラウザ等のツールから成り立っており、これらはメインウィンドウと呼ばれる統合ツールから使用することができる。また、これらのツールは CLMS(Center Line Message Server) と呼ばれるメッセージ通信サービスを利用してツール間のデータのやりとりを行なっている。なお、使用する C++のソースコードは、ファイルの形で入力される。

[7] -revise オプションを付けて起動することで、改訂 (revise) される。

[8] -revise < version number >を指定することで以前のバージョンの改訂を作成する。

3.2.2 統合の問題点及び解決策

これまでに判明したObject Centerの特徴から、統合上の問題点及びその解決策を示す。

1. **Object Centerはソースコードの公開されていないCASEツールである。**
   Object CenterもStP/OMTと同様、商用のCASEツールであり、プログラムのソースコードは公開されていない。そのため、今回の統合では、カプセル化技術を用いて統合を行なう。

2. **カプセル化ツールで統合する際にファイル名を引数にして実行することができない。**
   Object Centerでは、クラスごとのヘッダファイル・ボディファイルが必要である。ところで、Object Centerでは新たなクラスは生成されないので、新たなヘッダファイルやボディファイルが生成されることはない。そこで、Object Centerの作業ディレクトリからPCTEのオブジェクトへシンボリックリンクをはることで、直接PCTEのリポジトリを操作することができる。

3. **Object Centerでいくつかの新しいファイルが生成される。**
   Object Centerは、ソースコードをコンパイルするためにmakefileを必要とする。また、現在の環境を保存するためにプロジェクトファイルを生成する[9]。これらのファイルは予めファイル名がわかっているため、カプセル化ツールの中でPCTEのオブジェクトを生成しシンボリックリンクをはっておくことで、PCTEのリポジトリにデータが格納されるようになる。

4. **Object Centerにバージョン管理の機能がない。**
   Object Centerはソフトウェア開発で使用するCASEツールであるにもかかわらず、生産物のバージョン管理の機能がない。今回の統合では、PCTEのバージョン管理の機能を用いて、Object Centerにバージョン管理の機能を追加する。

3.2.3 スキーマの作成

Object Centerを統合するためのさまざまな制約を考慮したうえ、PCTEで使用するスキーマの作成を行なう。図3.7にObject Centerで使用するスキーマを示す。この中で、StP/OMTで使用しているオブジェクトはStP/OMTのスキーマから輸入する。表3.2にObject Centerで使用するオブジェクトとその型(括弧内はオリジナルの型)を示す。

また、キーや型参照子に関してもStP/OMTで使用しているものに関しては、輸入して用いる。表3.3にObject Centerで使用するキーを、表3.4に型参照子を示す。

[9] ocenter.projというファイルを生成する。

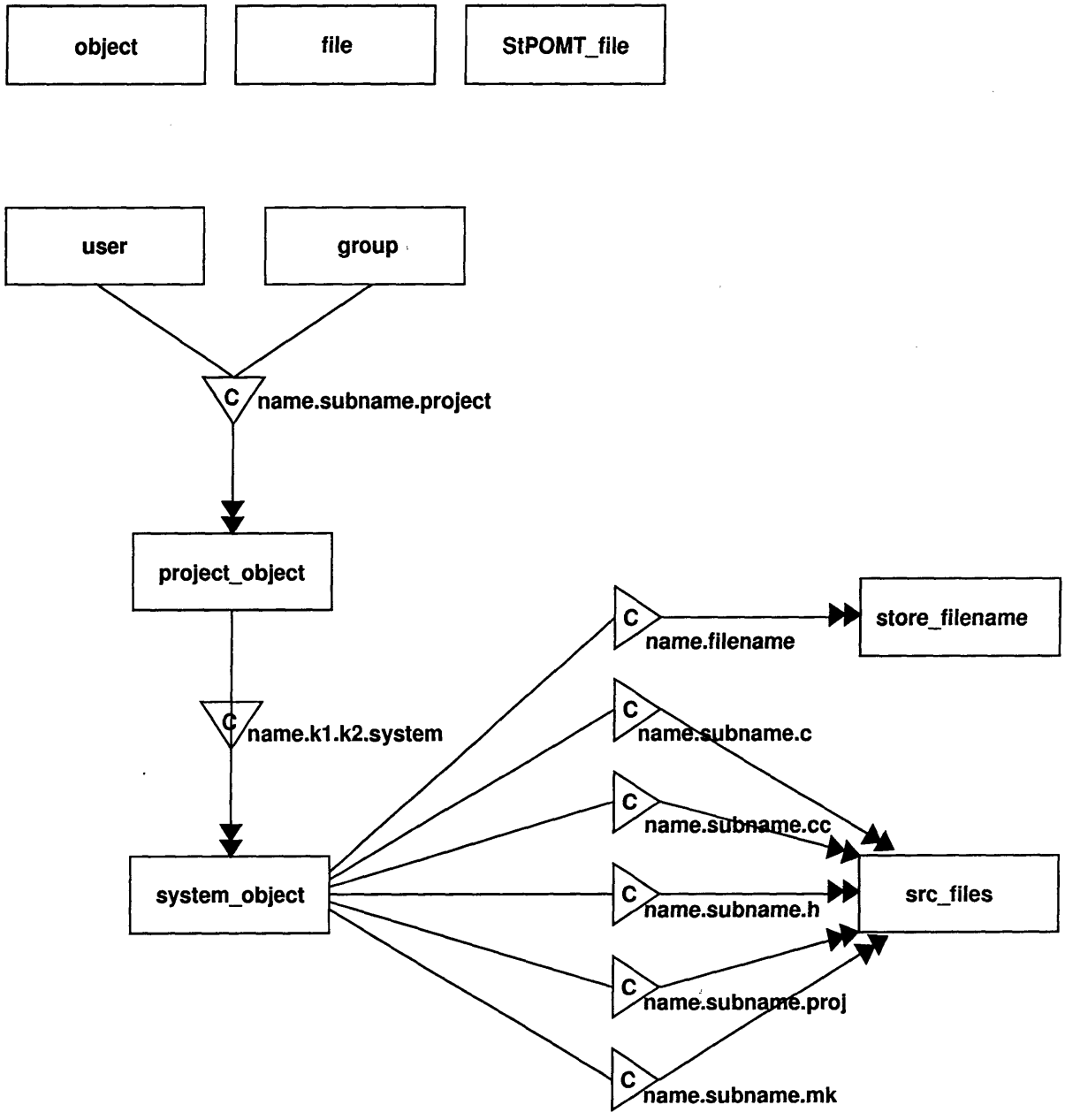

図 3.7: Object Center スキーマ

表 3.2: Object Center で使用するオブジェクトの型

オブジェクト型名	オブジェクト型
user	StP_OMT-user (env-user)
group	StP_OMT-group (env-group)
project_object	StP_OMT-project_object (sys-object)
system_object	StP_OMT-system_object (sys-object)
store_filename	StP_OMT-StPOMT_file (sys-file)
src_files	StP_OMT-StPOMT_file (sys-file)

表 3.3: Object Center で使用するキー

キー名	輸入元 (属性)
name	StP_OMT-name (string)
subname	StP_OMT-subname (string)
k1	StP_OMT-k1 (integer)
k2	StP_OMT-k2 (integer)

表 3.4: Object Center で使用する型参照子

型参照子	輸入元の型名
project	StP_OMT-project
system	StP_OMT-system
filename	StP_OMT-filename
c	StP_OMT-c
cc	StP_OMT-cc
h	StP_OMT-h

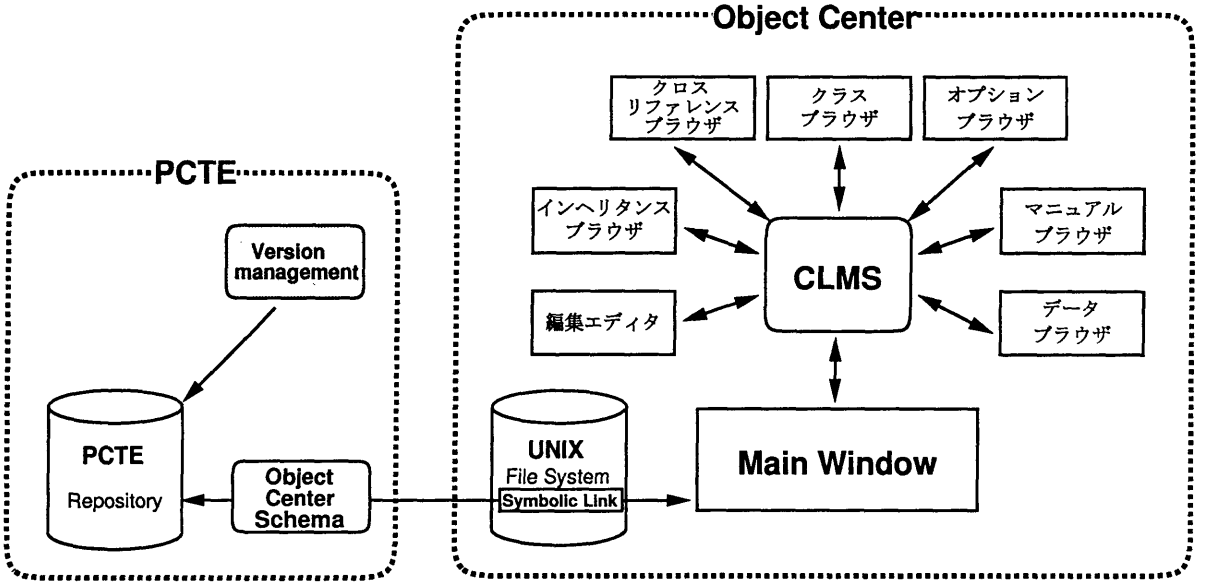

図 3.8: Object Center の統合方式

これらの表で示すように、ほとんどのものはStP/OMT のスキーマから輸入して使っており、新たに生成されるものは makefile とプロジェクトファイルの 2 つだけである。これら 2 つのオブジェクトは、作業スキーマから Object Center のスキーマを外すと (すなわち、StP/OMT のスキーマだけになると) 参照することができなくなる[10]。これは、2.2 で説明したように「リポジトリのフィルタリング」によるものである。

3.2.4 ObjectCenter の統合方式

図 3.8に、今回作成したツールの概要を示す。StP/OMT で生成された PCTE リポジトリにあるプログラムコードは、UNIX のファイルシステム上に作られたシンボリックリンクを通して操作される。そのため、Object Center のツール内部では UNIX 上にプログラムコードがあるのと同様に操作でき、既存のプログラムコードに関しては特に制限を受けない。しかしながら、新たに生成する場合は、PCTE のリポジトリに格納されない[11]。

また、これらの生成物は PCTE のバージョン管理機構を用いて管理されており、ユーザが PCTE で統合された Object Center を起動する際に、オプションを付けて起動することで新しいバージョンを生成することができる[12]。また、以前のバージョンに戻る際も、

[10] StP/OMT で参照する必要がある場合には、StP/OMT のスキーマの方で Object Center のスキーマから輸入するようにする。

[11] makefile とプロジェクトファイルに関しては、事前に生成されることが分かっているので格納される。

[12] -revise オプションを付けて起動することで、改訂 (revise) される。

オプションを付けて起動することで、以前のバージョンから新しいバージョンを生成することができる[13]。バージョン管理のオプションを特に指定しない場合は、最新のバージョンで起動する。

3.3 データ統合の指針

ここでは、これまで行なった統合から、データ統合の際に必要となる手法と、考慮すべき点をまとめる。

3.3.1 スキーマの作成指針

スキーマは、統合するCASEツールのデータ構造を表現したものである。そのため、スキーマを作成するには、統合するCASEツールのデータ構造を詳しく知る必要がある。

- **データの保存方法**
  ディレクトリとファイル・データベース管理システム(DBMS)など、どのようにしてデータが保存されているかを調べる。

- **データの操作方法**
  ファイルアクセス(ファイルをオープンして操作する)・問い合わせ言語(SQL言語など)の使用など、どのようにしたらデータを操作できるかを調べる。

- **データの流れ**
  どのようなデータがどのツールから生成し、どのような名前で保存されるか、また、どのようなデータがツールを操作するのに必要であるかを調べる。

これらを調査した後に、CASEツールのデータ構造をPCTEのリポジトリに格納できるようにマッピングを行ない、スキーマを作成する。なお、バージョン管理の機能を追加する場合には、構成管理を行なうオブジェクトを作成し、それらの下にオブジェクトを格納するようなスキーマを作成する[14]。

3.3.2 ツール作成の指針

CASEツールのデータインタフェースの特徴によって、ある程度の方針を示すことができる。ここでは、CASEツールのソースコードの存在の有無とデータの保存方法の視点から、どのような統合が最適であるかについて以下に示す。

[13] -revise < version number >を指定することで以前のバージョンの改訂を作成する。

[14] StP/OMTの場合は、system_objectを併用する形で実装されている。

1. **ソースコードを公開している CASE ツールの場合**

   この場合、CASE ツールの移植が最も最適である。CASE ツールが用いているデータモデルを SDS で表現し、ソースコード中のファイル入出力関数を PCTE の contents 関数に置き換えて再コンパイルを行なう。再コンパイルによって生成されたツールは、UNIX 上で動くツールと同様に振舞い如何なる制限も受けない。

