ソフトウェア工学　121－10
（1998. 11.　5）

# 自分自身を編集できるオブジェクトモデルエディタ

宮崎 善史　廣田 豊彦　橋本 正明

九州工業大学情報工学部
820-8502 飯塚市川津 680-4
e-mail: yosifumi,hirota,hasimoto@minnie.ai.kyutech.ac.jp

あらまし　ソフトウェアの品質や生産性を向上させたり、ドメインの特徴を効率的に記述する方法としてドメイン分析・モデリングがあげられる。ドメイン分析・モデリングを支援するCASEツールは、ドメインの特徴をより効率的に表現するために、それぞれのドメインに特化させる必要がある。本研究では、応用ドメインに特化したCASEツールを簡単に開発する手法として、GUIオブジェクト単位で構造やソースを部品として扱い、その部品の組合せを図的な操作で行い、その結果からプログラムを自動生成することを目指している。本報告では、上記ツールのプロトタイプであるオブジェクトモデルエディタと、そのエディタを用いて作成した状態遷移図エディタについて報告する。

キーワード　CASEツール, GUI, オブジェクト指向モデル, 部品の再利用

# Object model editor which can edit own GUI

Yoshifumi Miyazaki　Toyohiko Hirota　Masaaki Hashimoto

Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502 Japan
e-mail: yosifumi,hirota,hasimoto@minnie.ai.kyutech.ac.jp

Abstract　Domain analysis and modeling is useful approach for increasing the quality and productivity of software. CASE tools that support domain analysis and modeling should be specialized in each domain to describe the domain features effectively. This study aims at making a GUI generator for domain specific CASE tools. The generator should implement operations to combine graphical parts to generate a GUI program. This report describes an object model editor which is a prototype of our GUI generator and a state transition editor which is built from the former.

key words　CASE tool, GUI, object-oriented model, reuse of software parts

## 1 はじめに

ソフトウェアの再利用は、ソフトウェアの品質の改善や、開発コストの削減などにおいて大きな効果をもたらす。しかし、再利用は容易ではなく、実際それほど一般的に行われていない。そこで、再利用をより一般的なものにするために、再利用できる構成要素の整理や、再利用することによる生産性の向上を示さなければならない。

それらの解決を導くためには、過去の成功事例や知識を集積し整理する必要があり、その手法の一つとしてドメイン分析・モデリング[1]があげられる。

本研究室では、事務処理[2]や建築設計[3]などの分野におけるドメイン分析・モデリングを行っている。さらに、それを支援するための分野に特化したCASEツールについての研究を行っており、本研究はその一環として、分野に特化した様々なCASEツールGUI部を自動生成することを目指している。

汎用のオブジェクトモデルエディタでは、現在のところプログラムの枠組や、宣言部しか自動生成することができない。しかし、本研究で作成するオブジェクトモデルエディタは、CASEツールの分野に特化することによって、あらかじめ必要なコンポーネントを準備することができ、それらを適切に組み合わせることによってCASEツールを完全に自動生成することが可能になる。

本報告では、2章で応用ドメインに特化したCASEツールGUI部の自動生成についての概要、3章ではCASEツールGUI部自動生成ツールのプロトタイプであるオブジェクトモデルエディタと、そのエディタを用いた状態遷移図エディタの自動生成、4章ではオブジェクトモデルエディタの実現方法、5章でソフトウェア再利用に関する考察を述べる。

## 2 CASEツールGUI部の自動生成

応用ドメインに特化したCASEツールは、それぞれのドメイン毎に構築する必要がある。類似のドメインや、相互に対応関係があるようなドメインに関しては、一方のドメインのCASEツールを再利用して、他方のドメインのCASEツールを開発することができるかもしれない。しかし、一般には、ドメインによってモデルの形式や、モデル上で記述される知識や制約が全く異なるので、再利用は容易ではない。特にGUI部に着目した場合、以下のような課題がある。

