ソフトウェア工学 118－13
（1998．3．10）

メタ階層モデル記述言語 Bramble のポータビリティの実現

庄司 龍一　中野 喜之　上田 賀一

茨城大学 工学部 情報工学科

〒 316-8511 茨城県日立市中成沢町 4-12-1

本報告では，メタ階層アーキテクチャに基づくモデリング環境とその成果物であるモデルを記述するオブジェクト指向モデル記述言語 Bramble を，マルチプラットホーム化によるポータビリティ実現のためにバイトコードを用いた仮想機械化を行なう．この Bramble 仮想機械では，GUI と CORBA に対するインタフェースを提供しているため，ユーザが GUI 構築や CORBA を通した通信を容易に行なうことができる．また，バイトコード化されたオブジェクト群は実行環境外でも存在できるため，リポジトリへの格納や取り出し、環境間の移動を行なうことが可能である．これらの機能は，Bramble 仮想機械が構築されている各プラットホームにおいて仕様が統一されているため，ユーザはプラットホームを意識することなく，Bramble でのモデル開発を行なうことができる．

Portability of Meta Hierarchical Model Description Language : Bramble

Ryuichi Shouji　Yoshiyuki Nakano　Yoshikazu Ueda

Ibaraki University

4-12-1 Naka-Narusawa, Hitachi, Ibaraki, 316-8511 Japan

In this report, we make a virtual machine of object oriented model description language : Bramble which describes the modeling environment based on meta hierarchical architecture and its products in order to realize the portability of models on multiple platforms. This Bramble virtual machine presents the interfaces of GUI and CORBA. So, users can build the GUI and make communication by using CORBA. And the byte-coded objects can be stored to/restored from repository and can be migrated between the execution environment. By this way, users can develop the models in Bramble without concern of platform, because the Bramble virtual machines on each platform is unified in specification.

1 はじめに

本研究室では，メタ階層アーキテクチャに基づくモデリング環境の構築を進めてきている．このモデリング環境はメタ階層構造を持ち，各階層を各レベルのユーザが受け持つ形態を取る．また，モデリング環境，モデルはオブジェクト指向モデル記述言語 Bramble により記述される．Bramble は，モデルの動的な記述ができ，モデリング環境のメタメタモデルが有する概念を損なわずに記述・解釈が可能なインタプリタ言語である．

我々は，このバイトコードを用いた Bramble の仮想機械を構築することにより，マルチプラットホーム化によるポータビリティの実現を試みた．またこの仮想機械には，ユーザがアプリケーションを作成する際に必要となる GUI を構築するための機能や，CORBA を通じてオブジェクトへ通信を行なう機能を備え付けることにした．これらを実現するために，仮想機械上にこれまでの Bramble 基本オブジェクトに加え，バイトコードオブジェクト，GUI オブジェクト，CORBA インタフェースオブジェクトを構築した．この時，オブジェクトの仕様は統一してあり，かつプラットホーム毎にその違いをユーザに意識させないような実装方式を用いなければならない．こうすることにより，ユーザはどの様なプラットホームでも同様に Bramble での開発を行なうことが可能となる．本研究は，そのような仮想機械の構築を目的とする．

2 基盤環境

本研究の基盤環境である，モデリング環境とその基盤となるメタ階層アーキテクチャ，成果物を管理するリポジトリシステムについて簡単に説明する．

2.1 モデリング環境

我々の研究室では，メタ階層アーキテクチャを基盤としたモデリング環境を開発してきた．モデリング環境では，モデルの生成や編集，修正を行うことができる．本環境で扱う基本的なプロダクトは，モデルである．本環境とそのプロダクトであるモデルは，オブジェクト指向モデル記述言語 Bramble により記述される．また，本環境はリポジトリシステムの一部であり，リポジトリサーバと通信することにより，システムのサービスを利用できる．

2.2 メタ階層アーキテクチャ

メタ階層アーキテクチャ[1] では，メタモデルを記述するためのメタモデルである “メタメタモデル” を定義し，モデルのメタ階層を，ベースレベル，メタレベル，メタメタレベルの 3 つの階層に分割している．これらの階層はそれぞれ，ベースレベルモデルは，そのドメインに特化したメタレベルモデルによって記述され，そしてそのドメインに特化したメタレベルのモデルは，全てメタメタレベルのモデルによって記述されている．また，メタメタモデルは，メタメタモデルによって記述されている．これにより，この階層が，モデル–メタモデルという関係で一貫して記述されることなる．

