Common ESP におけるシステムコーディネート方式

4G－7

鳥居 悟　中澤 修　実近 憲昭
(株) AI 言語研究所

1 はじめに

Common ESP (以下 CESP と呼ぶ) [1] は, 第五世代コンピュータプロジェクトで開発された ESP 言語 [2] を基に, さらに高度化, 汎用化された, 論理型とオブジェクト指向型の二つの機能を合わせ持つ言語である.

一般に, Prolog や LISP, オブジェクト指向型言語の処理系は, さまざまなプログラミング環境を含んだものとなっている. このため, 小さなプログラムを動かすだけでも, また, デバッグの済んだプログラムの実行にも, このプログラミング環境がすべて組み込まれてしまい, さまざまな弊害を生じている.

CESP システムは, 多くのプログラミング環境を提供しているが, これらの問題に対して, ひとつの解決策を施している.

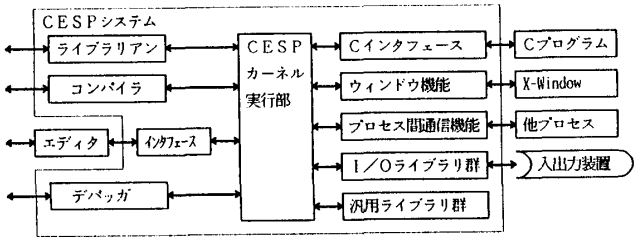
本稿では, この CESP システムの構成を自由自在にカスタマイズしたり, ユーザの実行専用モジュールを作り出すような, システムコーディネータと呼ばれるサブシステムについて述べる.

2 システムコーディネータとは

CESP システムは, 豊富な実行環境, プログラミング環境を提供している (図 1) が, これらは基本的に CESP 自身で実現されている. しかし, ユーザプログラムの実行は, 必ずしも, これら全てのシステム提供プログラム (CESP では「クラス」と呼ぶ) を必要としているとは限らない.

また, CESP システム上で開発したプログラムを汎用化させることを考えた場合, CESP システムの環境 (特にプログラミング環境) と切り離した実行モジュールが必要となる.

本システムコーディネータは,



<ライブラリ群>
=I/Oライブラリ群=
- ●標準入出力
- ●ファイル
- ●ディレクトリ
- ●ワイルドカード
- ●UNIXインタフェース

=汎用ライブラリ群=
- ●プール管理
- ● Big num
- ●部分項
- ●文字ユーティリティ
- ●シンボライザ
- ●メタ述語
- ●算術関数
- ●コールカウンタ
- ●クラス管理

図1 CESPのプログラミング環境

- ユーザが必要とするシステム提供クラスのみを取りこんだ, 独自の実行システムを生成する.
- ユーザ定義クラスの CESP 環境を生成する.

を可能とすることで, CESP システムの軽量化, ユーザプログラムの汎用化を図るものである.

2.1 選択的システム生成

システム提供クラスは, CESP ソースプログラムから, 機種に対応した機械語にコンパイルされ, それぞれ, ひとつのオブジェクトファイルとして提供されている.

これらのファイルの中から, ユーザが指定したシステム提供クラスに対応するファイルを選択し, これとオブジェクト指向実行の核となる実行部とを統合し, ひとつの実行形式を作りだすことで, 選択的にシステムを生成することが出来る. (図 2)

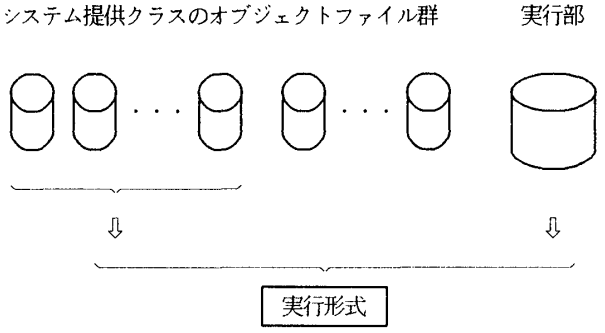


図2 選択的システム生成

2.2 ユーザ定義クラスの CESP 環境の生成

ユーザ定義クラスの CESP 環境を生成するというのは, あるユーザ定義クラスと, そのクラスが実行に必要とする他のクラスとを統合し, ひとつの実行形式を作り出すことである.

これも, 選択的システム生成の一種であるが, ここでは, ユーザ定義クラスで必要とするクラスの選択を, システムが自動的に行う. (図 3)

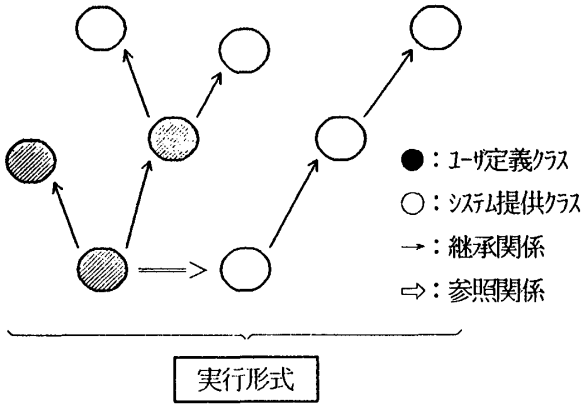


図3 ユーザ定義クラスのCESP環境の生成

System Coordinator of Common ESP System
Satoru Torii, Osamu Nakazawa, Noriaki Sanechika
AI Language Research Institute, Ltd.