1. CASEツール構築の基礎となるドメインモデルは、それぞれのドメインの特質に応じた形式や論理を持っている。GUI部にもその特質が反映されることは確かであるが、GUI部には独自の形式や論理がある。したがって、ドメインモデルの枠組の中でGUI部を構築すると、十分な、使い勝手のよいものを構築するのは容易ではない。
2. GUIを構築するためのさまざまなライブラリが存在するが、それらのライブラリを適切に使ってGUIプログラムを書くことは、それほど容易ではない。また、画面上でGUIを設計し、必要なプログラムを自動生成してくれるツールもいろいろあるが、それぞれの処理内容は、生成されたプログラム中に埋め込む必要があり、プログラムなしでGUI部が構築できるわけではない。
3. GUI部の仕様を記述するための言語を用意することによって、直接プログラムを書くことにくらべると負担を軽減することができる。しかし、GUI部の仕様をテキストで記述してもあまりわかりやすいものではない。GUI部の仕様はGUIを通じて図形的に記述できることが望ましい。

これらの課題を解決するために、本研究では以下のようなアプローチをとることにした。

1. 応用ドメインに特化したCASEツールといっても、GUI部に関しては共通部分が多いと考えられる。そこで、GUI部の仕様を与えることによって、各CASEツールのために、GUI部のプログラムを自動生成するツールを開発する。
2. GUI部のプログラムを構築するのに必要なプログラム断片を部品として準備し、部品間の結合をツールで自動化することによって、利用者は直接プログラムを扱う必要がなくなる。プログラム部品としては、汎用のGUI部品と、ドメインに特化したGUI部品を用意すればよい。
3. GUIはもともとオブジェクト指向に適合していることから、オブジェクトモデル図による仕様記述を行う。2で述べた部品のオブジェクトモデルを画面上で組み合わせて、望みのGUI部を設計することができ、その設計結果から、ツールがソースプログラムを自動生成する。

本研究では、これらのアプローチに従ってGUI部自動生成ツールの開発を目指している。GUI部自動生成ツールを用いた、CASEツール開発の流れを図1に示す。本ツールのGUI部は、オブジェクトモデルエディタを基礎とし、ソースを伴うオブジェクト指向のGUI部モデル記述を読み込むことにより、エディタ上にオブジェクトモデルを表示する。そのモデルをエディタの画面上で変更・修正した結果は、再びGUI

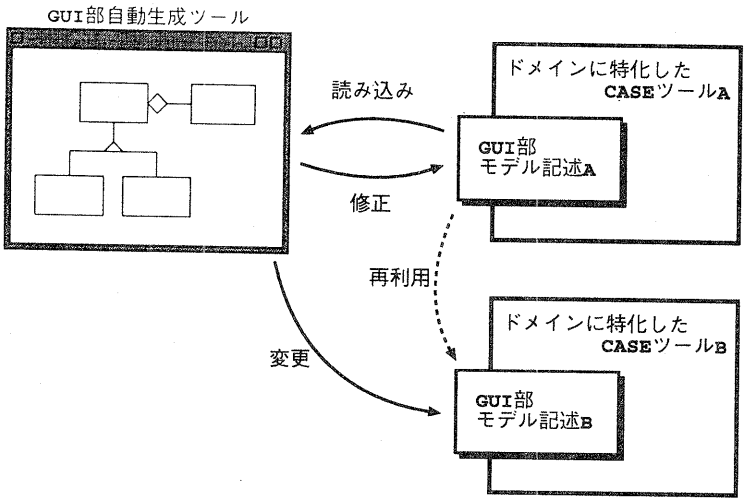


図 1: GUI 部自動生成ツールの概念図

部モデル記述へ変換することができ、そのとき必要に応じてそのモデルが持っているソースを自動的に変更することができる。さらに、1で述べたCASE ツール GUI 部の構造の共通性により、既存のモデルを再利用し必要な部分を変更するだけで、様々な種類のドメインに特化したCASE ツール GUI 部を作成できる。

このツールの実現によって、以下のような効果が期待できる。

1. 応用ドメインに特化したCASE ツールは、ドメインの分析とツールの試作を反復しながら、ツールを完成させる必要がある。また、完成後も、そのドメインをとりまく状況の変化、新たな知識の追加、ツールのプラットフォームの変更などによって、作り直しが必要となる。自動生成ツールは、このような開発の反復において、生産性を大きく向上させることができる。