このような分割により，モデリング環境が広範囲のソフトウェアドメインをカバー可能とする．そしてモデル記述者は，適切なドメインのメタモデルを選択することによってドメイン固有の問題の記述に専念することができ，また，メタメタモデルを用意することによって，ドメインのエキスパートがドメイン固有のメタモデルを構築することを助けることができる．

2.3 リポジトリシステム

リポジトリシステムは，システム利用者が成果物を把握しやすくするために，成果物をドキュメント化して管理し，高品質な成果物を作成するために開発プロセスを支援する．このリポジトリシステムは，メタ階層アーキテクチャに基づいており，このシステムが扱う成果物は，全てモデリング環境のモデルである．また，近年の開発環境の形態である分散環境を考慮して，それへの対応も行われている．

3 モデル記述言語 Bramble

モデリング環境とモデルは，本研究室で開発が進められているオブジェクト指向モデル記述言語 Bramble[2] により記述される．Bramble は，メタ

階層アーキテクチャの概念を損なわずに実装を行なうことが可能であり，簡素な文法体系をもつインタプリタ言語である．Bramble の特徴を以下に示す．

- 全ての要素をオブジェクトとして扱う
  オブジェクトの集合は，その集合で1つのオブジェクトとして定義される．また，言語自身が振る舞うために必要な要素もオブジェクトとして持つ．
- コピーベースのオブジェクト指向の概念を用いる
  クラスやインスタンスの概念はなく，オブジェクトをコピーすることでその生成を行なう．
- オブジェクトは属性のみを含む
  オブジェクト内の変数とメソッドは区別されず，全て属性として包含される．
- メソッドをオブジェクトとする
  メソッドの役割を持ったオブジェクトをメソッドオブジェクトと呼び，メソッドオブジェクトがオブジェクトの属性となる時，この属性をメソッド属性と呼ぶ．

Bramble 上の要素は，全てがオブジェクトであるので，言語の動作はオブジェクト間の通信によって行なわれる．Bramble のオブジェクトは，以下のような基本的性質を持っている．

- オブジェクトは，メソッド (オブジェクト) を実行することによりメッセージ (と引数オブジェクト) を送信できる．
- オブジェクトは，メッセージを受信でき，それに対応するメソッドを実行する．

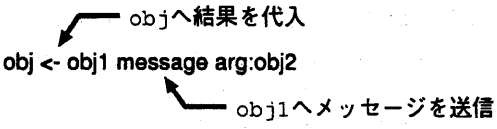
また Bramble のオブジェクトは，性質により型付けされ，Bramble 上に唯一つしか存在しないようなオブジェクトと，ユーザがオブジェクトを生成するための元となるオブジェクトタイプが存在する．前者には以下のようなものがある．

- 真偽値オブジェクト
  真偽値を示すオブジェクトで，条件分岐と論理演算に関するメッセージを受け取ることができる．
- システムオブジェクト
  新たなオブジェクト・メソッドの生成，ファイルのオープンなどを司る．
- ヌルオブジェクト
  一般に，メッセージに対して返すオブジェクトがないような時，返されるオブジェクトである．

さらに，オブジェクトの雛型であるオブジェクトタイプには，以下のようなものが存在する．

- 文字列オブジェクト
  文字列を持ち，文字列の操作に関するメッセージを受け取ることができる．
- 整数オブジェクト
  整数を持ち，整数の演算，乱数の生成に関するメッセージを受け取ることができる．
- 配列オブジェクト
  オブジェクトの参照と文字列の組を，連想配列として持ち，その処理に関するメッセージを受け取ることができる．
- リストオブジェクト
  オブジェクトへの参照をリストとして持つ．リスト処理に関するメッセージを受け取ることができる．
- メソッドオブジェクト
  オブジェクトに対する処理を持ち，それを実行するというメッセージを受け取ることができる．処理には以下のような種類がある．
  - メッセージ送信
  - 局所オブジェクトの操作
  - 文字列，メソッドオブジェクトの生成
- ファイルオブジェクト
  オープンされたファイルに関する情報を持っており，オブジェクトを読み書きできる．
- ユーザオブジェクト
  ユーザにより定義された属性を持ち，属性に対応したメッセージを受け取ることができる．属性がメソッド属性ならば，その中身が実行される．