この時，このユーザ定義クラスの関係するクラスを効率良く検索する方式が必要となる．検索は，各クラスが持つ種々の情報が格納してあるテーブルを使用して行う．このテーブルには，各クラスのコンパイル時に収集される情報が反映されており，システム提供クラスの情報も，前以て，このテーブルに格納しておく必要がある．

3 システムコーディネータの実現

システムをコーディネートするには，以下の3つの機能が必要である．

1) 選択する
2) 結合する
3) ロードする

3.1 選択機能

選択的にシステムを生成する時，ユーザは前以て必要なシステム提供クラスを指定する．しかし，システム提供クラス同士でも参照，継承関係があり，ユーザがこれらを意識して指定するのは大変なことである．

そこで，システム提供クラスを機能ごと，関連するクラスごとに，階層的なモジュールに分類した．

ユーザは，モジュール名を指定するだけで良く，システムコーディネータは，ユーザが指定するモジュール名から必要なクラス群を得て，それに対応する結合の対象となるファイルを見つけ出す．

3.2 結合機能

一般にクラスは，手続きの実行コードと，継承情報，内部状態（スロット），等のオブジェクト指向特有な情報とを保持している．結合の対象となる各クラスが持つこれらの情報を，ファイルの内部形式から実行に必要な形式に変換し統合して，実行部と結合可能なかたちにする．

3.3 ロード機能

選択的に生成したシステムの実行中に，システム提供クラスが不足したり，新たに必要になったり，することもある．このような場合の対処として，不足したシステム提供クラスを認識し，動的にシステム提供クラスを取り込む手段が必要となる．

そこで，未だロードされていないシステム提供クラスには，「未ロードオブジェクト」のタグを付けておく．

これで，実行時に，未ロードなシステム提供クラスを認識出来き，動的にそのクラスをロードすることも可能になる．

3.4 間接語テーブル

CESP システムでは，内部に存在しているオブジェクト，メソッドの位置を，間接語テーブルとして，一括で管理している [3]．このテーブルは，GC，再コンパイル，等のオブジェクトの再配置の手間を少なくするものである．

このテーブルを利用して，コンパイラは，コンパイル時に呼出し先オブジェクトが明白であれば，直接そのオブジェクトを呼出すコードを生成する．

このため，他クラスを継承，参照しているクラスは，コンパイル時に生成されたこれら関係クラスのテーブルのエントリ情報を，オブジェクトファイルに格納しておくだけで良い．

また，このテーブルを利用することで，未ロードなシステム提供クラスの認識は，そのクラスに対するテーブルのエントリに，「未ロードオブジェクト」が設定されているかどうかで対処可能となる．さらに，動的にクラスをロードした場合には，このテーブルのクラスに対応する情報を書き換るだけで良い．

4 システムコーディネータの構成

システムコーディネータの構成を図4に示す．

システムコーディネータは，以下の3つの部分から構成されている．

情報テーブル：システム提供クラスとそれが格納されているファイルとの対応を取る．

更新部：システム提供クラスのリンクをリンカに要求，並びに，情報テーブルと間接語テーブルの更新を行う．

検索部：情報テーブルの検索を行う．

また，情報テーブルから，以下のような情報を得ることが出来，生成される実行形式に取り込まれる．

(1) 全てのシステム提供クラスに対して，現在，何がロードされているか（いないか）の認識．

(2) 全てのシステム提供クラスに対して，そのクラス名とファイル名との対応．（この対応は，ユーザが追加，修正することが可能である．）

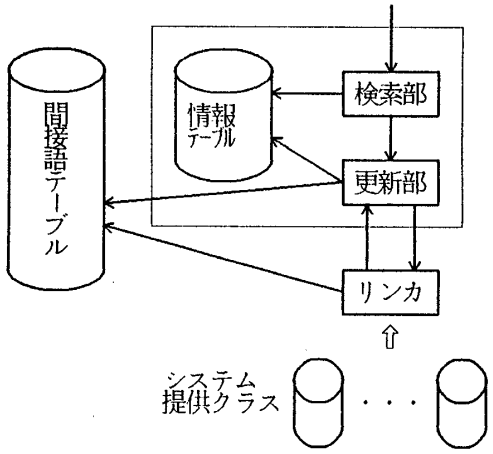


図4 システムコーディネータの構成

5 まとめと今後の課題

以上，CESP システムにおけるシステムコーディネートの方式について述べた．選択的なシステム生成を行うことで，CESP システムの軽量化，並びに，ユーザが開発したプログラムの汎用化が可能となる．

さらに，システム提供クラスのファイル形式，および，実行専用システム生成における関係オブジェクトの検索手法が，動的オブジェクトを扱う，ひとつの指針になるとも思われる．

今後は，実際的な使用環境下での評価をもとに，改良，拡張を行っていきたいと考えている．

また，オブジェクト指向を更に使い易いものとするため，動的オブジェクトの扱い，等，言語仕様とプログラミング環境を整えていくことも予定している．

6 参考文献

[1] 中澤 他：“AI システム記述用言語 COMMON ESP の構想”，情報処理学会第 37 回全国大会，1988

[2] Chikayama, T.：“Unique Features of ESP”，Proceedings of FGCS '84, ICOT, 1984

[3] 佐藤 他：“Common ESP におけるプログラミング管理システム”，情報処理学会第 39 回全国大会，1989