2. **UNIX ファイルシステムを使用している CASE ツールの場合**

   新たにオブジェクトが生成される場合と、既存の PCTE のオブジェクトを利用する場合で統合の方針が異なる。

   (a) **新たにオブジェクトが生成される場合**

   ツール統合の中で難しいケースである。特に、新たに生成されるオブジェクトのパス名が明らかになっていない場合は最も難しい。

   i. **パス名が明らかなツールの場合**

   エディタのように起動前にパス名が明らかなツールは、カプセル化ツール中で PCTE のオブジェクトを生成し、その後 contents_get 関数で UNIX のパス名を知り、UNIX のパス名を引数にしてツールを起動することによって直接リポジトリを操作することができる。

   ii. **パス名が明らかでないツールの場合**

   直接リポジトリを操作することが困難なため、UNIX のファイルを介して操作することになる。この場合、生成されたファイルに対して更新を監視する notifier(通知体) を生成し、notifier の更新通知をリポジトリ更新のトリガにしてファイルから PCTE リポジトリにオブジェクトをコピーすることになる。

   この他にも、入出力が単純な場合は、shared library を置き換えるリホスティングや、目的のファイルにフックをかけリポジトリの更新を行なう Emeraude TCI の使用も可能である。

   (b) **既存の PCTE のオブジェクトを利用する場合**

   UNIX 側から PCTE 側にシンボリックリンクをはることによって直接オブジェクトの操作が可能である。また、リホスティングも可能である。

3. **データベースを使用している CASE ツールの場合**

   データベース自体を PCTE のリポジトリに取り込む方法と、データベースが持っているインタフェースプログラミング言語を使用して必要なデータを PCTE のリポジトリにコピーをする方法が考えられる。

4. **UNIX ファイルシステムとデータベースを複合して使用している CASE ツールの場合**

   1 のファイルシステムの場合と 2 のデータベースの場合の双方をうまく利用して統合する。どの方法が良いとは一概に言えず、ケースバイケースになるであろう。CASE ツールのデータインタフェースを良く見極めた上で統合を行なう必要がある。

ソースコードの有無、データの保存方法の上にデータの操作方法・データの流れまで考慮すると CASE ツールのデータ統合は、CASE ツールに合わせた統合ツール (カプセル化ツール) を作成しなければならないという結論に達する。

しかしながら、統合ツールはこれまでに挙げた統合技術を複合して使用することでほとんどの場合、対応できるといえる[15]。

[15] Object Center との統合では 2-b の方法と 2-a-i の方法を複合して使用している。

第 4 章

統合ツールの評価

4.1 「家計簿シミュレーションシステム」による開発

作成したCASE統合ツールを用いて、簡単なソフトウェア開発を行なってみる。開発したシステムは「家計簿シミュレーションシステム[1]」である[5]。

4.1.1 OMT法によるモデリング

StP/OMTは、James Rumbaughのオブジェクト指向分析・設計方法論[3]を基に開発の支援するCASEツールである。従って、家計簿シミュレーションシステムの分析はこの方法論を用いて行なう。OMT法では、まず最初に対象世界の問題分析を行ない、次にモデリングを行なう。モデリングでは、まず最初にオブジェクトモデルから定義する。図4.1に家計簿シミュレーションのオブジェクトモデル図を示す。家計簿シミュレーションシステムでは、3つのクラス「収支の発生源」「家計簿」「お金の管理場所」があり、それらは図に示すようなサブクラスを持っている。例えば、「銀行」はお金を管理しているところの一つであり、「お金の管理場所」のサブクラスの一つである。また、それらのクラスの間にはある関連が存在しており、属性をもってつながっている。例えば、収支が発生したら家計簿に記録しなければならない。その記録の内容は、月日・氏名・金額・銀行名・口座番号・事象・操作・倹約可能項目である。

このような、オブジェクトモデル図を作成した後、次に動的モデルのモデリングを行なう。図4.2に家計簿シミュレーションシステムの動的モデルを示す。動的モデルは、問題世界に存在するオブジェクトの状態変化を示したものであり、この図において●は初期状態を●に○で囲んだものは最終状態を示している。シミュレーションを開始し、初期状態から「次の日を待つ」状態に移行すると、条件によって次の移行先が異なる。例えば、月日が指定月日を超えていたら実績報告書を作成して最終状態に移行する。また月日の日が

[1]本大学のソフトウェア設計論の講義で行なわれたものを今回再度行なった。

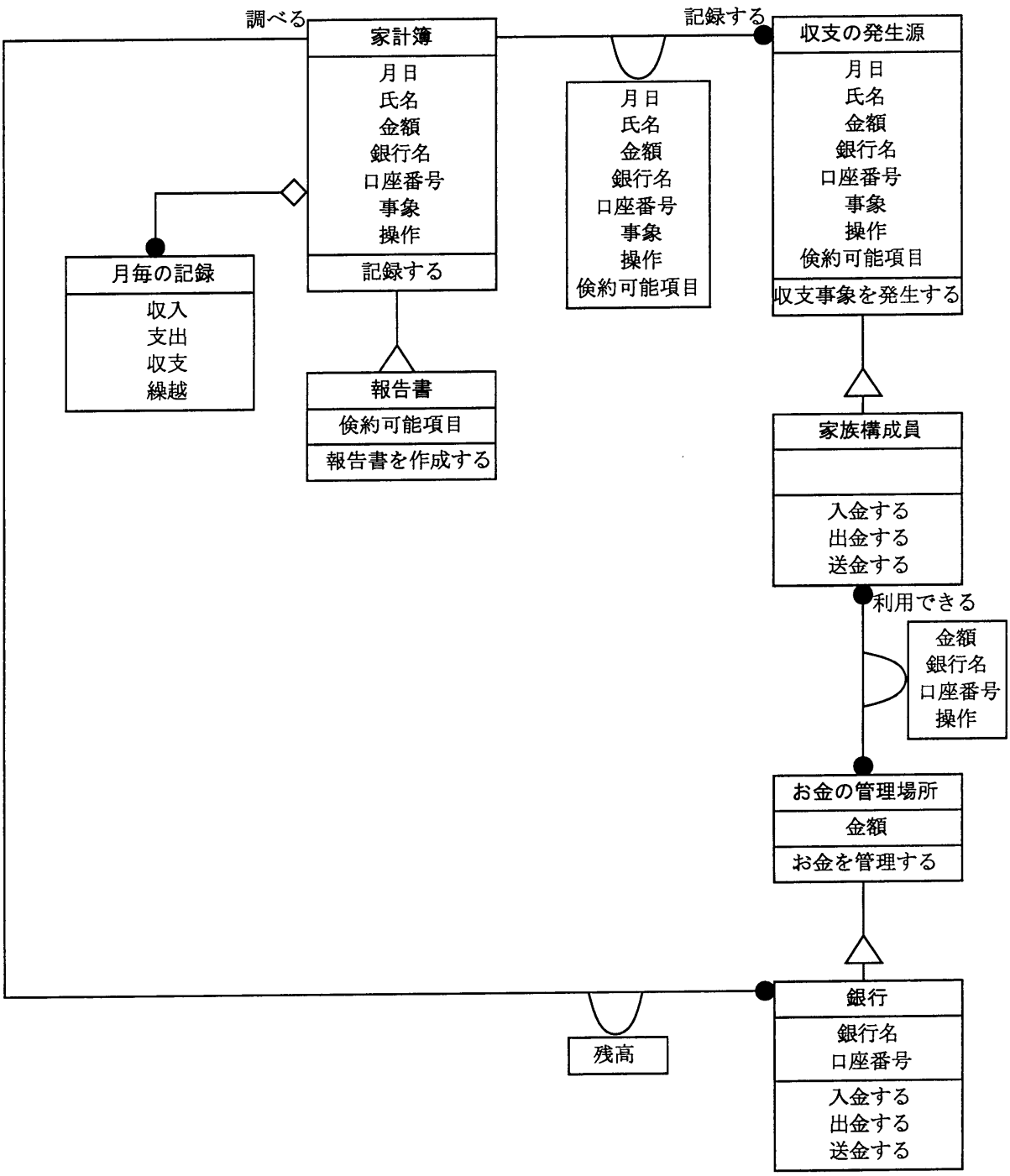

図 4.1: 家計簿シミュレーションシステムのオブジェクトモデル

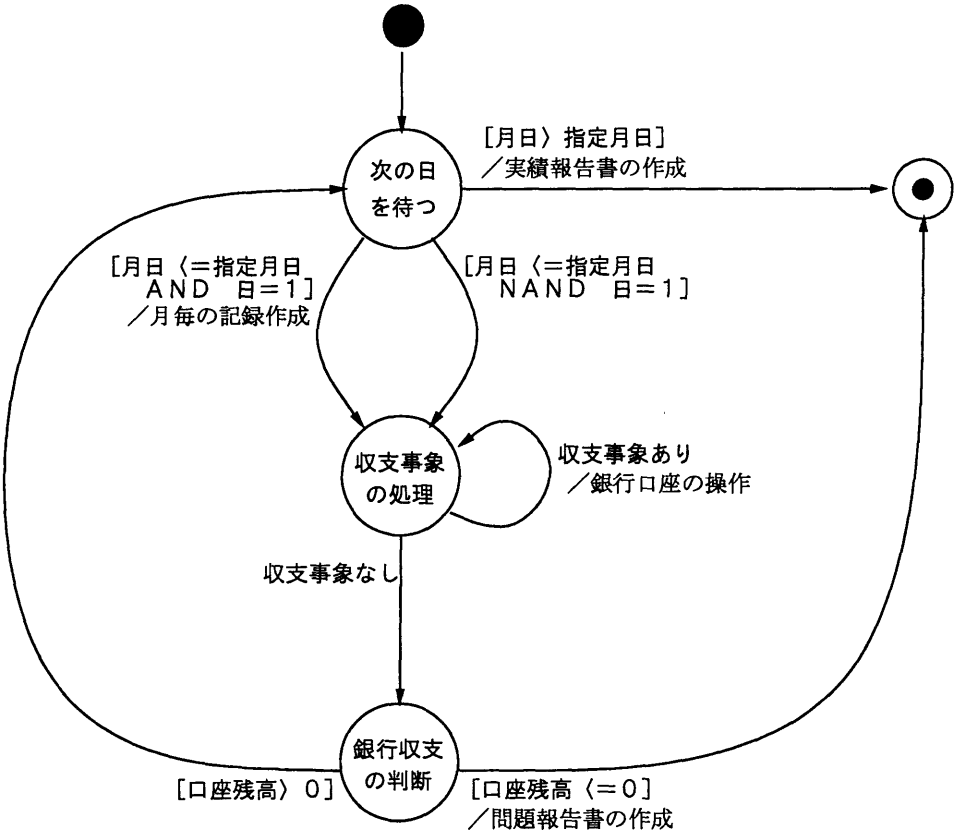

図 4.2: 家計簿シミュレーションシステムの動的モデル

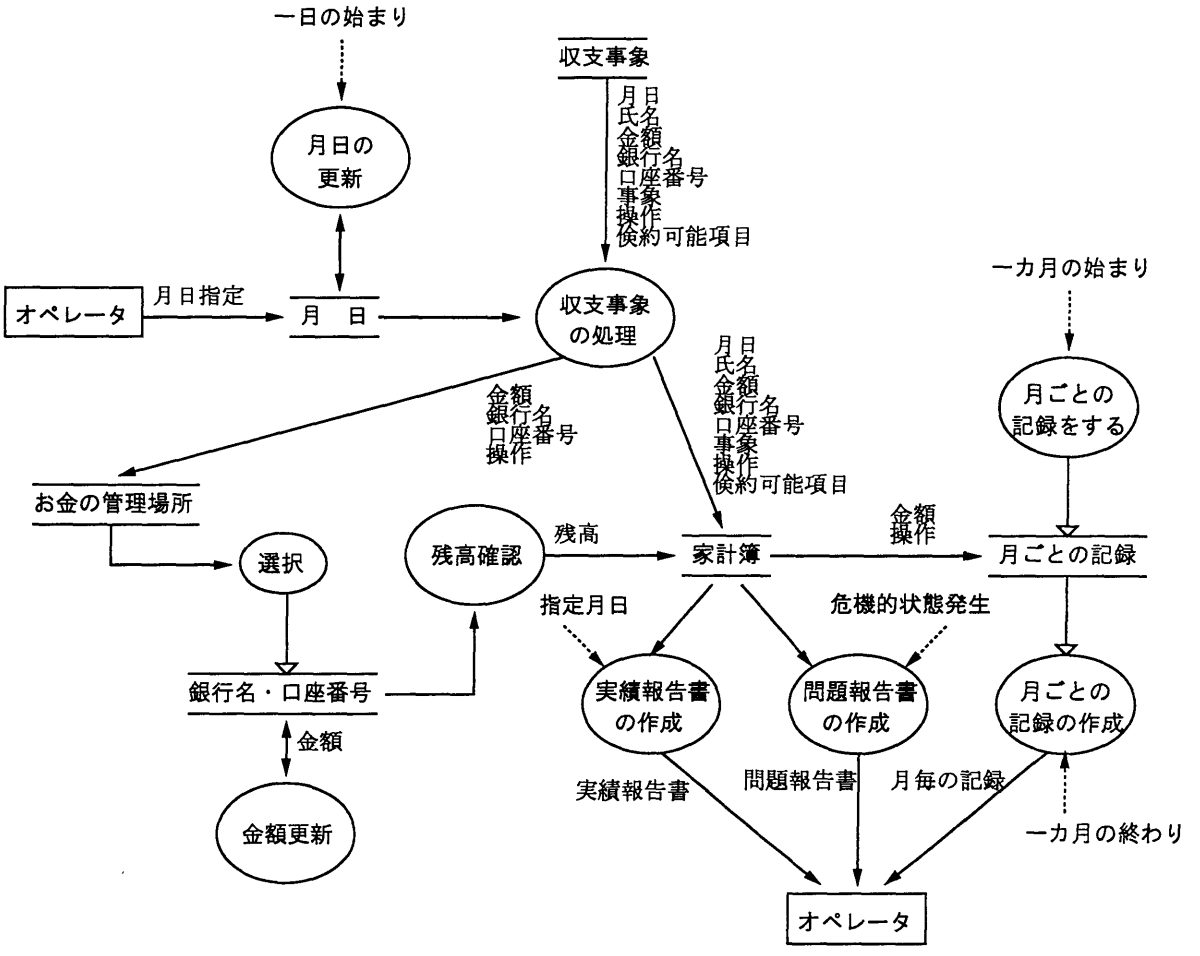

図 4.3: 家計簿シミュレーションシステムの機能モデル

1 日であったら月毎の記録を作成し、収支事象の処理に移行する。それ以外の場合はそのまま収支事象の処理に移行する。

動的モデルを作成した後に、最後に機能モデルのモデリングを行なう。機能モデルは、計算の出力値が入力値からどのように計算されるかを示したものであり、一般にはデータフロー図と呼ばれている。図 4.3に家計簿シミュレーションシステムの機能モデルを示す。家計シミュレーションで起こり得るプロセスを楕円で、データフローを実線の矢印で、制御フローを点線の矢印で、データストアを二重線で、オブジェクトの生成を三角矢印で、アクターを四角で表す。オペレータが月日を指定すると月日をストアした後、収支事象を得て収支事象のプロセスを行なう。収支事象のデータはお金の管理場所から銀行と家計簿から月ごとの記録へ流れ、データとして蓄えられる。これらの作業を繰り返しているうちに「指定日時」「危機状態発生」「1 カ月の始まり」「1 カ月の終わり」というイベントがどれか発生した場合、それぞれのプロセスを行なうようになっている。