2. ドメイン特有の GUI 部品には、そのドメイン独自の知識を持たせたり、そのような知識を持つ内部のドメインモデルと連係することによって、そのドメインに最も適した GUI を提供することが可能になる。

3. オブジェクトモデル図は、それ自体、仕様書として重要な役割を果たすことができ、クラス単位の再利用にとどまらず、モデル全体の再利用も可能である。

## 3 オブジェクトモデルエディタ

GUI 部自動生成ツール自身も、一種のドメインに特化したCASE ツールであるとみなすことができる。そこでまず、自分自身の GUI 部を動作例とする GUI 部自動生成ツールのプロトタイプとして、自分自身を編集対象とするオブジェクトモデルエディタを作成した。さらに、再利用と自動生成について確認するために、作成したオブジェクトモデルエディタ自身のモデル記述を変更して状態遷移図エディタを作成した。

### 3.1 オブジェクトモデルエディタの機能

オブジェクトモデルを構築する上でのクラス間の主な関連は、次の三つがあげられる。

- Inheritance … 抽象クラスと具象クラス間の関連
- Composition … クラス呼び出し (インスタンス化) をするクラスとされるクラス間の関連
- Association … 引数として参照するクラスと参照されるクラス間の関連

現段階でのオブジェクトモデルエディタでは、関連 Inheritance と Composition はサポートしており、図的にこれらの関連を操作することが可能であるが、関連 Association はサポートしていない。

表 1に、オブジェクト指向プログラムの作成に必要な機能と、現段階でのオブジェクトモデルエディタのサポート状況を示す。

表 1: オブジェクトモデルエディタのサポート状況

| 機能 | サポート状況 |
|---|---|
| モデル読み込み | プログラム起動時にモデル記述とソースを読み込む。 |
| モデル書き込み | モデル記述とソースをファイルに保存する。 |
| クラス新規作成 | モデルに新しいクラスを追加する。このとき自動生成するのはクラスの定義部分のみなのでメソッドなどは自分で追加する。 |
| クラス追加 | すでに実装されているクラスをモデルに追加する。これは、過去に作成したクラスを再利用したいときに用いる。 |
| クラス削除 | モデルからクラスを削除する。このクラスに関連する Composition や Inheritance は、自動的に削除される。これにより、このクラスのインスタンスやメソッドは利用できなくなるので、その点の変更はユーザの責任で行う。 |
| クラス名変更 | 名前の変更により不都合が生じないようなモデルの管理をしている。 |
| 具象クラス追加 | 抽象クラスに新しい具象クラスを追加する。具象クラスの定義部分は自動的に生成されるが、そのメソッドの実装はユーザの責任で行う。 |
| Composition 追加 | 2 つのクラス間に関連 Composition を追加する。ソースは自動的に更新され、インスタンス化を行うことができる。 |
| Composition 削除 | 関連 Composition を削除する。削除するとインスタンス化できないので、その点の変更はユーザの責任で行う。 |
| メソッド追加 | ユーザの責任で行う。 |
| メソッド作成 | ユーザの責任で行う。 |
| メソッド削除 | ユーザの責任で行う。 |

### 3.2 オブジェクトモデルエディタの編集

図 2は、作成したオブジェクトモデルエディタが出力した自分自身のクラス図である。このクラスにはそれぞれに対応したソースコードを持っており、クラス間の関連をつなぎ換えると、それに関する記述をしているソースコードの部分を自動的に書き換えることができる。

図 6に、オブジェクトモデルエディタの動作例として関連 Composition の表示の変更について示す。関連 Composition の表示は、左図のエディタでは CompositionView クラスで行い、線分で表示している。それをエディタの編集画面で、菱形と線分で表示する CompositionView2 クラスに変更して、保存・再起動すると右図の編集画面に変化した。

これにより、ソースコードを直接変更せずに図的な操作のみで変更・修正できることが確認できた。すなわち、クラス部品をエディタ上で入れ換えることにより、その変更箇所のソースコードが自動的に修正されたことを意味する。

