4M-8

ESP言語上のウィンドウ制御プログラムの開発を支援するシステムWINCS

藤本典士　今中　武　上原邦昭　豊田順一
大阪大学産業科学研究所

1．はじめに

現在のソフトウェア構築環境では、様々なライブラリ、ルーチンがシステム側から提供され、プログラマがこれらを利用できるようになっている。しかしながら、各ライブラリの仕様や利用法はマニュアルに頼るところが大きく、なかなか有効に利用できないのが現状である。このような問題点を解決するために、我々は既存のライブラリを有効にかつ容易に利用できるシステムのプロトタイプとしてWINCS (WINdow Class Synthesizer) を開発している。WINCSはPSI上で稼動するオブジェクト指向言語ESPが提供するウィンドウ関連のライブラリ（クラス）を有効利用するため、各クラスを既存部品として捉え、プログラマの機能要求に応じて、これらを選択・構成するプログラム開発支援システムである。

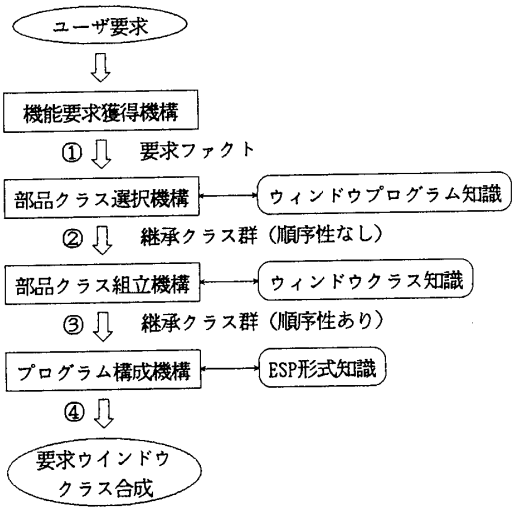

図1 本システムの構成

2．既存部品の有効利用

ESPによるプログラミングにおいて、ウィンドウを利用したプログラムを作成する場合、システム側から提供されたウィンドウ関連のクラスを必要に応じて継承し、新たなクラスを作成することになる。しかしながら、システムが提供するウィンドウ関連のクラスは、それぞれの機能によって詳細に分類され多岐にわたっているために、プログラマが必要なクラスを即時に選択することは非常に困難である。また、継承する各クラス間には継承時の順序関係や禁止事項が存在しているため、規則に従って選択した継承クラスを組み立てるにも労力を要することになる。

WINCSではプログラマのウィンドウ機能に対する要求を取り入れるために、マルチウィンドウシステムやマウスを利用し、サンプルウィンドウを例示しながら機能を選択させていく手法を用いている。また、継承に必要なクラスを選択・構成するために、ウィンドウプログラムに関する知識、各ウィンドウクラスに関する知識を用意している。さらに、EPSプログラムを記述するために、プログラム形式知識を利用している。

3．WINCS 実現

3.1 システム構成

WINCSは、図1のように、4つの合成機構と3つの知識源により構成されている。①の時点でプログラマのウィンドウに対する要求が獲得される。②の時点では、要求を満たすクラスがすべて選択されており、③の時点では、各クラスの制約条件に従って、整列している。最後に④の時点で、ESPの形式に沿ったプログラムが合成される。

3.2 知識源

WINCSでは、プログラマがプログラミング時に利用する様々な知識を次の3つの知識源に分類している。

・ウィンドウプログラム知識

プログラマの要求をもとにして、ウィンドウ関連のクラスを選択するための知識である。ユーザインタフェース部分で獲得したプログラマの各機能要求から、順次、必要なクラスを選択していくため、この知識は図2のようなプロダクションルール形式になっている。この例は出力項目について、ファイルの使用を要求している場合、"as-standard-io-window" クラスを選択する知識である。

Window Class Synthesizer WINCS

Norio FUJIMOTO, Takeshi IMANAKA, Kuniaki UEHARA, Jun'ichi TOYODA
The Institute of Scientific and Industrial Research, OSAKA Univ.

```
require(input,X) -> input(X);
require(input,X) -> outin(X);
outin(file) -> window_class(as_standard_io_window);
```