4.1.2 統合ツールによる実装

問題からモデリングを行なったモデル図を用いてStP/OMTで開発を行なう。図4.4にStP/OMTを用いて作成し、生成したオブジェクトモデル図を示す。この図の作成には、StP/OMTのオブジェクトモデルエディタを用いて図4.1のオブジェクトモデルを描写する。その後、クラステーブルエディタを用いてクラスの属性や操作の情報を与える[2]。図4.5にStP/OMTのクラステーブルエディタの例を示す。クラステーブルで与えた情報は、StP/OMTのオブジェクトモデルに反映され、それぞれの属性の型や初期値、操作の引数や返り値の型がモデル図に記載される。

このような作業の後、StP/OMTでC++のソースコードが生成する。StP/OMTでは、オブジェクトモデルの各クラスに対してインタフェースファイル(classname.h)と実装ファイル(classname.cc)が生成する[3]。これらのファイルに対して、操作を実装していく。

この実装は、下流工程ツールにあたるObjectCenterを用いて行なう。StP/OMTで生成したインタフェースファイルは図4.6、実装ファイルは図4.7のようになっており、

// stp/omt codeと// stp/omt code endの間に実際の操作を実装することになる[4]。これらを実装した後に、コンパイルを行ない、実行ファイルを生成する。なお、クラスの変更(新たなクラスの追加や既存のクラスの削除)や属性・操作の変更(追加や削除等)を行なう場合は、StP/OMTに戻ってオブジェクトモデル図の変更から行なわなければならない。

4.2 統合ツールの効果と問題点

ここでは、これまで統合ツールを使用した結果から、統合ツールを利用した時の効果と問題点について考えてみる。

4.2.1 統合ツールの効果

1. データの統合

   PCTEをCASE環境の中心に据えることにより、CASEツールがそれぞれ保有しているデータをPCTEのリポジトリに蓄えるという形で統合できる。このため、CASEツール同士のデータ変換がなくなり、データの汎用性が増す。すなわち、PCTEのカプセルツールでCASEツール固有のデータから標準データに変換、またはその逆

[2]型(int,charなど)・初期値や引数・返り値の型(void,int,charなど)を設定できる。

[3]生成されたコードは標準ANSI C++3.0準拠のものである。

[4]これらのコメントは、StP/OMTで再度プログラムコードを生成する際に、既に実装した部分を保持するために使用される。

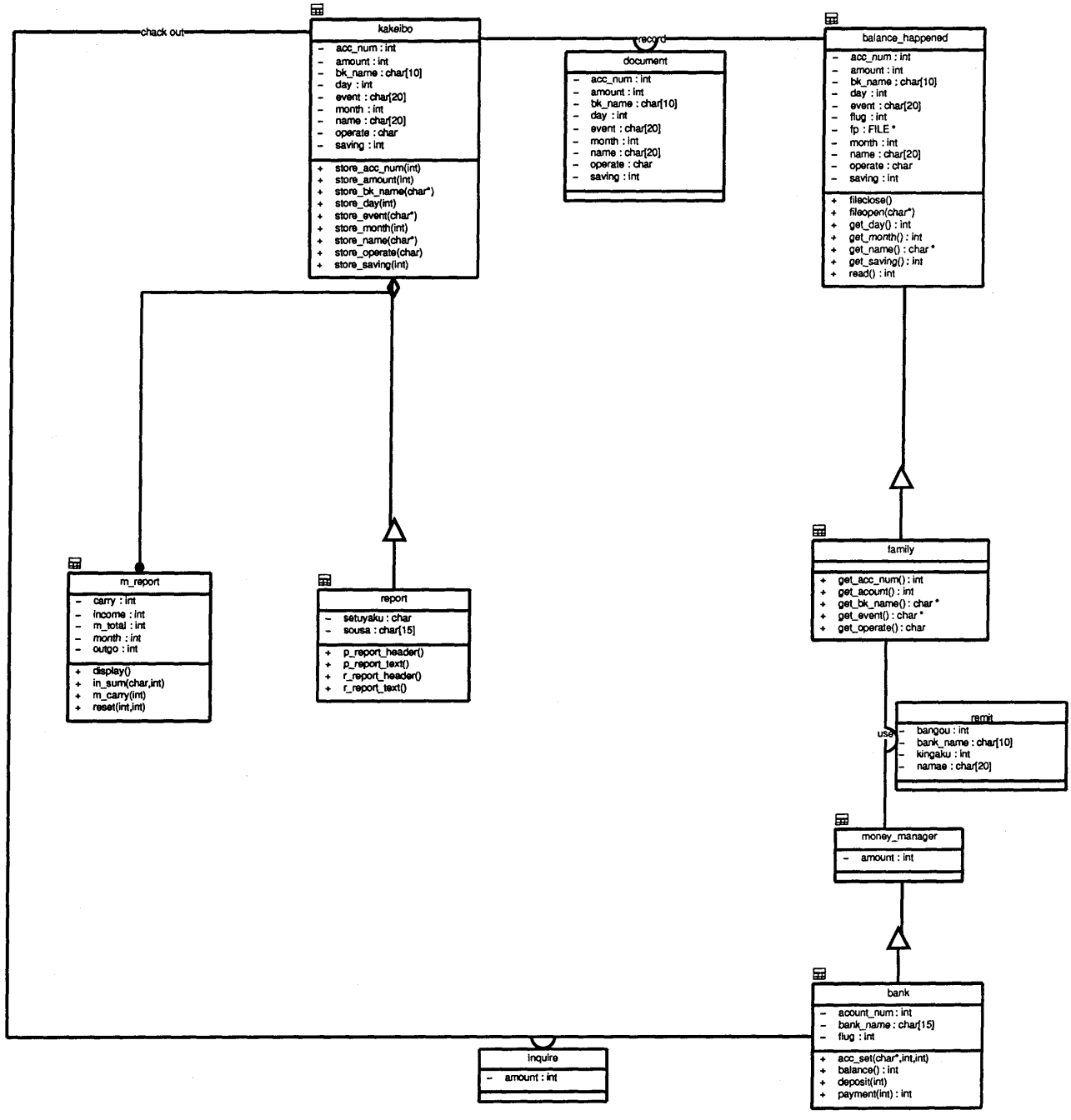

図 4.4: StP/OMT で作成したオブジェクトモデル図

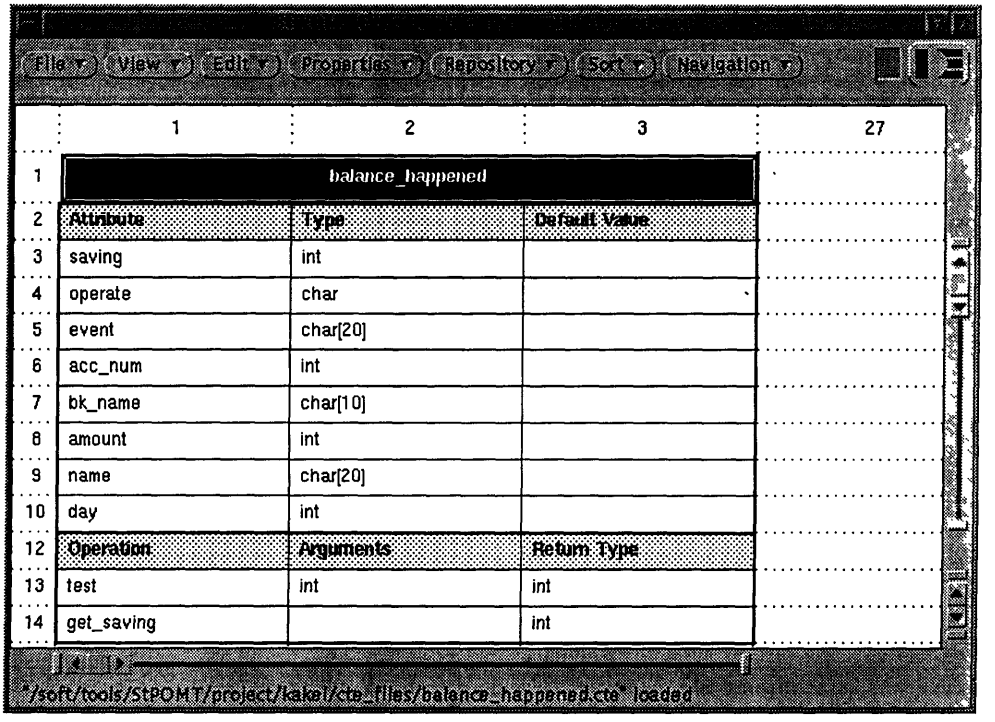

図 4.5: StP/OMT のクラステーブルエディタ

の操作を行なうことでCASE ツールのデータ構造に依存しないようになる[5]。また、それぞれのツールことにスキーマを用意するため、不要なツールのデータを参照することはできない (スキーマによるリポジトリのフィルタリング効果)。そのため、作業中のオブジェクトの構造が簡素になっているので、作業者が注目するオブジェクトの数が減少しており、見通しの良いデータ構造になっている。

2. バージョン管理の導入

今回作成した統合ツールでは、PCTE のバージョン管理を導入し、生産物のバージョン管理を行なっている。そのため、ユーザが意識しなくても常に最新のデータを利用することができる。また、以前のデータまで後戻りをして、そこから新たにデータを生成することが容易になる。PCTE による CASE ツール統合では、バージョン管理のない CASE ツールにバージョン管理機能を容易に追加することができる。

4.2.2 統合ツールの問題点

1. notifier によるパフォーマンスの低下

[5]今回のツールではこのような変換は行なっていないが、カプセル化ツールの利点の一つであろう。

```
#ifndef _balance_happened_h_
#define _balance_happened_h_
// stp/omt class declarations
class balance_happened;
// stp/omt class declarations end
// stp/omt class definition 5004
class balance_happened
{
// stp/omt class members
public:
     void fileclose(void);
     void fileopen(char*);
     int get_day(void);
     int get_month(void);
     char * get_name(void);
     int get_saving(void);
     int read(void);
protected:
private:
     int acc_num;
     int amount;
     char bk_name[10];
     int day;
     char event[20];
     int flug;
     FILE * fp;
     int month;
     char name[20];
     char operate;
     int saving;
// stp/omt class members end
};
// stp/omt class definition end
// stp/omt footer
#endif
// stp/omt footer end
```

図 4.6: StP/OMT で生成したインタフェースファイルの例 (balance_happened.h)

```
#include "balance_happened.h"
// stp/omt operation 5004::5222
int
balance_happened::get_saving()
{
        // stp/omt code
        // stp/omt code end
}
// stp/omt operation end
// stp/omt operation 5004::5224
char *
balance_happened::get_name()
{
        // stp/omt code
        // stp/omt code end
}
// stp/omt operation end
```

図 4.7: StP/OMT で生成した実装ファイルの例 (balance_happened.c)

notifier を利用して統合した StP/OMT では、数十以上のファイルを利用しており[6]、その全てのファイルの変更を notifier で監視している。そのため、常時監視しているとプロセスが重くなるので 10 秒に 1 回監視するようなプログラミングを行なっている[7]。StP/OMT での変更が PCTE リポジトリに反映されないといったことはないが、ある時点においては StP/OMT のデータの状態と PCTE リポジトリの状態に差異が存在している。notifier の監視間隔を短くすれば、差異の存在する時間も少なくなるが、その分プロセスが重くなるというジレンマを抱えており、この部分が notifier を使う上での最大の問題点である。

2. Object Center での変更が StP/OMT に波及しない

StP/OMT は ObjectCenter と閉じられた世界で統合されているが、StP/OMT のクラスから Object Center のプログラミング環境をナビゲートしたり、プログラミング環境から StP/OMT のクラスをナビゲートする程度の統合しかされていない。今回の統合でも、StP/OMT で生成したソースコードに Object Center で変更を加えても、その結果が StP/OMT のモデル図やクラステーブルエディタに反映されない[8]。その理由として、