### 3.3 状態遷移図エディタの作成

図 4は、状態遷移図エディタのクラス図である。このエディタは、作成したオブジェクトモデルエディタを用いて、エディタ自身の構造から必要なクラスを再利用しながら作成した。また、再利用できない部分は新たにクラスを作成して追加した。

図 5に、作成した状態遷移図エディタの実行画面を示す。これにより、オブジェクトモデルエディタを用いて、他の CASE ツールの GUI 部を作成できることが確認できた。

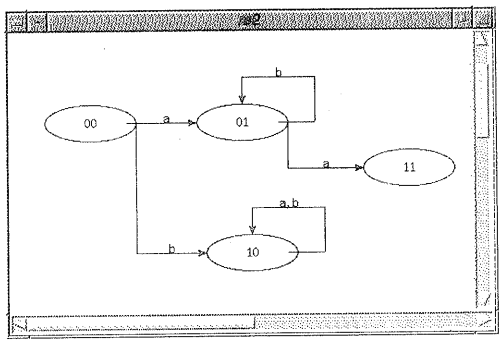


図 5: 状態遷移図エディタの実行画面

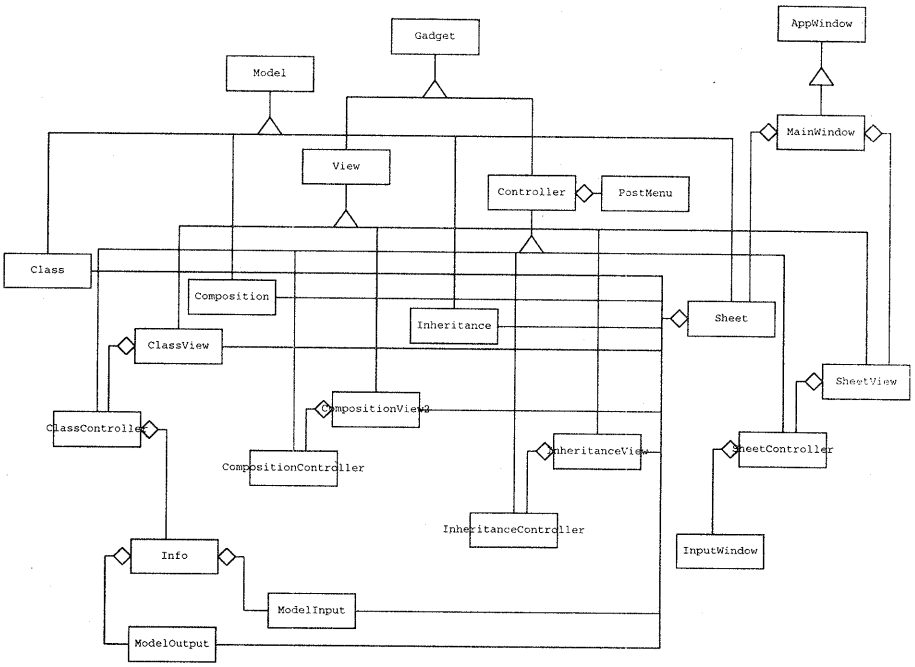


図 2: オブジェクトモデルエディタのクラス図

## 4 オブジェクトモデルエディタの実現方法

本研究では、オブジェクトモデルエディタをオブジェクト指向のスクリプト言語 Python で作成した。このエディタの実現方法を図 6で示したような部品の交換を例に説明する。

### 4.1 ソースからモデルへの変換

ソースからモデルを構築するとき主に以下のようなことを行っている。

- クラス、Inreritance、Composition の情報を取り出し各オブジェクトとして管理する。
- メソッドは、編集しないのでそのままクラスオブジェクトで保存

図 6 (a) は、ClassB のソースの一部である。このソースから ClassB のクラス情報と Composition の情報を取り出し、図 6 (b) のようなモデルとして管理する。このモデルは、エディタ上では図 6 (c) のように表示する。