図 2 プロダクションルール

```
☆メタクラスに関する知識形式
    meta_class(メタクラス名, 順序関係)
  ＊基本メタクラスの知識例
    meta_class(fundamental,sequence(succeed,[sash]))
☆各ウインドウクラスに関する知識形式
    window_class(クラス名, メタクラス名, 順序関係,
                 排反クラス([クラス名, 例外クラス名] ‥ ),
                 結合クラス([クラス名, 例外クラス名] ‥ ))
  ＊最適画面表示クラスの知識例
    window_class(with_cwi,output,sequence(precede,[input]),
                 prohibition([output,as_markers]),joint([]))
```

図 3 ウィンドウクラス知識の表現形式

・ウィンドウクラス知識

継承クラスの継承順序を決定するため、この知識は図３のようにウィンドウ関連の各クラスの継承順序と制約条件をファクト形式として保持している。入力関連、出力関連、枠組み関連など、機能によって各クラスをメタクラスとして分類し、継承順序、制約条件をメタクラスとクラスの階層別に記述している。これは、ウィンドウクラスの継承順序は機能（メタクラス）により大枠が決定しており、また、同時に共存できない排反クラスや、必ず共存しなければならない結合クラスなどの制約条件は個々のクラスごとに異なるためである。図３の meta-class ファクトはメタクラスの順序関係を、window-class ファクトは各クラスの制約条件を表わしている。

・ESP 形式知識

プログラマの要求するウィンドウクラスを ESP プログラムの形式に従って記述するための知識である。この知識は、図４のように、ESP プログラムを階層的に記述したものである。

3.3 合成機構

WINCS では ESP プログラムを合成するための処理を４ステップに分け、次の４つの機構に割り当てている。

・機能要求獲得機構

この段階では、システムがプログラマからウィンドウに対する機能要求を獲得する。WINCS では、ウィンドウの各機能（枠組み、入出力など）について、その機能を持った各ウィンドウを実際にシステムが作成して例示し、メニュー形式で各機能を選択させる方法をとっている。これにより、プログラマの持つイメージ通りの機能要求を獲得することができる。獲得した機能要求は、どの種の機能（入出力、枠組みなど）についての要求であるかを示した要求項目と、要求内容を示した要求機能を組とする要求ファクトとして、次の段階に送られる。

```
☆プログラム階層に関する知識形式
    consist(プログラム部分, 構成要素)
  ＊クラスオブジェクト定義はスロット定義と
    メソッド定義から成るという知識例
    consist(class_def,[slot_def,method_def])
```

図 4 ESP形式知識

・部品クラス選択機構

ここでは前段階で完成した要求ファクトをもとにして、ウィンドウプログラム知識のプロダクションルールを起動する。これにより、機能要求を満たすのに必要なウィンドウクラスがすべて選択される。

・部品クラス組立機構

この段階では、選択されたウィンドウクラス全てについて、ウィンドウクラス知識を用いて、どのメタクラスに分類されるかを検索する。次に、メタクラスを meta-class ファクトをもとに配列する。最後に window-class ファクトを利用してメタクラス内での並び替えを行う。また、排反クラスが共存していないか、結合クラスが共存しているかを調べる。

・プログラム構成機構

この段階では、部品クラス組立機構で継承順序の決定したクラス群を ESP プログラムに組み込んでいく。ESP 形式知識の中から ESP プログラムの階層構造を抽象度の高いものからトップダウンに取り出して、最も詳細なプログラム部分に至ったものを、ESP の予約語や前段階で完成したクラス群に交換してプログラムを作成する。

4. まとめ

ウィンドウ関連のライブラリを有効に利用するためのシステム WINCS の実現法について報告した。WINCS は、知識源と各機構が独立しているために汎用性に優れており、他分野の知識と入れ替えることによって、様々な ESP のプログラムを合成することができる。WINCS を利用することによって、プログラミングの工数を減少させることができた。今後、要求クラス選択機構において、プログラマがより適切に機能要求を与えることができるインタフェースをつけ加える予定である。

【参考文献】

Tim Teitelbaum, Thomas Reps : "The Cornell Program Synthesizer : A Syntax-Directed Programming Environment", Communication of ACM, Vol.24 Sep. pp.563-573, 1981.