[6] 今回作成した「家計簿シミュレーションシステム」では、およそ 30 ものファイルを利用する。

[7] システムコール sleep() を用いている。

[8] ここで言う変更とは、クラスや属性・操作の追加や削除などのモデル図に関係する部分である。

- Object Center で操作するファイルは「ソースファイル(.h,.cc)」であるのに対し、StP/OMT で操作するファイルは「オブジェクトモデル図ファイル」「クラステーブルファイル」等の複数のファイルである。
- 「ソースファイル」から StP/OMT で操作するファイルへの変換が現状ではできない[9]。

ということが挙げられる。StP/OMT では、クラスキャプチャというユーティリティが存在し、C++のソースコードからクラステーブルやある程度のオブジェクトモデル図を生成することができる。しかしながら、ソースコードに与えられる情報だけでは完全なモデル図を作成することはできず、結局上流へ戻って作業を行なう必要がある。

3. **バージョン管理導入による起動レスポンスの低下**

PCTE のバージョン管理のシステムについては、2.6 章でその概要を述べたが、基本的に既存のオブジェクトのコピーを行なうことである。今回統合したツールで新たなバージョンを作成する場合、カプセル化ツールの中で PCTE の version_revise 機能を利用して行なっている。version_revise を実行すると PCTE のシステムはオブジェクトのコピーを作る作業を行なう。StP/OMT では数十以上のオブジェクトがあるため、この操作は非常に時間がかかるものになり、CASE ツール起動までのレスポンスが大幅に低下する[10]。この問題は、PCTE リポジトリ上のデータの粒度の問題[11][6] とも絡んでおり、データの粒度が細かくなるにつれオブジェクトの数も増大するので、大きな問題となる。根本的な解決のためには、PCTE のオブジェクト管理システムの見直しが必要である。

4.3 ソフトウェア開発の後戻りによる問題点

ソフトウェア開発は、要求分析→仕様作成→設計→実現→検査→運用→保守の順に行なわれており、滝の水が流れ落ちるように、遂次、前段階で得られた成果を基に作業を行なう「ウォータフォール」の考え方があった。しかしながら、現在では、下流での作業中に上流での考慮もれや誤りを発見し、上流に戻って設計の修正を行なうという「行き戻り」

[9] この機能はオプションであり、標準では提供されていない。

[10] オブジェクトの改訂を行なうためには、そのオブジェクトが安定化(stable) されている必要があり、その操作も原因のひとつである。

[11] Emeraude PCTE の contents は、UNIX のディレクトリとファイルを利用して実現している。データの粒度が細かくなるとオブジェクトへのアクセスオーバーヘッドが増大し、その効率は大きく低下する。

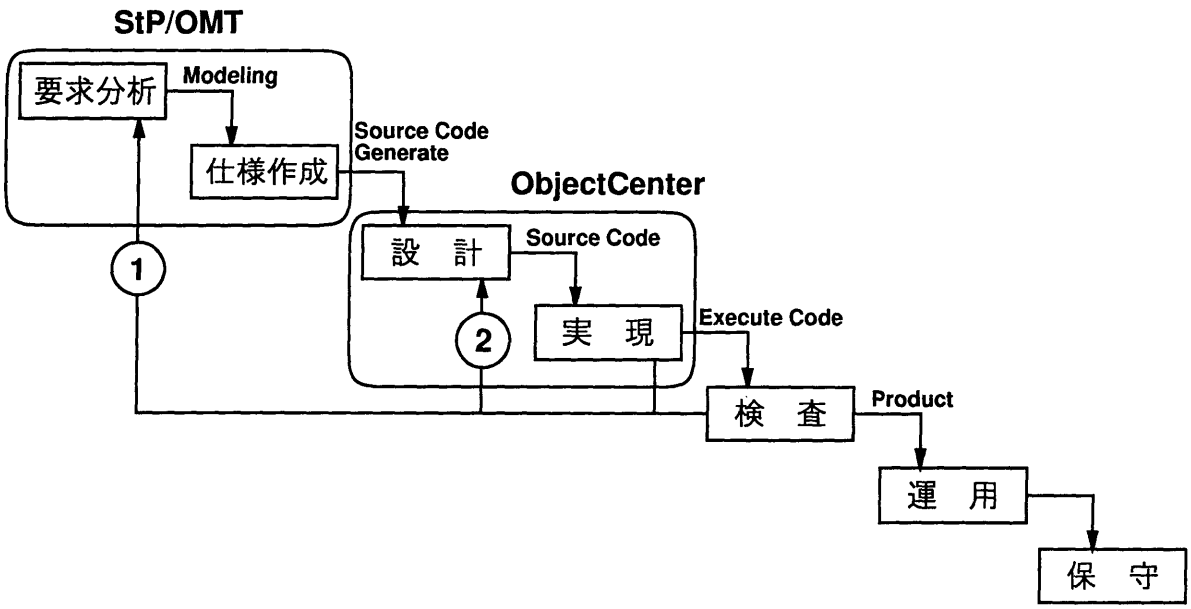

図 4.8: ソフトウェア開発の流れ

があるという考え方が主流となっている。実際に今回のツールの評価でもこのような場面に何度も遭遇している。

ところで、今回用いた CASE ツールは、図 4.8のようになっており、上流への後戻りに 2 通りあることがわかる。これらの違いは次の通りである。

1. **要求分析まで戻る場合**
   変更による影響の範囲が、StP/OMT にも及ぶ場合である。具体的には、StP/OMT で生成されたソースコードの変更など、モデル図にまで影響を及ぼす場合である。

2. **設計まで戻る場合**
   変更による影響の範囲が、ObjectCenter の内部で済む場合である。具体的には、StP/OMT で生成されたソースコードに書き加えたものに対する変更の場合は、ObjectCenter の内部での変更で良い。

ツール統合で問題となるのは、1 の場合である。今回作成したツールではこのような場合の修正は、上流の要求分析の段階まで戻る必要がある。その理由は、先ほど述べている通り、ソースコードからオブジェクトモデルを生成することができないからである。

しかしながら、オブジェクトモデルを再生成したとしても問題が生じてくる。

1. **変更が他のツールまで影響する**
   オブジェクトモデルの変更が、StP/OMT 内の他のモデル図やクラステーブルだけでなく、これらのデータを参照している他のツールにも影響してくる (図 4.9 参照)。

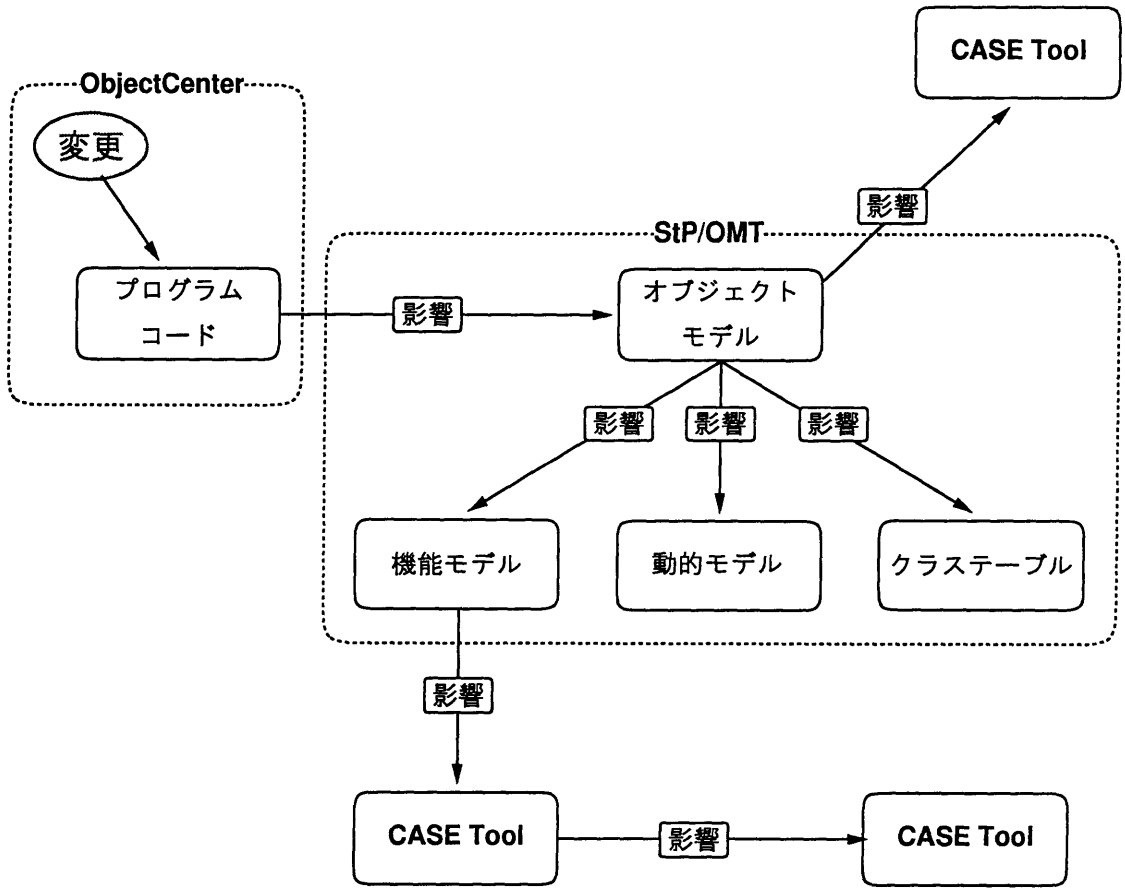

図 4.9: 変更の影響の伝搬

2. 変更の影響の範囲がわからない

   ひとつのツールの変更の影響が、どのツールまで影響するかということがわからない。すなわち、ツール生産物の依存関係は、人間か管理しており、影響を及ぼす範囲内のデータの更新を行なっている。そのため、複雑な構成になると影響を見落す可能性がある。

これらの問題を解決するために、ソフトウェア開発の修正を支援する機能が必要になってくる。解決策としては、次のような方法が考えられる。

1. 制御統合の導入

   ツール統合の必要条件のひとつに「制御統合」がある。制御統合とは、個々のツール間の協調を行なうためのものである。そこで、データの変更が行なわれると影響のあるツールが起動する、または自動的にデータの修正を行なうような機能が必要になる。PCTE では、制御統合のために PCTE notifier と、更に高度な統合を実現するためツール間のコミュニケーションを行なう TCI サーバーを利用した Emeraude

TCI が用意されている。この機能を利用して他のツールまで変更が影響する問題は解決することが可能である。

2. **スキーマによる依存関係の表現**
   PCTE のスキーマは ERA[12] モデルを表現したものであり、オブジェクト間の関係を容易に表現することができる。そこで、スキーマ上でツール間のオブジェクトの生成に関する依存関係を表現することで、データの変更による影響を及ぼす範囲を特定することができる。

3 章において、CASE ツールのデータ構造をそのまま PCTE のリポジトリに格納をするようなスキーマを用いた。しかしながら、この構造ではこれまでに説明したような問題が生じる。そこで、PCTE のスキーマを作成する指針 (3.3.1) に次のことを追加する。

- ツール間及びデータの依存関係を考慮してスキーマの作成を行なう

図 4.10に StP/OMT と ObjectCenter におけるツールとデータの依存関係を追加したスキーマを示す。この図におけるデータの中心は、オブジェクトモデル図のデータ ome_files[13] であり、ここからそれぞれのデータに依存関係を示すリンクが張られている。また、これらのオブジェクトは、バージョン管理を行なうためのリンクも持っている。すなわち、あるバージョンのオブジェクトモデル図に対応する動的モデル図・機能モデル図・クラステーブル図及びソースコードを特定することができ、さらにその中でもバージョンを管理することが可能である。また、データ間の関連がリンクで表されているので、あるデータを変更した際の影響の範囲が容易に特定することができる。

