1N－9

オブジェクト指向開発アプローチ Crossover (2) - 生成系 -

野田夏子, 前川佳春, 岸 知二
NEC マイコンソフトウェア開発研究所

1 はじめに

我々はオブジェクト指向開発アプローチ Crossover の検討を進めている [1]。本アプローチにおいては、部品を用いた設計モデルから実行可能なコードを生成する必要がある。本稿では本アプローチにおける生成系の要件とその方式について論じる。

2 生成系の要件

生成系の要件を以下に示す。

- **プラットフォームの利用も含めた生成をすること:** 一般に、実行可能なコードの生成を行う場合、設計モデル中の概念を対象言語に単純に置き換えるだけでは不十分であり、OS 機能や通信機能等を利用しなければならない。例えば、C++ 自体には並行性の概念がないため、モデル中の並行性を実現するためにはプロセス等 OS の機能を必要とする。

- **物理世界との対応付けが可能であること:** リンク指示を生成したり既存ライブラリを参照する際には、物理的なファイル名が必要となる。したがって、論理的なモデルが物理世界とどう対応付けられるかを管理できなければならない。

- **生成のカスタマイズが可能であること:** 対象環境 (言語、OS 等) によって生成方法が異なるため、それらに応じたカスタマイズが可能でなければならない。

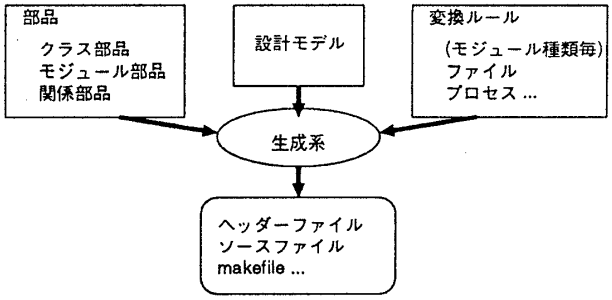
"Object-Oriented Development Approach: Crossover (2) - Generator -", Natsuko NODA, Yoshiharu MAEGAWA and Tomoji KISHI, NEC Corporation

3 生成方式

3.1 生成系の全体像

生成系の全体像を次に示す。

図 1: 生成系の全体像



- **設計モデル:** インターフェース記述言語で記述されたモデル。

- **部品:** クラス部品、モジュール部品、関係部品がある。

  クラス部品、モジュール部品: 対応するヘッダーファイルやライブラリの所在の情報を持つ。

  関係部品: 現段階で我々は、関係部品を次の 2 つに分類している。

  直接関係部品: 関係に参加するクラスが相手に対するリファレンスを直接保持する場合の関係部品。相手のリファレンスを管理する Set や List のような関係管理クラス の情報を持つ。

  間接関係部品: 関係に参加するクラスが間接的に相手に対するリファレンスを保持する場合の関係部品。(1) 関係管理クラス、(2) 関係管理クラスによって管

理され、間接的に相手を示す被管理クラス (ポート番号等)、(3) 被管理クラスから相手を参照する手段を提供するユーティリティモジュール (ソケットによる通信手段を提供するモジュール等) の情報を持つ。

- 変換ルール: モデルをどのようなファイルやコードに変換するのかを記述したルール。モジュール種類毎に定義される。モジュール種類は対象環境に応じて用意する。

上記の情報から生成系は次のような処理を行う。

- 部品の展開: クラス部品、モジュール部品については、実体のヘッダーファイルやライブラリの所在を求め、include 宣言や makefile へ反映させる。

  関係部品に関しては、関係に参加するクラスに対して関係管理クラスを属性として宣言し、さらにユーティリティモジュールをそのクラスが属するモジュールの include 宣言に反映させる。

- モジュール毎の形式変換: モジュール種類毎に定められたルールに従って、インタフェース記述言語からターゲットのコードへと変換を行う。

なお、生成のカスタマイズは部品と変換ルールの修正で行う。

3.2 生成例

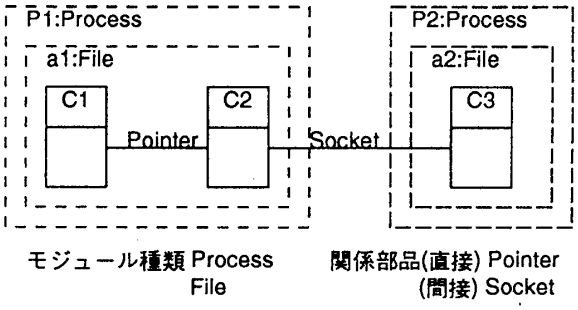
関係部品の展開を中心に、UNIX 環境における生成例を考える。設計モデルとして図 2を考える。

UNIX 環境に対しては Process、File をモジュール種類として用意する。Process モジュール、File モジュールに対して定められている変換ルールの内容は以下のようになる。

- Process モジュール: 自分に含まれる File モジュールの名前やインポート関係から make-file を生成し、さらにプロセスの main 関数を含むソースファイルを 1 つ生成する。

- File モジュール: ヘッダーファイルとソースファイルをそれぞれ 1 つずつ生成する。

図 2: UNIX 環境における設計モデルの例



したがって、このモデルからは、モジュール P1、P2 それぞれに対する main 関数を含むファイル P1Main.c、P2Main.c と makefile make.P1、make.P2、モジュール a1、a2 それぞれに対応するヘッダーファイル a1.h、a2.h とソースファイル a1.c、a2.c が生成される。

なお、関係は次のように展開される。

- 直接関係部品 Pointer の展開: クラス C1 はクラス C2 のポインタを属性として持つように宣言される。

- 間接関係部品 Socket の展開: C2 は C3 のポート番号を属性として持つことで C3 を識別し、C2 を含む a1.h がソケット通信の手段を提供するライブラリを include する。

4 おわりに

現状では関係部品の種類を 2 つに限定しているが、これで十分かどうか整理する必要があり、これは今後の課題である。

参考文献

[1] 岸知二、他: オブジェクト指向開発アプローチ Crossover (1) - 設計モデル -. 情報処理学会第 49 回全国大会 (1994).