Bramble のプログラムの基本的な要素は，メッセージ送信と代入である．以下に Bramble プログラムの例を示す．



現在，Bramble のコアは3度のバージョンアップを行ない，Bramble ver.4 となっている．

4 Bramble仮想機械

我々は，マルチプラットホーム化によるポータビリティの実現のため，バイトコードを用いたBrambleの仮想機械化を行った．本章では，そのBramble仮想機械について説明する．

4.1 Bramble仮想機械の概要

Bramble仮想機械のシステムを図1に示す．

Bramble仮想機械は，メソッドオブジェクトよりメッセージを読み取り，解釈し，メッセージの目的のオブジェクトへ送信する役割を持つメッセージエンジン，前述したBrambleの基本オブジェクトとバイトコードに関するオブジェクトを含むBramble基本オブジェクト群，CORBAを通じて，オブジェクトにメッセージを送信する際のインタフェースを提供するCORBAインタフェースオブジェクト群，Brambleを用いて視覚的な表現を実装するためのGUIオブジェクト群からなる．Bramble仮想機械は現在，UNIXとWindows95/NT上にそれぞれ構築されている．

以下では，仮想機械の内部について説明する．

4.2 バイトコード

Brambleでは全てがオブジェクトで構成されるので，バイトコードもバイトコードオブジェクトというオブジェクトとして表される．Bramble仮想機械におけるバイトコードオブジェクトは，「Bramble仮想機械外におけるBrambleオブジェクト群のイメージ」と定義される．バイトコードオブジェクトは複数のオブジェクトそれぞれのバイトコードオブジェクトの集合であり，それらの参照関係も保持される．この時，参照関係がオブジェクト群内で閉じていないこともあり得るため，このオブジェクト群外への参照の情報を持つ環境オブジェクトという概念を導入する．環境オブジェクトは実際，配列オブジェクトで表され，バイトコードオブジェクト内での環境オブジェクトへの参照は，配列のインデックスで表される．

バイトコードの内部構造は，ヘッダ，インデックス，バイトコードデータから成る．ヘッダはこのバイトコードのサイズとオブジェクトの数が含まれる．インデックスは，バイトコードデータのインデックスであり，バイトコードデータは，各オブジェクトがエンコードされたオブジェクトバイトコード，環境オブジェクトへの参照名である環境文字列，Brambleのプログラムをコンパイルしたプログラムバイトコードから成る．環境文字列は，バイトコード化されたオブジェクトが環境オブジェクトが持つオブジェクトを参照するためのインデックスである．また，プログラムコードは，メソッドオブジェクトからのみ参照される．バイトコードの内部構造を図2に示す．

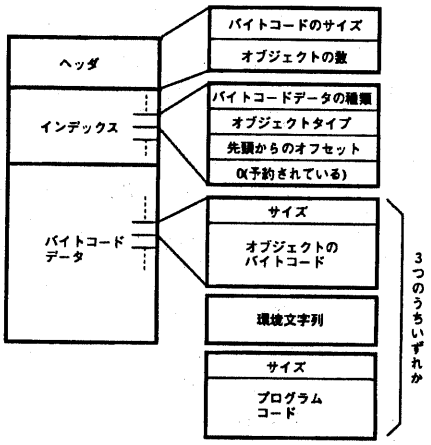


図2: バイトコードの内部構造

次に，Brambleオブジェクトからバイトコードオブジェクトへのエンコード，バイトコードオブジェクトからBrambleオブジェクトへのデコードの方法について述べる．エンコードは，以下に示す手順で行なわれる．

1. 環境オブジェクトを作成する．これによりバイトコード化するオブジェクト群の範囲が定まる．
2. バイトコード化したいオブジェクトを一つ用意する．
3. 用意したオブジェクトから，環境オブジェクト内のオブジェクト以外の再帰的に辿れるオブジェクトが全てバイトコード化される．

デコードは，引数に環境オブジェクトを与えて，

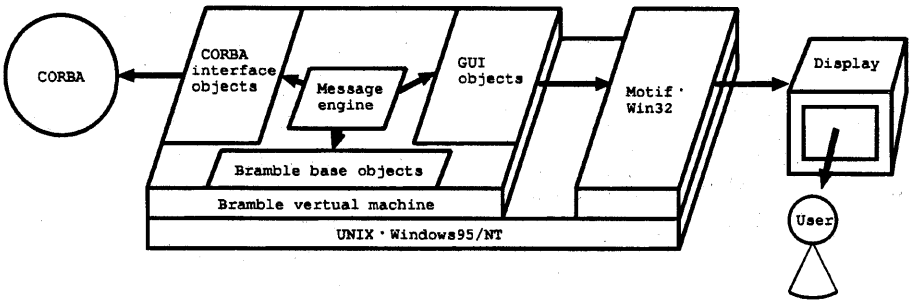


図 1: Bramble 仮想機械の構成

バイトコードオブジェクトに decode メッセージを送信することにより行なわれる.