このような、スキーマを利用することで StP/OMT と ObjectCenter で生成されるデータを一貫して管理することが可能になる。

[12] Entity-Relationship-Attribute

[13] このオブジェクトは、構成管理をするオブジェクトであり、実際のデータは ome_objects になる。

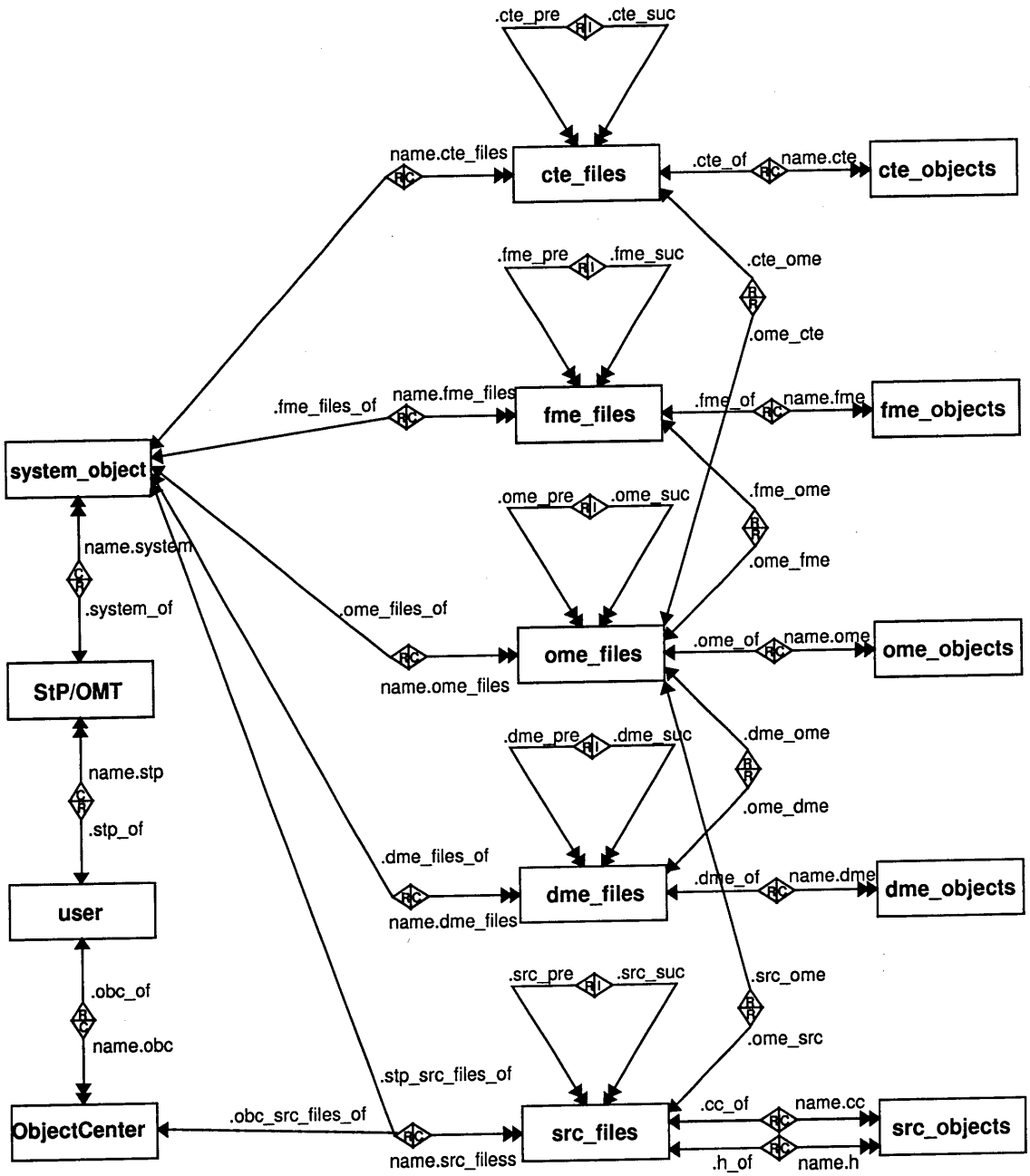

図 4.10: 依存関係を考慮した新しいスキーマ

第 5 章

おわりに

本研究では、ソフトウェア開発で使用されている既存の CASE ツールを、CASE 環境のフレームワークである PCTE を用いてデータ統合するための統合技術の調査・開発を行なった。その概要は以下の通りである。

- PCTE ネーティブな CASE ツールの作成
- CASE ツールの移植
- リホスティング
- CASE ツールのカプセル化
  - contents_get 関数の使用
  - CASE ツールのインタフェースプログラミング言語の使用
  - notifier の使用
  - シンボリックリンクの使用 (export/import mechanism)
- Emeraude TCI の使用

また、これらの技術を用いて、PCTE 上で動作する統合ツールの作成を行なった。統合ツールでは、PCTE を中心にした CASE ツールのデータ統合を行なっただけではなく、既存の CASE ツールには有しないバージョン管理の機能を、PCTE のリポジトリ上で実現することを可能にした。これらの機能により、上流工程から下流工程へのソフトウェア開発がスムーズに行なわれ、品質の高いソフトウェアの開発が期待できる。

しかしながら、PCTE による CASE ツール統合は、これで全てではなく、まだ残された課題がある。その 1 つが、CASE ツール間のコミュニケーションである「制御統合」である。今回作成した統合ツールにおいて、下流工程のツールの変更が上流工程のツールに波及しないという問題があったが、制御統合の機能[1]を用いることでかなりの部分の改善が期待できる。

[1]Emeraude PCTE では Emeraude TCI で提供されている。

また、PCTE のバージョン管理は、現在リンク関係を用いて実現されている。この場合、異なるバージョンのオブジェクトが全てユーザから見えることになり、情報隠蔽の面からあまり好ましいものではない。PCTE では、スキーマを利用して情報のフィルタリングを実現しているが、これをバージョン管理にも導入することで、構成管理の機能を使用せずにバージョン管理を行なうことができるようになる。現在の PCTE の作業スキーマは、ツールの実行中に作業スキーマを変更するといった機能をサポートしていないが、このような機能が実現することでより強力なバージョン管理を実現することができるであろう。

謝辞

本研究を進めるにあたり，多方面から御助言を賜わりました篠田陽一助教授には深く感謝致します。

本研究を進めるにあたり、適切なコメント、御助言を賜わりました片山卓也教授には深く感謝致します。

本研究を進めるにあたり、有益な御助言を頂いた海谷治彦助手には深く感謝致します。

本研究を進めるにあたり、有益な御助言を頂いた藤枝和宏氏には深く感謝致します。

参考文献

[1] Lois Wakeman Jonathan Jowett 著 松本吉弘 監訳, PCTE:開放型リポジトリのための標準, 日科技連出版社, 1994.

[2] 鯵坂恒夫, 開放型 CASE プラットフォーム, コンピュータソフトウェア,Vol.10,No.2, Mar. 1993.pp.4-12.

[3] James Rumbaugh,et al. 羽生田栄一 監訳, オブジェクト指向方法論 OMT, トッパン,1992.

[4] European Computer Manufactures Association, PCTE C Programming Language Binding, Standard ECMA-162,June 1991.

[5] 落水浩一郎, ソフトウェア工学実践の基礎, 日科技連,1993,pp101-190.

[6] 鯵坂恒夫, 沢田篤史, 満田成紀,Emeraude PCTE, コンピュータソフトウェア, Vol.10,No.2,Mar. 1993.pp.65-77.

[7] GIE Emeraude, Emeraude PCTE Tool Catalogue 1994.

[8] GIE Emeraude, Emeraude PCTE Version Management Common Service C Language, 1994.

[9] GIE Emeraude, Emeraude TCI,User Guide 1994.

[10] GIE Emeraude, Emeraude TCI,Reference Manual 1994.

[11] Interactive Development Environmente Inc., StP/OMT Creating OMT Models, 1995.

[12] CenterLine Software Inc., アステック (訳), Object Center リファレンスマニュアル Oct. 1993.

[13] CenterLine Software Inc., CLIPC Programmer's Reference Manual Nov. 1992.

[14] D.Schefstrom,G.van den Broek, TOOL INTEGRATION, John Wiley & Sons Ltd,1993.

[15] Interactive Development Environments, StP Core Object Management System, Release1,Feb 1994.

[16] Interactive Development Environments, StP Core Query and Reporting System, Release1,Feb 1994.

付録 A

「家計簿シミュレーションシステム」の問題

こういちろう一家は妻えつこ、長男ひろき、次男ともきの 4 人家族です。静岡県で仲良く平凡な暮らしを送ってきました、平成 4 年の 4 月に生活にちょっとした変化がありました。こういちろう氏の勤務先が静岡から金沢に転勤になり、都合により単身赴任することになったのです。こういちろう氏は転勤後しばらくは新しい勤務地での仕事に追われ、また休日にはまわりの素晴らしい大自然を満喫していましたが、ある日突然考え込むようになりました。どう考えても、生活費が足りないのです。こういちろう一家には申し訳ありませんが、少し一家状況を説明いたしましょう。

家計の収入は給料とボーナス、それからこういちろう氏の臨時収入です。給料は毎月 17 日に、兼六銀行の口座に 40 万円振り込まれます。ボーナスは 7 月 10 日と 12 月 10 日に、兼六銀行の口座にそれぞれ 60 万円と 80 万円振り込まれます。兼六銀行の口座では、こういちろう氏とえつこさんがキャッシュカードを使ってお金を現金化することができます。こういちろう氏の臨時収入は不定期で三葉銀行に年 6 回ほど、1 回当り 5 万円程度の金額が振り込まれます。三葉銀行の口座のキャッシュカードは、こういちろう氏しか持っていません。えつこさんとの契約で、主にこういちろう氏の交際費 (東京出張時の飲み代) に使われていますが、父親の権威を保持するためにも若干利用されているようです。

こういちろう氏が悩んだのは、4 人一緒に生活していた時は一家の大黒柱のえつこさんが全体の財布を握っており、収入と支出のバランスや収入と支出のタイミングがあわないときのやりくりを、えつこさんの知恵でうまく調整していたのですが、それがうまくいかなくなってきて、家族全体の生活費が大幅に不足する状況が 8 月ごろに顕著になってきたからなのです。

家族は、いま 3 箇所に別れて別々に生活しています。長男のひろき君は大学生で、横浜に下宿しています。家からの仕送りは月 12 万円で、このまえ帰省したときの彼の報告によると、だいたい次のような配分で毎月生活しているようです。家賃に 2 万 5 千円、光熱水道費に 5 千円、食費に 3 万円、電話代に 3 千円、これもつきあいがよすぎて飲み代に

1万円、写真クラブと管弦楽の部活関係に2万円、読書家で本題に1万円、あと出途不明金が1万7千円程度だそうです。最近ガールフレンドができたので、多分デート代でしょう。ときどきNTTから電話代金が振り込まれていないとの連絡があるので、多分全体的には足らずにアルバイトをしているのかもしれませんが、そのへんは干渉しないことにします。送金は、郵便貯金口座を使っておこなわれます。つまり、えつこさんが毎月17日に兼六銀行から引き出すお金のうち、12万円を隣の郵便局までいって振り込むという段取りです。郵便貯金のキャッシュカードは、えつこさんとひろき君が持っています。

こういちろう氏の金沢での生活は、月10万円の約束になっています。兼六銀行から必要な額だけキャッシュカードを使って、そのつど引き出します。こういちろう氏の悩みの直接の動機は、ここ数カ月、17日のお昼ごろお金を引き出しにいくと残高が5万円から7万円しか残っていないことが続いたからです。

こういちろう氏の毎月の支出は、以下のような状況です。食費が3万円、電話代が1万円(晩ご飯のおかずの作り方をいちいち電話で聞くので、こんなに費用がかさんでいます)、光熱水道費に5千円、飲み代に1万5千円、たばこ代に1万円、生活用品の購入に1万円、ガソリン代に1万5千円、あと日曜ごとに近くの温泉に出掛けるのでその費用が5千円です。すこし無駄遣いが多いので、いつも注意されています。最近は飲み代と食費、電話代を切り詰めていますが、単身赴任なのであまり切り詰めると精神衛生上よくありません。

静岡でのえつこさんとともき君の生活費は、2人で18万円です。毎月17日に兼六銀行からえつこさんが30万円引き出し、浜名湖銀行に振り込んで、必要に応じて引き出しながら使います。内訳は食費が5万円、電話代が5千円、光熱水道費に6千円、あと生活用品の購入に月2万円ほどが共通的な出費です。ともき君の小遣いは5千円です。ともき君は高校3年生で、来年は受験です。受験勉強のための費用が、月に3万円ほど必要です。えつこさんの小遣いは、月2万円程度です。残りの4万4千円は、次に説明する「家計を圧迫する支出」のために浜名湖銀行に積み立てられます。

以上が定期的な支出ですが、これ以外に家計を圧迫する支出がたくさんあります。

まず年2回、4月と10月に、ひろき君の授業料を1回につき20万円大学に送金しなければなりません。9月には、車検代と自動車税を合計20万円くらい支払う必要があります。自動車税の任意保険料の支払いが5月に4万円必要です。盆と正月には、家族そろって田舎に帰省するので、1回当り約15万円の費用がかかります。春と秋には近くの観光地に家族旅行に出掛けますので、それぞれ10万円くらい費用がかかります。ひろき君のクラブの合宿やともき君のスキー旅行、こわれた電化製品の購入、田舎から出てきた親戚の接待など、出費時期は不定期ですが、年間を通じると30万円程度のお金は必ず出ていきます。

ここまでの収支のバランスを計算してみますと、まだ余裕があるみたいですが、とてもそういうわけにはいきません。貯金などぜんぜんできないのです。というのは、病気や外食、春夏の洋服の購入など、使用目的は特定されないのですが、余裕があれば使い、余裕がなければがまんするというたぐいの出費があります。

付録 B

Emeraude PCTE 管理&操作マニュアル

B.1 PCTE の起動

B.1.1 server の起動

PCTE を動かすためには、server を起動する必要があります[1]。落水研究室の server は現在[2]oserver になっています。server を起動するには以下の手順で行なって下さい。

1. PCTE 管理者[3]として oserver に login します。
2. host_stat コマンドで server の状態を調べます。STOPPED の場合 3 へ、NOT CONNECTED の場合 4 へ、CONNECTED の場合 5 へそれぞれ進みます。
3. host_start コマンドで server をスタートさせます。次のような表示があります。

```
$ host_start
Copyright (C) 1995 GIE Emeraude, Emeraude V12.6.1  - Sun4 SunOs 4.1.3 -

 ===  Starting pcte_actl Emeraude V12.6.1

 ***  HOST STARTED
 *** CHECKING ADMIN VOLUME 0
** Phase 1 ==> Checking header area                          O.K.
** Phase 2 ==> Checking free packet list                     O.K.
** Phase 3 ==> Checking free object list                     O.K.
** Phase 4 ==> Checking objects, links and attributes        O.K.
** Phase 5 ==> Checking lost blocks and lost objects         O.K.
```

[1]rc.local に起動のためのスクリプトを記述してある場合、必要無い場合があります。

[2]平成 8 年 1 月 28 日現在

[3]現在は login name:pcte になっています。

```
** Phase 6 ==> Checking relationship consistency                O.K.
                ------ No error ----
 ***  ADMIN VOLUME 0 MOUNTED
 ***  COMMON ROOT INITIALISED
 ***  SYSTEM SCHEMA ESTABLISHED
$
```

4. host_connect コマンドで server と接続します。しばらくするとプロンプトに戻ります。host_stat コマンドで調べると CONNECTED になっているはずです。