### 4.2 図的な操作によるモデルの変更

ClassA で実装されているメソッドの名前と引数はすべて ClassA' と一致しているとする (メソッドの中身は一致させる必要はない )。このときエディタ上で ClassA を削除して ClassA' を読み込み、ClassA' と ClassB 間に ClassA との間にあった関連と同じ関連名を持つ関連 Composition を追加する (図 6 (d))。これにより、ClassA と ClassA' の部品が交換され、モデルは図 6 (e) のように管理される。このモデルから ClassB をソースに戻したのが図 6 (f) である。

## 5 考察

以下、ソフトウェア再利用の観点から考察を行う。

- 部品化
  本研究で行った再利用の実験は、表 2に示すように、状態遷移図エディタは 17 個のクラスで構成されており、そのうち本試作ツールを構成するクラスをそのまま再利用したのが 8 個で、一部修正して再利用したクラスが 9 個である。この状態遷移図エディタでは、一から新しく作成したクラスはなかった。これにより、CASE ツー

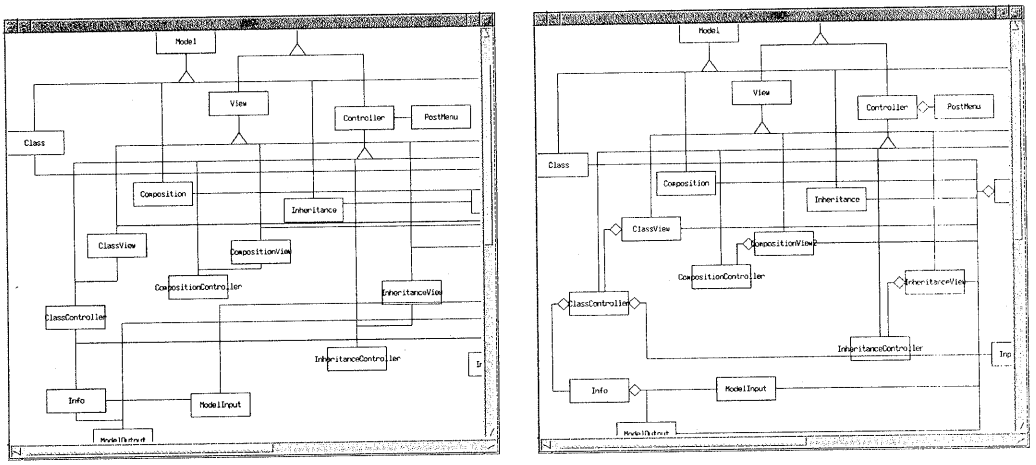


図 3: 関連 Composition の表示方法の図的な編集による変更

表 2: 状態遷移図エディタを構成するクラス

| | |
|---|---|
| オブジェクトモデルエディタ総クラス数 | 23 |
| 再利用したクラス数 | 8 |
| 新しく作成したクラス数 | 0 |
| ソースコードを一部修正したクラス数 | 9 |
| 状態遷移図エディタ総クラス数 | 17 |

ル GUI 部は部品による再利用が容易であることがわかる。今後、再利用をしやすいように構造を整理したり、部品をより多く蓄えることにより、更に再利用の範囲を増やすことが可能であると考えられる。

- **修正を伴う再利用**

本試作ツールを構成するクラスのソースを一部修正して状態遷移図エディタに再利用した 9 個のクラスは、状態遷移図エディタに特化している MVC の具象クラスである。これらのクラスは、本試作ツールの GUI 部を構成する MVC の具象クラスのインタフェースや処理の手順などを再利用しながら、そのクラスのソースコードを一部修正したものである。すなわち、CASE ツールの GUI 部はドメインに特化した部分でさえ、一から作る必要はなく、参照すべきクラスのインタフェースや処理手順に沿ってソースコードを記述していけば、簡単に新たなクラスを作成できる。

- **仕様書の再利用**

本研究で目標としているツールの利用者は、CASE ツールの開発者であり、彼らにとって、オブジェクトモデルはわかりやすい表現の一つであると考えられる。実際、GUI 部をオブジェクトモデルエディタとする本試作ツール上で状態遷移エディタを作成するとき、本試作ツール自身の構造から簡単な操作で状態遷移図エディタの構造に変換できた。このことから、オブジェクトモデルは GUI 部の仕様書としても適しており、再利用を容易にすると考えられる。