以下に，エンコード，デコードのプログラム例を示す．また，図 3にプログラムに対応したバイトコードの生成例を示す.

```
# encode
#環境オブジェクトの生成
env <- [a:obj1 b:obj2 ....];
#バイトコードオブジェクトの生成
bytecode <- system create_bytecode
object:a_object env:env;
# decode
a_object <- bytecode decode env;
```

このバイトコード用いることにより，ファイルへの出力や，CORBA を通してのデータベースへの格納等が可能となる．現在，Bramble コアの部分のみ，バイトコードに対応している.

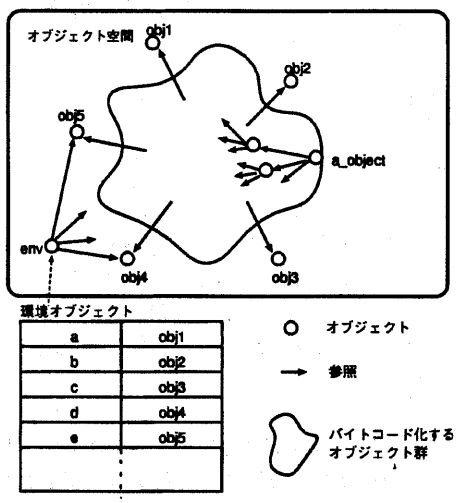


図 3: バイトコードの生成例

4.3 メッセージエンジン

メッセージエンジンは，メソッドオブジェクトの持つプログラムの処理を行なう．処理の手順は，初めに新しいメソッドオブジェクトが実行される際に Context オブジェクトが生成される．この Context オブジェクトは，線形リストとして以前に生成されたものと共に保存される．そして，プログラムが全て終了した後，そのメソッドオブジェクトをつ Context オブジェクトがリストから削除される．この Context オブジェクトは内部に，現在実行されているメソッドオブジェクトとプログラムの位置，ローカル変数領域へのポインタ，スタックへのポインタといった 4 つの情報を持つ．ここで，このスタックは一般のものとは異なり，スタックに積まれる各要素は，名前とオブジェクトへのポインタの情報を組みにして持っている．現在実行されているメソッドオブジェクトとプログラムの位置は，プログラムの取得時に用いられる．ローカル変数領域へのポインタとスタックへのポインタは，プログラムの実行時に用いられる．Context オブジェクトの構造を図 4に示す.

前にも述べたように，Bramble におけるプログラムの実行は，メッセージ送信と代入という 2 つの操作である．それぞれの操作は，次のように行

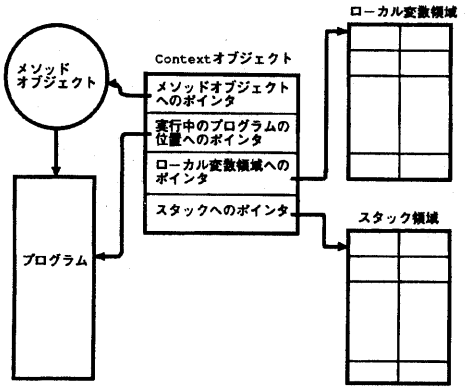


図 4: Context オブジェクトの構造

われる.

- メッセージ送信

  まず, 引数となるオブジェクトとメッセージを送信するオブジェクトを push によりスタックにプッシュする. 次に, 目的のオブジェクトに対して send により送信する. すると, 返却オブジェクトがスタックにプッシュされる.

- 代入

  代入は, store により行い, 返却オブジェクトがローカル変数領域, またはユーザオブジェクトの属性に格納される.

4.4 Bramble 基本オブジェクト群

Bramble 基本オブジェクト群は, 仮想機械上における Bramble の基本的な機能を持つオブジェクトであるシステムオブジェクト, 真偽値オブジェクト等である. メッセージエンジンより送られてきたメッセージを受け取り, 対応するオブジェクトを返す.

