オブジェクト指向バッチプログラム合成方式

1M－6

千葉俊哉　高舘公人
(株) 日立製作所

1　はじめに

ダウンサイジングに伴い、従来ホストで行われてきたバッチ業務をWS/PC上に移行し、オブジェクト指向プログラミング言語で再構築する例が見られるようになってきた。従来のEAGLE/P[1]のプログラム自動生成の特長を生かしつつ、オブジェクト指向プログラミング技術の良さを加え、より高い柔軟性や保守性を実現するオブジェクト指向バッチプログラム合成方式を開発した。

まず、バッチプログラムに普遍的に見られる次の２種類のクラス群をクラスライブラリの形で用意しておく。

(1) データ入出力クラス群

ファイル、プリンタ等のデータ入出力デバイスに対応し、レコードの入出力動作を行なう。

(2) フィルタクラス群

マージ、ソート等、バッチ処理の基本的なレコード処理を行なう。

上記２種類のオブジェクト群をユーザの仕様定義情報よりカスタマイズし、繋ぎ合わせる事でバッチ処理プログラムを合成する。

2　合成方式概要

構成図を図1に示す。

本方式では、バッチプログラム一般において、アプリケーションに非依存な部分をあらかじめクラスライブラリの形で用意しておき、これを再利用する。使用するファイルの書式、オブジェクトの組み合わせ等、アプリケーション依存の情報は、ユーザが定義した仕様定義情報を元に、クラスライブラリのサブクラスおよびオブジェクト制御ルーチンの形で生成する。生成されたコードとクラスライブラリをコンパイル／リンクする事により、完成プログラムを得る。

Mechanism of Object-Oriented Batch-Programs Generation
Toshiya Chiba , Masato Takadachi
Hitachi, Ltd.

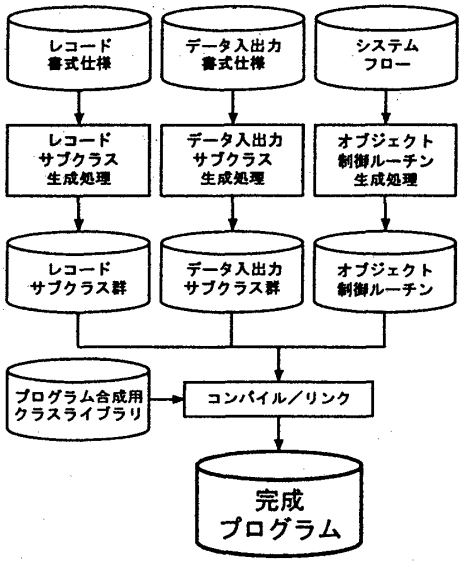

図 1: 構成図

3　合成用クラスライブラリ

本プログラム合成方式は、レコードを入出力するオブジェクト(データ入出力クラスのインスタンス)とレコードに対し定型的処理を行なうオブジェクト(フィルタクラスのインスタンス)を自由に組み合わせ、少量の仕様定義情報で容易にプログラムを合成する事を可能にしている。データ入出力クラス群およびフィルタクラス群のインスタンス群を自由に組み合わせる事を可能にするために、これら２種類のクラスの共通親クラスとしてレコード媒介抽象クラスを持たせた。

レコード媒介抽象クラスは、データ入出力クラス群およびフィルタクラス群の、レコードを互いに受け渡すという共通の性質を定義する抽象クラスであり、レコード書き込みメソッドとレコード取得メソッドを持つ。レコード媒介クラスのサブクラスであるデータ入出力クラス群およびフィルタクラス群が、レコード媒介クラスのインスタンスとの間でレコードを受け渡すように設計することにより、データ入出力クラス群およびフィルタクラス群のインスタンス群を自由に組み合わせてバッチ処

理を行なわせる事を可能にした。

レコード受け渡しのメッセージインターフェースを統一するためには、インターフェースがレコードの書式に非依存となるよう、設計する必要がある。本方式では、インターフェースを「レコード抽象クラスに属するオブジェクトの受け渡し」とする事で統一を実現した。

レコード抽象クラスは、レコードをオブジェクトとし本オブジェクトのメッセージインターフェースを規定する抽象クラスである。メッセージの種類は、フィールドデータ項目取得、キー項目での大小比較等、ごく少数に限定している。メッセージインターフェースを統一するためには、インターフェースが、メッセージで受け渡しされるフィールドデータ項目のデータ型に非依存となるよう、設計する必要がある。上記同様の考えで、インターフェースを「データ項目抽象クラスに属するオブジェクトの受け渡し」とする事で、統一を実現した。

データ項目抽象クラスは、データ項目をオブジェクトとし、本オブジェクトのメッセージインターフェースを規定する抽象クラスである。メッセージの種類は、文字列と内部データとの間の相互変換、データ正当性検査、大小比較などである。

バッチ処理プログラムにおいてデータ入出力に用いられるデバイスはファイル、プリンタなど、非常に限られている。またバッチ処理は、レコードのマージやソートなどの定型的な処理を組み合わせる事で殆んどのものが実現可能である。さらに、金額や日付など、多くのアプリケーションで共通に使用されるデータ項目がある。これらに対応し、レコード媒介抽象クラスのサブクラスとしてデータ入出力クラス群とフィルタクラス群、データ項目抽象クラスのサブクラスとしてデータ項目クラス群を幾つか用意した。これらを選択し、組み合わせて再利用し、必要に応じてアプリケーション依存の情報を付加することにより、アプリケーションプログラムを実現する。どのクラスを再利用し、どのような情報を付加するかは、ユーザ仕様定義情報より決定される。

4 ユーザ仕様定義情報と生成コード

ユーザが定義する仕様情報は、次の3種類である。

(1) レコード書式仕様

レコードにおけるデータ項目の並びを定義する。本定義より、レコード抽象クラスのサブクラスが生成される。

(2) 入出力書式仕様

データ入出力デバイスで入出力されるレコード、入出力書式を定義する。本定義より、データ入抽象クラスのサブクラスが生成される。

(3) システムフロー

データ入出力デバイスと、データ処理との間のデータの流れを記述する、一種のデータフロー図であるが、図の情報に位置する処理から処理が行われ、データの流れを上方から下方への一方向としている点に特徴がある (記述例：図 2)。本図におけるノードは、データ入出力オブジェクト又はフィルタオブジェクトに対応しており、オブジェクト間のつながりの情報が定義される。本定義からは、オブジェクトを生成し、それらのつながりの関係を初期化する、オブジェクト初期化ルーチンを生成する。

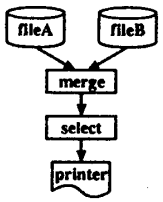

図 2: システムフロー図記述例

5 おわりに

オブジェクト指向プログラミングによるバッチ処理プログラムの合成方式を開発した。クラスライブラリにあらかじめ用意したデータ入出力オブジェクトと、フィルタオブジェクトをユーザ仕様定義情報によりカスタマイズ (サブクラス生成) し、それらを組み合わせるコード (オブジェクト初期化ルーチン) を併せる事で、完成プログラムを得る。

データ入出力オブジェクトと、フィルタオブジェクトとの自由な組み合わせを可能とするためにこれらオブジェクトで受け渡すレコードをオブジェクトとして抽象化し、メッセージインターフェースを統一した。

6 参考文献

[1] 降旗他：「EAGLE/P ディクショナリを用いたプログラムジェネレータの開発と適用効果」情報処理学会第 43 回全国大会, 1K-1 pp.5-381 ～ 5-382, 1991