## 6 おわりに

応用ドメインに特化した CASE ツール GUI 部の自動生成と、そのプロトタイプであるオブジェクトモデルエディタと、そのエディタを用いて自動生成した状態遷移図エディタについて述べた。本研究により、図的な編集による CASE ツールの自動生成が可能であることが確認でき、さらに既存の CASE ツール GUI 部の構成要素を再利用して、新しい CASE ツール GUI 部が作成できることも確認できた。

今後、作成したオブジェクトモデルエディタを利用し、様々な CASE ツールの GUI 部を自動生成することを目指す。さらに、現在不十分

である以下の点について研究を進める。

- **メソッドと Association の管理**

3.1節にあるように、現在のエディタでは、関連 Association がサポートできていない。これは、(a) メソッドで用いる変数の型などの管理をしていない、(b) メソッドの編集をサポートしていな

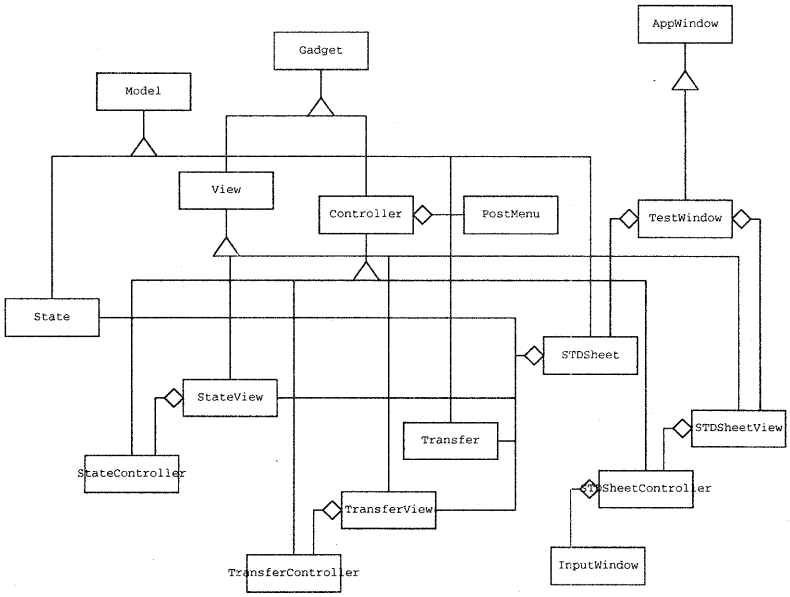


図 4: 状態遷移図エディタのクラス図

いので、ソースから関連 Association を見つけることができない、(c) 関連 Association をソースに反映させることができない、などのためである。今後、この関連 Association のサポートと、メソッドの編集をできるだけ図的な操作で簡単に行えるようにすることを目指す。

- **インタフェースの管理**
  部品間のインタフェースは常に一致させておかねばならず、その管理は現在のところユーザの責任のみである。しかし、部品の数が増えインタフェース数が膨大になるとユーザの力だけでは管理が難しくなる。そこで、インタフェースをより効率的に管理する方法が必要となってくる。

- **再利用する構成要素の整理**
  CASE ツールの分野においてよりよい再利用を提供するために、CASE ツールにおいて再利用可能な構成要素とそうでない部分とを明確に整理し、構造化する必要がある。

- **部品ライブラリの作成**
  より効率の良い CASE ツールの開発を行うために、CASE ツールに必要となるであろうと考えられる構成要素を予め部品としてライブラリ化し、それを自由に取り出せる環境を作る必要がある。

- **非 GUI 部との結合**
  本研究では CASE ツールの GUI 部の自動生成を行っており、非 GUI 部を利用する場合は、インタフェースクラスを作成して、それを利用して外部にある非 GUI 部とアクセスできるようにする。

## 参 考 文 献

1) 伊藤 潔, 杵嶋 修三, 田村 恭久, 廣田 豊彦, 吉田 裕之編. ドメイン分析・モデリング. 共立出版, 1996 年.