5. server は無事起動しました。

B.1.2 client の起動

PCTE は 1 台の sever と 63 台までの client[4] で運用することができます。以下のように client の起動を行なって下さい。

1. PCTE 管理者として client のマシンに login します[5]。

2. host_stat コマンドで client の状態を調べます。STOPPED の場合 3 へ、CONNECTED の場合 4 へそれぞれ進みます。

3. host_start コマンドで client をスタートさせます。次のような表示があります[6]。

```
$ host_start
Copyright (C) 1995 GIE Emeraude, Emeraude V12.6.1  - Sun4 SunOs 4.1.3 -
 ***  HOST STARTED
 *** CORRESPONDING ADMIN VOLUME IS 0
$
```

4. client は無事起動しました。

---

[4] client を使用する場合は、host_create コマンドで新たに client を結合する必要があります。client を結合方法には新たに管理ボリュームを作成する方法と、既存の管理ボリュームを使用する方法 (ディスクレスステーションと呼んでいる) の 2 通りあります。詳しくは、Emeraude PCTE Release Bulletion(日本語版) を参照してください。

[5] client のマシンにつけた login 名とパスワードがあります。

[6] client の状態 (論理ボリュームがある、管理ボリュームがある等) で若干表示が異なります。

B.2 PCTE の終了

マシンの電源を切る等のシャットダウンの操作を行なう前に、PCTE のシャットダウンを行なう必要があります。

1. PCTE 管理者として server のマシンに login します。
2. host_disconnect コマンドで server との結合を切ります。server から次のようなシャットダウンメッセージかきます。

```
$ host_disconnect
Broadcast Message at 21:50 ...
CORRESPONDING HOST BEING DISCONNECTED, LOCAL HOST IS STOPPED IMMEDIATELY
```

3. その後、host_stop コマンドで server を停止させます。host_start コマンドで確認すると STOPPED になっているはずです。この操作で、すべての client も PCTE の動作が停止します。

B.3 PCTE での C プログラム実装について

PCTE 上で動く C プログラムを作成するには、PCTE のリポジトリ上に作成する必要があります。作成手順を以下に示します。

B.3.1 ワーキングスキーマの変更

PCTE 上で C プログラミングを行なうために、C プログラミング用のスキーマ 'c_prog' を加える必要があります[7]。以下の操作を行なって下さい。

1. esh からは ws_set コマンドを用いてワーキングスキーマをセットします。

```
esh$ ws_set c_prog pact env
```

2. oms_browser からは、Misc の中の Chenge Working Schema を選ぶと図 1 のようなダイアログが現れます。ここに、必要なスキーマを書きます[8]。

---

[7] 最低限 pact スキーマがなければなりません。

[8] スペースで区切ることによって複数のスキーマをワーキングスキーマにすることができます。

B.3.2 オブジェクトの作成

C ソースファイルを格納するオブジェクトを作成します。C ソースファイルを格納するオブジェクトは、c_prog スキーマで定義されている場所に作成しなければならないので、スキーマを見ながら作成します。

オブジェクト program の作成

1. esh では obj_create コマンドを使用して作成します。作成する際は、必ず自分のホーム (loginname.usr) の下で行ないます。

```
esh$ co ~
esh$ obj\_create c\_prog-program programname.prog
```

2. oms_browser からは、Tools の中の Create object を選ぶと図 2 のようなダイアログが現れます。ここには、現在の場所からリンクを作ることのできるオブジェクトが示されています[9]。この中の c_prog-program の c_prog-prog をマウスでクリックし、Keys の部分に programname を入力します。

オブジェクト subset の作成

1. esh ではオブジェクト program の作成と同様に obj_create コマンドを使用して作成します。作成する際は、先ほど作成した programname.prog の下で行ないます。

```
esh$ co programname.prog
esh$ obj_create subset subsetname.sub
```

2. oms_browser からは、Links の中にある先ほど作成した programname.prog をマウスでクリックし、GO TO をクリックしてチェンジオブジェクトします[10]。その後、Tools の中の Create object を選ぶと図 2 のようなダイアログが表示されますので、先ほどと同様に、この中の c_prog-subset の c_prog-sub をマウスでクリックし、Keys の部分に subsetname を入力します。

[9] ワーキングスキーマによって表示内容が異なります。

[10] 作成した直後は、作成したオブジェクトは表示されていません。一旦、GO TO ボタンをクリックすると、表示が更新されます。

オブジェクト c_source の作成

この部分に、C のソースコードを格納します。

1. esh ではオブジェクト subset の作成と同様に obj_create コマンドを使用して作成します。作成する際は、先ほど作成した subsetname.sub の下で行ないます。

```
esh$ co subsetname.sub
esh$ obj_create c_prog-c_source c_source-name.c
```

2. oms_browser からは、Links の中にある先ほど作成した subsetname.sub をマウスでクリックし、GO TO をクリックしてチェンジオブジェクトします。その後、Tools の中の Create object を選ぶと図 2 のようなダイアログが表示されますので、先ほどと同様に、この中の c_prog-c_source の c_prog-c をマウスでクリックし、Keys の部分に c_source-name を入力します。

B.3.3 オブジェクトの編集

オブジェクトの編集には、エディタ evi[11]、enemacs[12]、もしくは、oms_browser の Edit の中の obj_edit を用いて C プログラムを作成します。

B.3.4 オブジェクトのコンパイル

オブジェクトのコンパイルには、コンパイラ ecc を用います。コンパイルされた実行オブジェクトは、オブジェクト program の下に name.out という名前で生成されます[13][14]。

```
esh$ ecc -o ../name.out c_source-name.c
```

なお、リンクしなければならないファイルが多数ある場合は、subset の下に makefile.mk[15] を作成すると良いでしょう[16]。

---

[11] これは、vi をカプセル化して作成しているらしいので、キーバインドその他は vi と同様です。

[12] 私が nemacs をカプセル化したもの。contents_get で unix path を得て、プログラム内部から nemacs を exec している。

[13] 生成場所は、-o オプションを付けて明示しなければなりません。

[14] もしくは、オブジェクト program の下にオブジェクト.build を作成し、その下に name.tool という名前で生成することができる。

[15] make は emake で行なう。

[16] PCTE の機能を使う場合は、UNIX の include ファイルの他に、PCTE の include ファイルも使わなければならないので使用した方が楽です。Emeraude PCTE のソースディレクトリの下にある example の中の makefile を参照すると良いでしょう。

B.3.5 オブジェクトの実行

コンパイルによって生成された実行オブジェクトは、オブジェクト program の下に name.out という名前であります。これは、esh 上で実行することができます。

```
esh$ co ~/programname.prog/
esh$ name.out
```

B.4 スキーマの定義について

PCTE は、使用するツールのデータ構造をスキーマで表現する必要がある。スキーマを作成する方法を以下に示す。

B.4.1 sds_design 起動までの準備

スキーマの定義には sds_design という、グラフィックツールを用いて描写する。このツールを起動するには、ワーキングスキーマに pact を加える必要がある。

```
esh$ ws_add_sds pact
```

次に、定義するスキーマを保存するオブジェクトを作成する。このオブジェクトはホームオブジェクトの下に作成する。

```
esh$ obj_create pact-dir ~/sdsdir.e
```

B.4.2 スキーマの定義

新たにスキーマを定義する場合

新たにスキーマ mysds を定義する[17]場合は、以下の操作を行なう。

```
esh$ sds_design ~/sdsdir.e/mysds.sds_draft
```

この場合の sdsdir.e は「4.1 sds_design 起動までの準備」で作成したオブジェクトを指定する。また、mysds の拡張子は、必ず.sds_draft とする。

既存のスキーマを拡張する場合

既存のスキーマ mysds をユーザー名 taro が拡張する場合は、以下の操作を行なう。

```
esh$ sds_design -e mysds _/.users/taro.usr/sdsdir.e/mysds.sds_draft
```

[17]当然、既存のスキーマと同じ名前は付けられない。

この場合の sdsdir.e は「4.1 sds_design 起動までの準備」で作成したオブジェクトを指定する。また、mysds の拡張子は、必ず.sds_draft とする。さらに、mysds.sds_draft は root からの絶対パスで指定しなければならない[18]。

B.4.3 スキーマの描写

4.1.1 または4.1.2 の操作を行なうと、スキーマ描写のツールが起動する。ツールの右側にあるアイコンを図 3 に示す。詳しい使い方は Tool Catalogue Volume 1 の sds_design を参照のこと。

B.4.4 スキーマのコンパイル

作成したスキーマ図は、PCTE で使用するために、コンパイルをする必要がある[19]。なお、コンパイルの際は、必ず host0[20]で行なう必要がある。また、この操作も、mysds.sds_draft の指定は root からの絶対パスで指定しなければならない。

```
esh$ sds_design -v _/.users/taro.usr/sdsdir.e/mysds.sds_draft
```

この操作を行なうと、sdsdir の下に、mysds.lsds というオブジェクト名でスキーマが生成される。このスキーマをワーキングスキーマに加えることで、新たなスキーマを使用することができる。

B.4.5 スキーマ図のポストスクリプト変換

作成したスキーマ図は、プリンターで印刷するためにポストスクリプトに変換することができる。

```
esh$ sds_design -ps _/sdsdir/mysds.lsds
```

このままでは、ターミナルに流れるので、リダイレクトを使ってファイルに落します。

[18]新たに作成する場合は、相対パス指定でも生成されるが、既存のスキーマを拡張する場合は、何故か絶対パスで指定しないと生成されない。

[19]ここでは、sds_design を用いているが、sds_design の内部から sds_compile を呼んでいる。

[20]現在の host0 は oserver

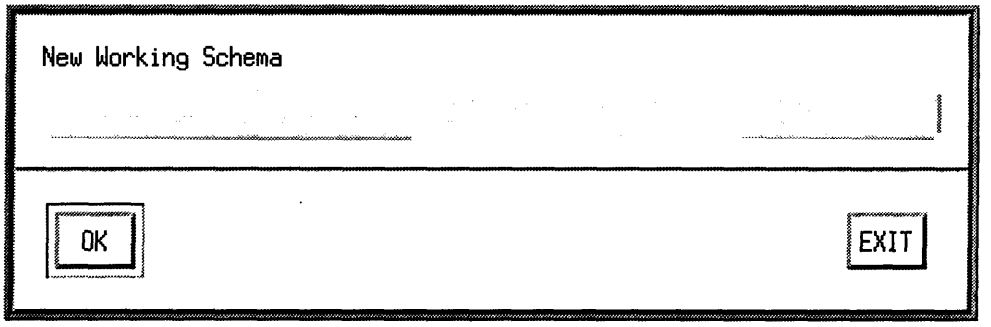

図 B.1: Chenge Working Schema ダイアログ

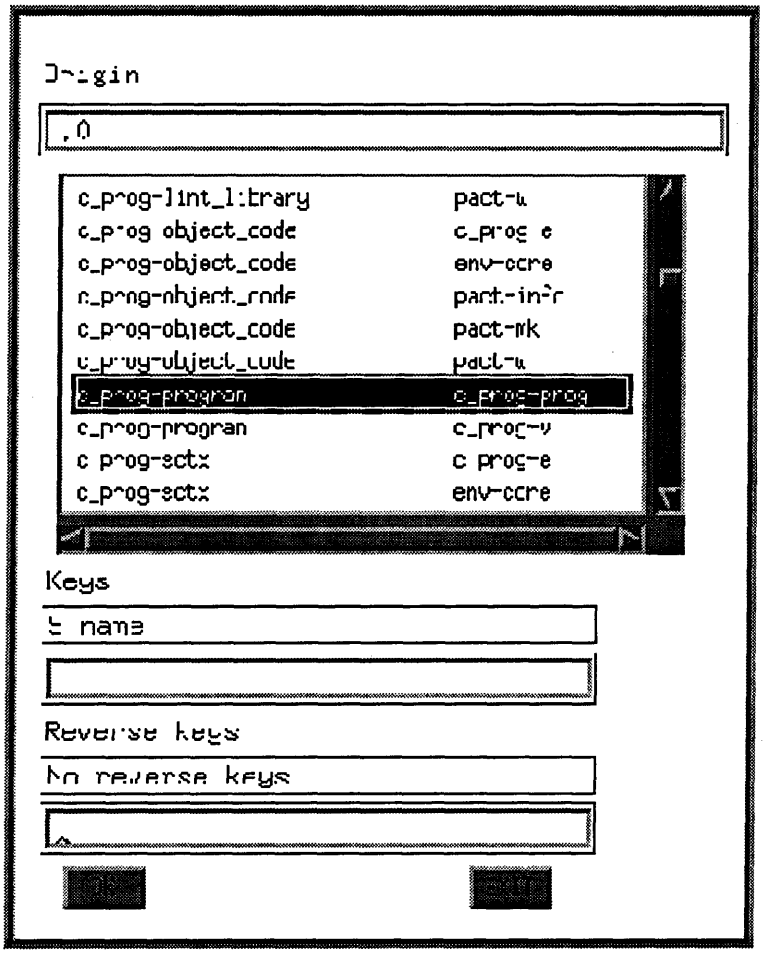

図 B.2: Create object ダイアログ

Crane：他の sds からオブジェクトを import する際に使用。drawing area でクリックして位置を決めたのち、アイコンをクリックする。オブジェクトの名前を入力する。