4.5 GUI オブジェクト群

GUI オブジェクトは, GUI を構築するための部品である. これらは, ユーザがオブジェクトを生成するための雛型であり, 文字列オブジェクト, リストオブジェクト等と同様の扱いである. また, 実際に表示される GUI の構成要素をウィジェットと呼ぶ. GUI オブジェクトは, ユーザに対し, 視覚的なモデル記述を支援することができるインタフェースを提供する.

GUI オブジェクト構築にあたり, 我々は以下のような設計方針を掲げた.

- GUI を用いたアプリケーション構築の際に必要な機能を備えていること
- どのプラットホームでも実現可能な仕様であること
- プラットホームの違いをユーザに意識させないこと

この方針に則り, 決定された GUI オブジェクトの種類と仕様を表 1 に示す.

Frame は必ず 1 つの Panel を持ち, それが唯一の子となる. Panel はウィジェットを縦, または横に配置することができ, さらに Panel を入れ子で持つことも可能である. Menu 関係のウィジェット以外のウィジェットは, 全て Panel より生成される. Menu のみ, Frame より生成される.

GUI オブジェクトは, UNIX では Motif, Windows95/NT では Win32 を用いてそれぞれ実装した.

4.6 CORBA インタフェースオブジェクト群

ネットワーク上に分散された開発環境間の通信を可能とするために, CORBA を用いて分散環境に対応させた. そのためのインタフェースが, CORBA インタフェースオブジェクト群である.

CORBA インタフェースオブジェクトを実装したことにより, リポジトリサーバと Bramble 間においてバイトコードを用いて Bramble により記述されたモデルの送受信が可能となり, リポジトリへの格納, 取り出しを行なうことができる.

5 応用例

Bramble を用いたアプリケーション例として, 履修申告表作成支援ツール, インタフェースビルダを作成した. ここでは, それらについて説明する.

表 1:GUI オブジェクトの種類と仕様

オブジェクト名	オブジェクトの説明
Frame	ウィジェットを集約する最上位のウィジェット
Panel	ウィジェットを配置するためのウィジェット
Button	押されることにより対応する動作が起動するウィジェット
Canvas	描画領域となるウィジェット
Label	文字列を表示するためのウィジェット
CheckBox	選択肢からの複数選択時に使用されるボタンの形をしたウィジェット
RadioButton	選択肢からの単数選択時に使用されるボタンの形をしたウィジェット
List	複数の要素を縦に並べるウィジェット
ScrollBar	スクロールバーの役割をするウィジェット
TextField	単数行におけるテキスト編集のためのウィジェット
TextArea	複数行におけるテキスト編集のためのウィジェット
MenuBar	フレームが一つだけ持つことができるメニューバー
Menu	プルダウンメニューとなるウィジェット
MenuItem	プルダウンメニューに並べられた個々のメニューとなるウィジェット

5.1 履修申告表作成支援ツール

本研究室において，ドメイン分析法に基づいたソフトウェア開発事例の研究が行なわれ，その一環として，茨城大学工学部における履修申告表の作成を支援するツールが Bramble によって開発された (図 5)．これは，リポジトリサーバとの通信に CORBA を用いており，データにはバイトコードを用いている．また，このリポジトリサーバに格納されるデータの形式もバイトコードである．すなわち，この応用例は Bramble 仮想機械が構築されている様々なプラットホームからのアクセスが可能なものとなっている．

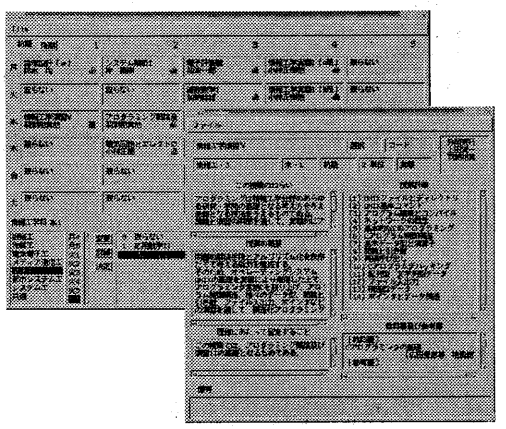


図 5: 履修申告表作成支援ツール

5.2 インタフェースビルダ

インタフェースビルダとは，ユーザが GUI を構築する際の作業を容易にするために視覚的支援を行なうツールである．このインタフェースビルダは，メインパネル，メニューリスト，ツールボックス，プロパティウィンドウの 4 つのウィジェットを持つ．