2) 廣田 豊彦, 田畑 宏治, 橋本 正明. 事務会計業務プロセスの分析. ソフトウェア開発のためのドメイン分析・モデリング技術シンポジウム論文集, pp. 95 103, 1996 年 9 月.

3) 廣田 豊彦, 橋本 正明, 長澤 勲. 応用ドメインに特化した概念モデル記述言語に関する一考察. 情報処理学会論文誌, 36(5):1151 1162, 1995 年 5 月.

4) 長澤 勲, 手越 義昭, 牧野 稔. IBDS: 建築物の統合化設計支援システム. 情報処理学会論文誌, 30(8):1058 1067, 1989 年 8 月.

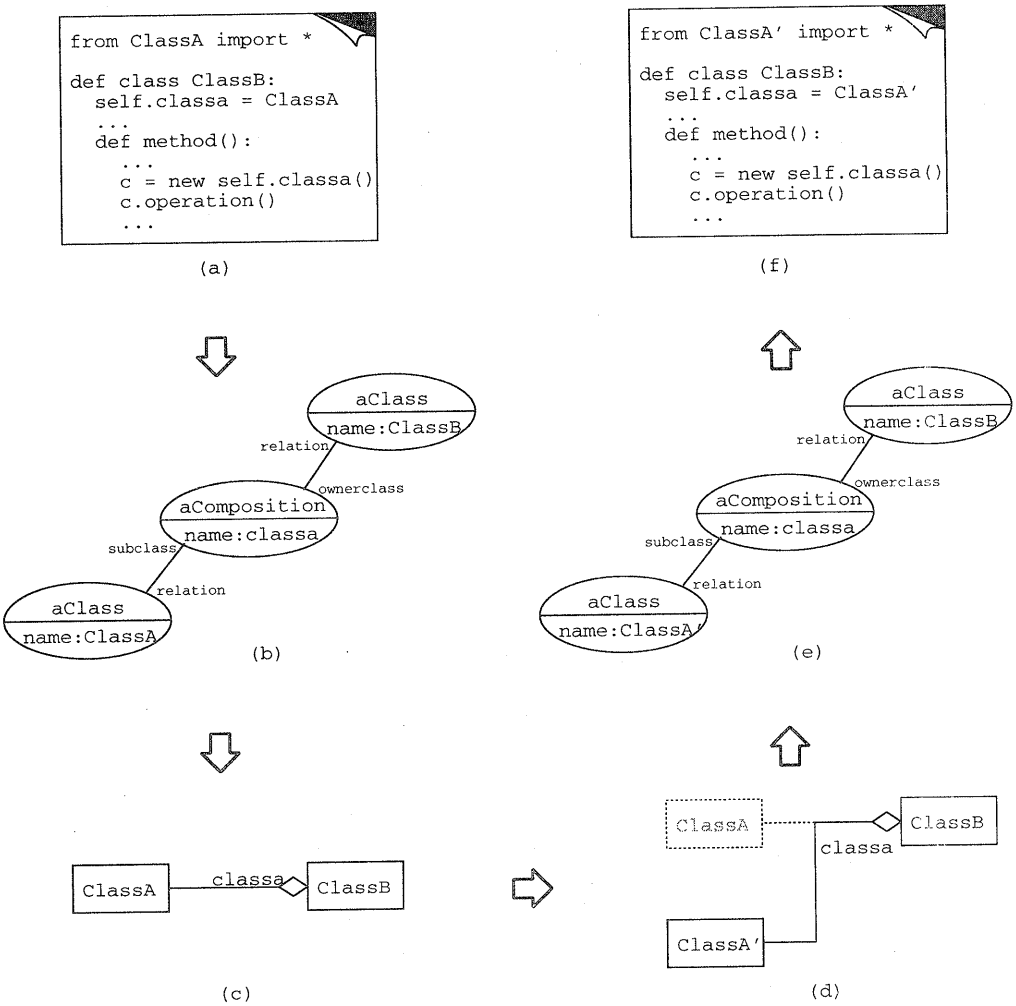


図 6: オブジェクトモデルエディタにおけるソース・モデル変換