Rectangle：オブジェクトの生成に使用。drawing area でクリックして位置を決めたのち、アイコンをクリックする。オブジェクトの名前を入力する。

Ouals：属性の設定に使用。アイコンをクリックして、識別子を入力する。

Triangle：リンクの生成に使用。4 方向に三角形のリンクを書くことができる。リンク型のカテゴリー I(暗示リンク)・R(参照リンク)・C(構成リンク) をクリックし、属性をクリックした後 (複数の場合は、シフトを押しながら)、オブジェクトの方向に合わせて適切な三角形のアイコンをクリックする。リンク名を入力する。逆リンクは

Arrow：リンクとオブジェクトを結合する。つなぎたいオブジェクトを順にクリックしたのち (2 つ目はシフトを押しながらクリック)、アイコンをクリックする。矢印は、先にクリックしたオブジェクトから後にクリックしたオブジェクトの方向に作られる。

Scissors：削除する。削除したいオブジェクトをクリックして反転させた後、アイコンをクリックする。

図 B.3: sds_design で用いられるアイコン

付録 C

Emeraude PCTE によるツール統合技術

C.1 はじめに

PCTE とは、可搬型共通ツール環境 (Portable Common Tool Environment) の略で、開放型リポジトリのための共通インタフェース標準であり、CASE ツールが高度な統合性と可搬性を可能にするための共通で標準的なサービス群を定義している。PCTE に関して理論的なことは、文献 [1] に記載されているが、どのような方法で統合するかといった技術的な内容についてはほとんど触れられていない。そこで本稿では、どのように CASE ツールを統合していくかといった技術的な内容を中心に進めていく。

C.2 CASE ツール統合の考え方

CASE ツールを統合する方法には以下の 3 つの方法がある [2]。

1. 表示・操作統合
2. データ統合
3. 制御統合

Emeraude PCTE[1]では、1 の表示・操作は UNIX の X-Window システムを用いることによってその統合を可能にしている。しかしながら、2 のデータ統合や 3 の制御統合は新たな技術の導入が必要となってくる。

そこで PCTE では、「カプセル化」という技術を用いることによって既存の CASE ツールを容易に統合することができるようになる。図 C.1に PCTE の概略を示した「トースタモデル」[2]を、また図 C.2にカプセル化の概念を示す。

[1] この資料中で Emeraude PCTE に関する部分は、Emeraude PCTE V12.6.1 に基づいています。

[2] ECMA と NIST が提唱した CASE 環境の枠組。

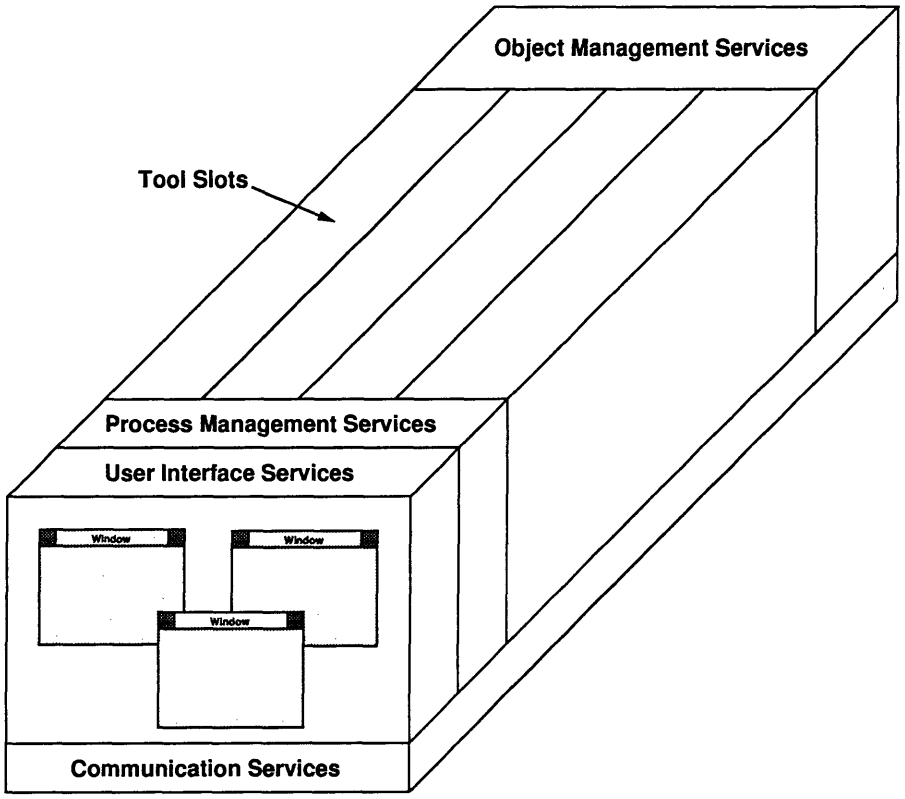

図 C.1: トースタモデル

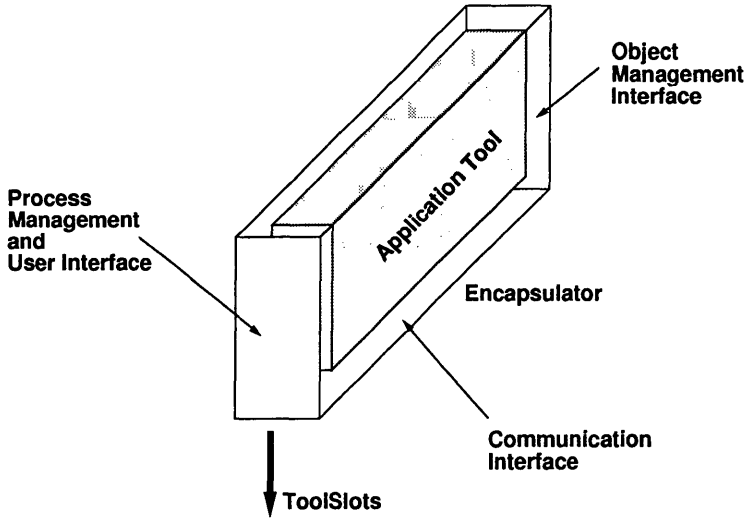

図 C.2: カプセル化の概念

図 C.1のトースタモデルは、PCTE の環境を表しており Tool Slot のツールが表示や操作・データの共有やツール間の制御統合が可能であることを示している。ところで、この Tool Slot で使用できるツールはこのままでは、PCTE に準じたツール、すなわち PCTE ネーティブなものでなければ使用することはできない。しかしながら、図 2 に示すようなカプセル化の技術を用いることで PCTE に準じていないツールでも使用することができるようになる。すなわち、統合するアプリケーションツールのインタフェースと PCTE インタフェースの間で「対話」をできるようにする「カプセル」ツールを作成することで統合することが可能になる。

以降の章では、カプセル化の方法を含め、CASE ツールのデータを統合するさまざまな方法を紹介する。

C.3 PCTE によるデータ統合の方法

PCTE によるデータ統合の方法には以下の方法があると考えられる。

1. PCTE ネーティブな CASE ツールの作成
2. CASE ツールの移植
3. リホスティング
4. CASE ツールのカプセル化
   (a) contents_get 関数の使用
   (b) CASE ツールインタフェースプログラミング言語の使用
   (c) notifier の使用
   (d) シンボリックリンクの使用 (import/export メカニズム)
5. Emeraude TCI の使用

これらの方法が、PCTE 上どのように使用されるか詳しく説明していく。

C.3.1 PCTE ネーティブな CASE ツールの作成

図 C.1のトースタモデルの Tool Slot に直接差し込んで使用することができるツールを作成する方法である。この方法は、PCTE のインタフェースに直接アクセスするツールを作成するので、ツールを実行する上で如何なる制限も受けない。しかしながら、新たにツールを作成するのと同様の労力が必要であり、あまり現実的な方法ではない。

表 C.1: contents 操作関数とそれに対応する file 操作関数

contents 操作関数	file 操作関数
contents_close	fclose
contents_copy_from_foreign_system	-
contents_copy_to_foreign_system	-
contents_get_position	ftell
contents_open	fopen
contents_read	fread
contents_seek	fseek
contents_set_position	fseek
contents_write	fwrite

PCTE 上にネーティブな CASE ツールを作るために、PCTE を操作するためのバインディング言語[3]が用意されている [1]。これは、PCTE の抽象操作を具体的な言語に対応させ、プログラムとして実装することができることを示している。Emeraude PCTE ではバインディング言語として C 言語が用意されており、これを用いて PCTE のインタフェースを操作する CASE ツールを作成する。

Emeraude PCTE の PCTE 操作関数の資料は、PCTE をインストールするために展開したデリバリディレクトリ[4]の下の man3 ディレクトリ以下に存在する。PCTE が定義している操作を Emeraude PCTE が全て提供しているわけではないので注意が必要である。また、これらの関数を使用する際には、インクルードファイルの指定及びライブラリのパスの指定をする必要がある。

C.3.2 CASE ツールの移植

CASE ツールの移植とは、既存の CASE ツールのファイル入出力に関する部分の関数を PCTE のリポジトリ入出力の関数に置き換えて再コンパイルを行ない、PCTE で使用できるようにする方法である。そのため、移植する CASE ツールのソースコードが必要である。

表 C.1に PCTE のリポジトリ入出力関数である contents 操作関数とそれに対応する file 操作関数を示す。file system の file を操作するのと同様の操作を PCTE のリポジトリに対

[3]Ada・C 言語 [4]・C++の 3 言語が定義されている。

[4]現在のところ、/soft/src/pcte-v12.6.1 である。

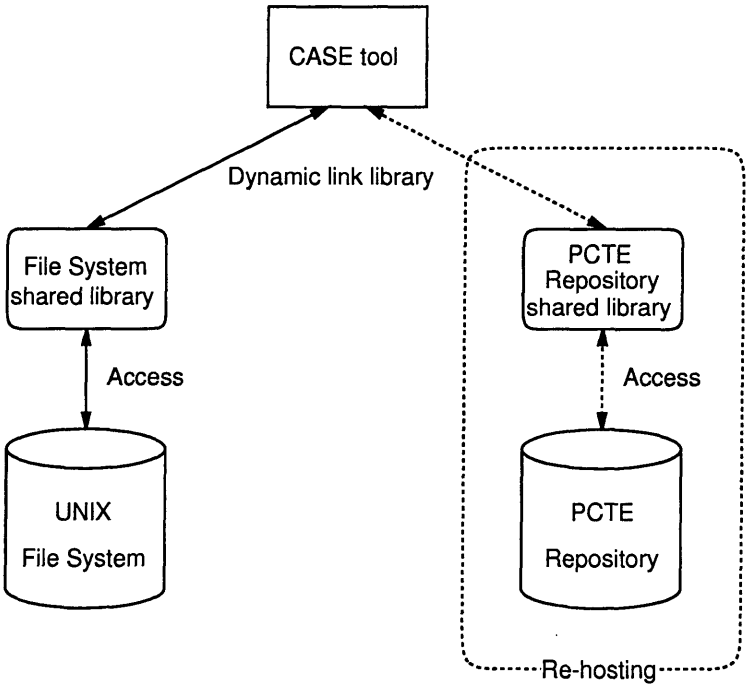

図 C.3: リホスティングの概念

して行なうことが可能である。変更に必要な作業は、元のソースコード中の file 操作関数を、表 C.1に示す contents 操作関数に置き換えて再度コンパイルを行うことだけである。

なお、これらの関数の資料は、デリバリディレクトリの下の man3 以下に存在する。

C.3.3 リホスティング

通常 UNIX 上で動作するツールは、実行ファイルに必要な資源を全て持っているのではなく、必要なライブラリを実行時にリンクしている。そこで、実行時にファイル入出力に関する shared library を PCTE の contents 入出力に関する shared library に置き換えて実行することで、PCTE のリポジトリにアクセスすることができるようになる (図 C.3参照)。この方法は、目的のツールの実行ファイルのみ存在すればデータ統合が可能である。しかしながら、Dynamic にライブラリとリンクされているツールのみで使用でき、Static にリンクされているツールでは使用することができない。この場合は、目的のツールのオブジェクトコードをリンクする際に PCTE リポジトリへアクセスするライブラリを使用して、再度リンクを行なうことで使用することが可能になる。しかしながら、この方法は目的のツールのオブジェクトファイルが必要となる。

これらの統合方法は、デリバリディレクトリの下の TCI/lib/rehosting.readme に詳しく書かれている。

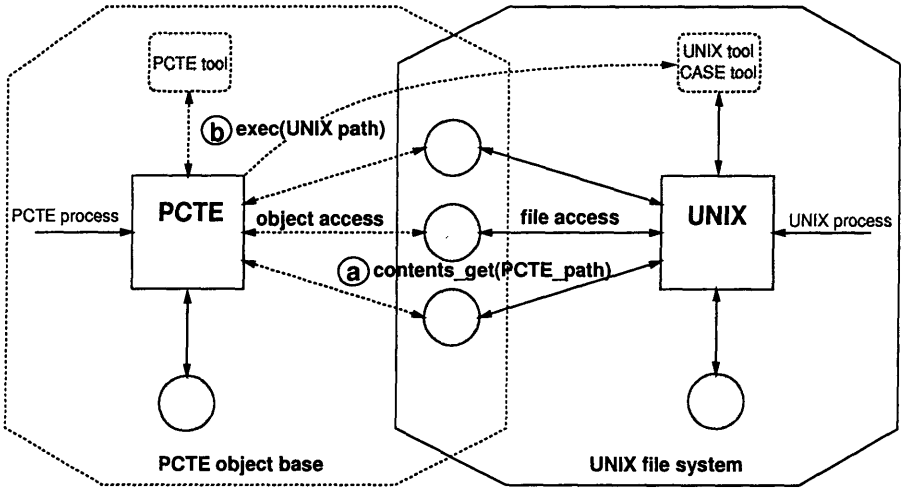

図 C.4: contents_get 関数を用いたカプセル化ツールの概念

C.3.4 CASE ツールのカプセル化

既存の CASE ツールのデータインタフェースと、PCTE のデータインタフェースを合わせ、PCTE に準じていない CASE ツールを PCTE 上で使用する方法をカプセル化と言う。この方法は、元のツールの実行ファイルのみ存在すれば統合が可能である。