GUI 作成の手順はまず，メインパネルより追加するパネルを選択し，ツールボックスより実際に追加するウィジェットを選択する．その後プロパティウィンドウが出現するので，そこにラベル名等のウィジェットに関するプロパティを書き込み，“OK” と表示されたボタンを押すことにより，ウィジェットが生成される．メニューはメニューリストより追加するウィジェットを選択し，追加する．

ユーザが望む GUI が作成された後，メインパネルのメニューより “save” を選択することにより，プログラムがファイルに出力される．また，ファイルにプログラムを出力しただけでは GUI オブジェクトについて扱うための情報が存在しないの

で，ウィジェットオブジェクトの名前とそれ自身を組にしたものを格納した配列オブジェクトを同時に返す．これによりユーザは，GUIオブジェクトの扱いが容易になる．作成されたGUIに対応して出力されたプログラムは，他のプログラムに組み込むなどユーザが望むようにカスタマイズすることができる．

このインタフェースビルダは，メタモデルとしてリポジトリに格納され，Bramble仮想機械が構築されている様々なプラットホームにおいて実行することができる．

インタフェースビルダを用いた電卓の作成例を図6に示す．

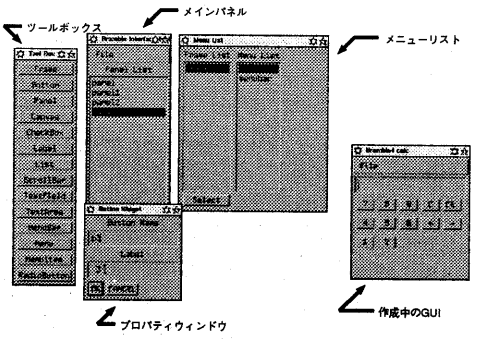


図 6: 電卓作成例

6 関連研究との比較

本研究と同様に仮想機械を用いる言語として，Smalltalk，Javaが存在する．ここでは，Brambleとそれらを比較する．

6.1 Smalltalk との比較

バイトコード化する際，Smalltalk [3] では環境全てを変換する必要があるが，Bramble は環境オブジェクトの概念を導入したため，バイトコード化するオブジェクト群の範囲を任意に指定できる．このため，Bramble は Smalltalk よりも柔軟なバイトコードへの変換を行うことが可能となっている．

6.2 Java との比較

Java [4] におけるバイトコードはクラスを変換したものであるが，Bramble では言語がコピーベースのオブジェクト指向を用いているため，存在するオブジェクトそのものがバイトコード化される．そのため，Bramble はバイトコード化を行なった時点におけるオブジェクトの状態をそのまま保持することが可能となっている．

7 まとめ

メタ階層モデル記述言語 Bramble のバイトコードを用いた仮想機械を構築し，GUI と CORBA に対するインタフェースオブジェクトを仮想機械上に実現した．これにより，Bramble 上で生成されたオブジェクト群はバイトコード化されることにより実行環境外でも存在できるようになり，バイトコードのリポジトリへの格納等を行なえるようになった．そして，ユーザがプラットホームを意識することなく GUI や CORBA を使用するアプリケーションを構築することが可能になった．また，応用例にあったインタフェースビルダや履修申告表作成支援ツールは，本研究で開発した Bramble 仮想機械を用いた実用的アプリケーションの構築が十分可能であることを示している．

今後は，他のプラットホームへの仮想機械の移植によるポータビリティの促進，各オブジェクトの仕様の見直し，言語自体の洗練を行なう予定である．

参考文献

[1] 中野 喜之，上田 賀一："メタ階層アーキテクチャによるモデリング環境の構築"，ソフトウェア工学の基礎 IV, pp.71-74, 近代科学社 (1997)

[2] 上田 賀一，中野 喜之，金村 星吉，高橋 大輔："オブジェクト指向モデル記述言語 Bramble の開発"，情報処理学会，研究報告 (SE), Vol.96, No.32, pp.65-72 (1996)

[3] L.J.Pinson, R.S.Wiener, 羽生田 栄一 監訳："Smalltalk:オブジェクト指向プログラミング"，トッパン (1990)

[4] Gary Cornell, Cay S.Horstmann, 岩本 信一 訳："コア Java"，アスキー出版局 (1997)