カプセル化ツールでは、基本的にデータインタフェースを合わせた後に UNIX のシステムコール system() を用いて、カプセル化ツールの内部から CASE ツールを呼び出す形になる。

ここでは、カプセル化ツールの具体的な例を含めながら、カプセル化技術を紹介していく。

contents_get 関数の使用

contents_get 関数は、PCTE のパス名を UNIX のパス名に変換する関数である。この関数を利用してエディタ vi のカプセル化ツールを作成することができる。カプセル化ツールの操作の主な流れは図 C.4に示すように、

1. 作業を行なうファイル名を PCTE のパス名から UNIX のパス名に変換する。

2. システムコール system() を用いて UNIX のパス名を引数にして実行する。

という流れになる。図 C.5に vi のカプセル化ツール evi の例を示す。このツールは vi と同様に作業を行なうファイル名を引数にして起動する。この引数を contents_get 関数で

```
#include <stdio.h>
#include <stdlib.h>
void main(argc,argv)
      int argc;
      char **argv;
{
     char unix_path[128],pcte_path[128],buff[255];
     int lock_id;
     strcpy(pcte_path,argv[1]);
     /*Contents get*/
     lock_id = contents_get(pcte_path,O_RDWR,unix_path);
     sprintf(buff,"vi %s",unix_path);
     /*Execute vi*/
     system(buff);
     /*Contents free*/
     contents_free(lock_id);
}
```

図 C.5: カプセル化ツール evi のプログラム

UNIX のパス名に変換し、その後 system() を用いて vi を実行している[5]。

このように、contents_get 関数を用いることによって PCTE のオブジェクトを UNIX ツールで容易に操作することができるようになる。しかしながら、今回の例のように単一のファイルを操作し、また予めオブジェクトの名前が分かっていて、ファイル名を引数に起動できるようなツール以外で使用するには少々工夫が必要である。

CASE ツールインタフェースプログラミング言語の使用

CASE ツールのデータベースを操作するためのインタフェースプログラミング言語が、CASE ツールに用意されている場合がある。これを利用して PCTE リポジトリと CASE ツールのデータベースの間でデータのやりとりを行なう。

今回使用した StP/OMT では SQL 言語が、ObjectCenter では CLIPC[6]が用意されている。これらの言語を利用して PCTE のリポジトリに格納するデータの取り出し・格納するカプセル化ツールを作成することでデータの統合が可能である。

この方法では、これまでに紹介した方法がファイル単位で格納しているのと違い、自由なデータの粒度でデータを操作することができる。そのため、これまでの方法に比べ、高

[5]実際に evi を作る際にには、PCTE 上にオブジェクトが存在するか、存在しなかったら新たにオブジェクトを生成する、等の操作も実装する必要があります。今回の例では、分かりやすくするために入れてありません。

[6]CenterLine Inter-Process Communication

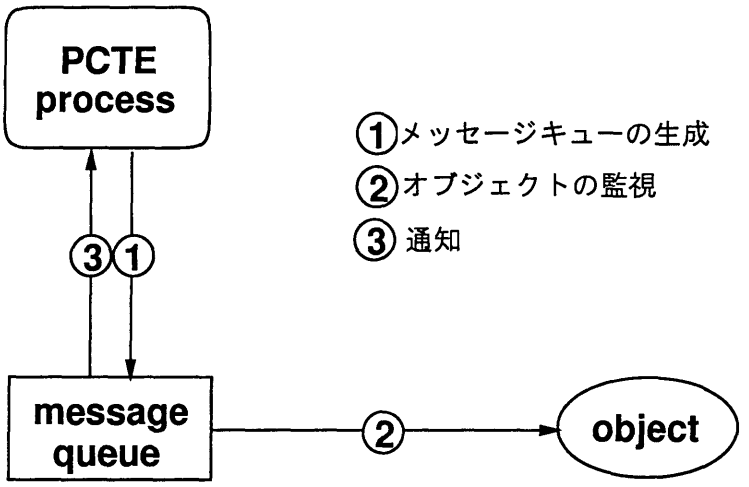

図 C.6: PCTE 通知体の概念

度なデータ統合が可能であると考えられる。

StP/OMT の SQL 言語に関する資料は、文献 [15][16] に記載されている。また、CLIPC に関する資料は、文献 [13][7]に記載されている。

notifier の使用

notifier とは、特定のオブジェクトの生成や更新・削除などを監視する通知体である。図 C.6にその概念を示す。PCTE では、PCTE 通知体という notifier が存在し、オブジェクトの生成や更新・削除などをイベントをメッセージキューに通知してくる。この通知体は、PCTE のオブジェクトを監視することは可能であるが、UNIX のファイルを監視することはできない。このような機能が CASE ツールにも存在すれば notifier を利用したデータ統合が可能である。図 C.7に StP/OMT の統合に用いた方式を示す。StP/OMT の統合では、StP/OMT が生成するファイル (.ome,.fme,.dme,.cte,.cc,.h) を監視する notifier を作成した。notifier の動作内容は、新たなファイルの生成とファイルの更新の監視である。新たなファイルの生成に関しては、notifier 内部で既存のファイルの情報を持っており、ファイル情報に無いファイルが生成すると、PCTE にオブジェクトを生成してその内容をコピーする。またファイルの更新は、notifier がファイルの更新日付を監視しており、内部のファイル情報の更新日付と相違があった場合には、PCTE のオブジェクトを更新するためにコピーを行なうようになっている。

なお、この notifier を実際に使用するにはカプセル化ツールの内部で子プロセスとして fork して使用する。

[7]ObjectCenter のインストールされているディレクトリに PS ファイルで存在する。

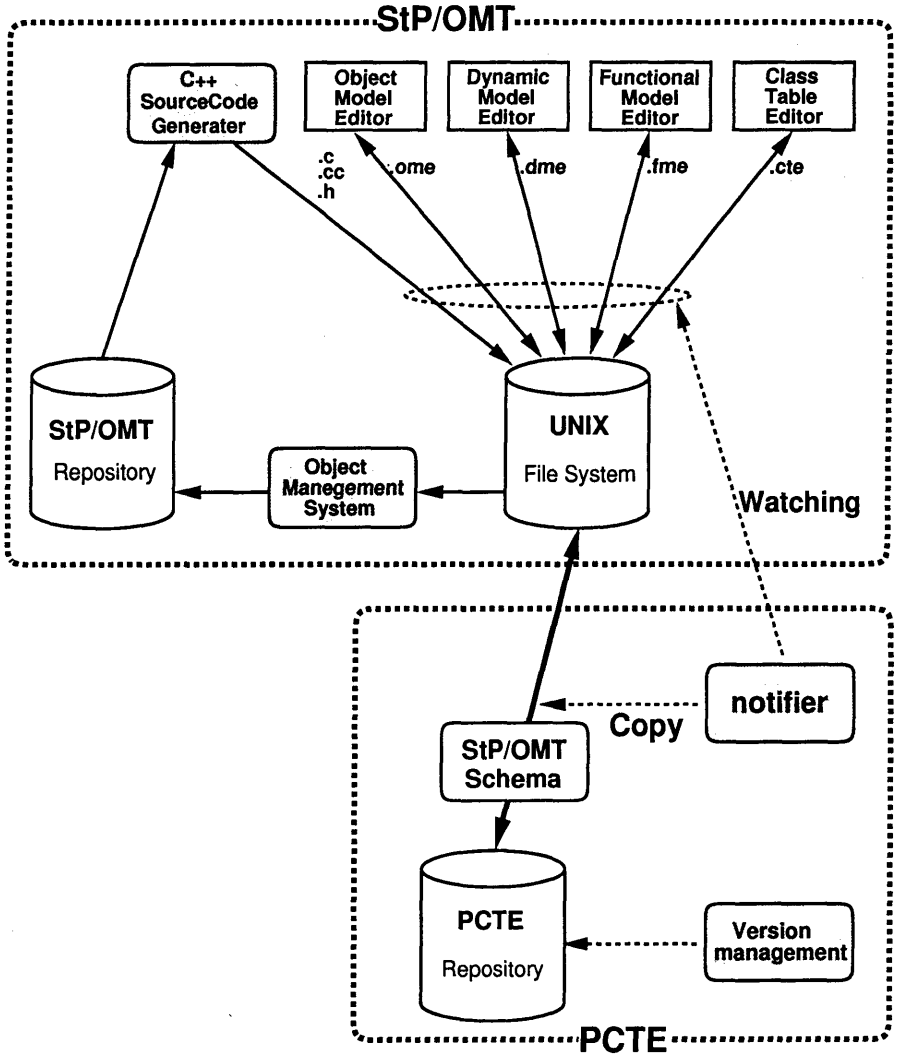

図 C.7: StP/OMT の統合方式

シンボリックリンクの使用 (import/export メカニズム)

import/export メカニズムは、作業ディレクトリから操作するオブジェクトに対しシンボリックリンクを張る方法である。import メカニズムでは、UNIX 側の作業ディレクトリから PCTE のオブジェクトへ、export メカニズムでは、PCTE の作業ディレクトリから UNIX のファイルへシンボリックリンクを張る。これらの方法では、コンテンツのある実体は1つであるので、CASE ツールからも普通に操作するのと同様の扱いをすることができる。しかしながら、この方法では新たにファイルやオブジェクトが生成するツールでは使用することができない。その理由は、起動時にシンボリックリンクを張るからである[8]。

なお、ObjectCenter のカプセル化ツールでは最初に contents_get 関数を用いてオブジェクトの UNIX パス名を得たのち、システムコール ln() を用いて作業ディレクトリからオブジェクト (UNIX パス名) にリンクを張る作業を行なう。その後、system() を用いて ObjectCenter を起動している。

ところで ObjectCenter では、新たに Makefile とプロジェクトファイルという 2 つのファイルが生成する。このように事前に生成するファイル名が分かっている場合は、カプセル化ツールの中でオブジェクトを生成しリンクを張っておくことで通常と同様に扱うことができるようになる。

C.3.5 Emeraude TCI の使用

TCI サーバーを利用して、メッセージの交換を行ないツールの統合を行なう。基本的に notifier を利用した統合と考え方は一緒であるが、TCI サーバーというメッセージ交換サーバーを介して行なう。ツール連動といったような制御統合を行なう際に利用すると、高度な統合が可能である。これらに関する資料は、文献 [9][10] に記載されている。

C.4 PCTE のバージョン管理

PCTE では、オブジェクト管理システムの一部としてバージョン管理が用意されている。これを利用して、バージョン管理の無い CASE ツールにバージョン管理を導入することが可能である。PCTE のバージョン管理には、「version_snapshot」と「version_revise」の 2 つの操作が用意されている。これらの違いは、前者が現在のバージョンの安定したものを新たに作るのに対し、後者は現在のバージョンを安定して新たに操作できるバージョンを作るという違いがある。また、これら以外の PCTE に用意されているバージョン操作の機能を表 C.2に示す。

[8] notofier のようにバックグランドで生成を監視して新たに生成したらリンクを張るようなツールを作成すれば可能。

表 C.2: PCTE のバージョン操作

version_add_predecesor	オブジェクトに新たな先行バージョンを付け加える
version_remove_predecesor	バージョングラフから先行バージョンを取り除く
version_revise	オブジェクトの変更可能なバージョンを生成する
version_snapshot	オブジェクトの安定なバージョンを生成する
version_test_ancestry	あるオブジェクトに対して別のオブジェクトが先祖バージョンであるか調べる
version_test_descent	あるオブジェクトに対して別のオブジェクトが子孫バージョンであるか調べる

また、PCTE のオブジェクト管理システムでは、リンク名の自動的な名前付けの機能を有している。これは、+という特別な値をリンク名に使用することで、最大の整数値を暗示するものである。すなわち、version.1.object と version.2.object という 2 つのオブジェクトが存在していた場合、version.+.object というのは version.2.object を指示している。また、++という値も存在し、これは最大の整数値に 1 を足した整数値を暗示している。すなわち、先ほどの例で version.++.object は version.3.object を指示していることになる。これらの機能は、バージョン管理を行なう際に最新のオブジェクトにアクセスする (+の場合)、新しいバージョンを生成する (++の場合) などの操作に用いることができる。

これらの機能をカプセル化ツールの中で使用することで、バージョン管理のない CASE ツールでもバージョン管理を導入することができる。具体的には、カプセル化ツールの中で新しいバージョンを作成した後に system() を用いて CASE ツールを呼び出す。なお、PCTE オブジェクトへのアクセスに+というキーを用いる[9]ことで、意識せずに最新のオブジェクトにアクセスすることが可能である。

PCTE のバージョン管理に関する資料は、文献 [1][8] に記載されている。

C.5 おわりに

これまでに紹介したように、PCTE による CASE ツールの統合はさまざまな方法があり、どの方法を用いたら良いかは、CASE ツールのインタフェースの特徴を見極めて使用する必要がある。そのため、ツール毎にカプセル化ツールを作る必要があり、統合の妨げとなっている。CDIF(CASE Dara Interchange Format) と呼ばれる、異なるベンダーから

[9]name.subname.k1.k2 というリンク名を使うことで高度な管理が可能である。

提供されるツール間でCASE情報を交換するための規格や、PCTEに準じたCASEツールを提供するといった動きもあるが、ベンダー間の思惑もあり順調に運んでいるわけではない。このような中で、今回の統合技術がCASEツール統合に役立つことを期待する。