2K－8

オブジェクト指向開発アプローチ Crossover (4) - 設計モデル II -

野田夏子, 前川佳春, 岸 知二
NEC マイコンソフト開発環境研究所

1　はじめに

我々は、オブジェクト指向分析、設計方法論とオブジェクト指向プログラミングの両方の利点を取り入れた開発アプローチ Crossover の検討を進めている [1] [2]。これは基本構造を実際のコードとして設計モデルから早期に生成し、その枠組みの中で徐々に内容を詳細化していくアプローチである。このアプローチを実現するための設計モデルのうち、[1] ではインタフェース記述について述べた。本稿では、インタフェースと実装との関係を含め、設計モデルの全体像について論じる。

2　インタフェースと実装との関係

設計モデルは以下から構成される:

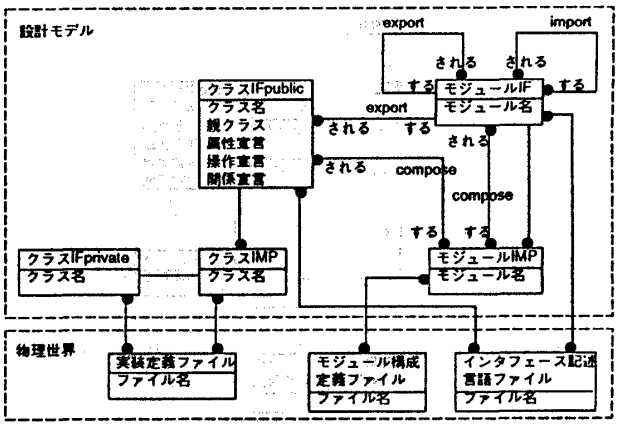
- モジュールインタフェース (モジュール IF) : モジュールを構成するモジュールもしくはクラスのインタフェースのうち export するものと、import するモジュール IF とを示す。
- モジュール実装 (モジュール IMP) : モジュールを構成するモジュールもしくはクラスのインタフェースの集合を示す。1 つのモジュール IF に対し、複数存在し得る。
- クラス公開インタフェース (クラス IFpublic) : クラス IMP に対する公開インタフェース。クラス IMP と 1:1 に対応する。
- クラス実装 (クラス IMP) : 1 つのクラス IF が複数のクラス IMP を持つことができる。
- クラス非公開インタフェース (クラス IFprivate) : そのクラス自身の実装のみが必要とするクラスの非公開インタフェース。クラス IMP と 1:1 に対応する。

クラスの実装は様々な定義方法 (状態モデルでの記述、プログラミング言語での記述等) があるため、設計モデル中では定義方法を規定しない。クラス IMP、クラス IFprivate は単に定義の所在管理のみを行う。

特定のインタフェースに対して複数の実装を許しているため、例えば設計初期に状態モデルとして与えた実装と、その後言語にブレークダウンした実装との対応関係を管理することができる。

ファイルシステム等の物理世界との対応関係を含めた設計モデルの全体像を図 1 に示す。なお、モジュール IMP とそのモジュールを構成するモジュール IF もしくはクラス IFpublic の間の関係を、compose 関係と呼ぶ。

図 1: 設計モデルの全体像



3　明確な構成

設計モデル中には 1 つのインタフェースに対して複数の実装が対応付けられており、設計モデルから以下の手順で特定の構成を求めることができる:

1. モジュール IF を 1 つ選ぶ。

"Object-Oriented Development Approach: Crossover (4) - Design Model II -", Natsuko NODA, Yoshiharu MAEGAWA and Tomoji KISHI, NEC Corporation

2. そのモジュールIFに対応するモジュールIMPを1つ取り出す。

3. そのモジュールIMPがcomposeするクラスIFpublicに対し、それぞれクラスIMPを1つ定める。またモジュールIFがimportするモジュールIF、モジュールIMPがcomposeするモジュールIFに対し、それぞれモジュールIMPを1つ定める。

4. 全てのモジュールIF、クラスIFpublicについて1つずつ実装が対応付けられるまで、上記を繰り返す。

以上の手続きにより得られた構成が以下の条件を満たす時、それを明確な構成と呼ぶ:

- **親クラス、属性、操作の宣言に必要なモジュールIFのimport:** クラスの親クラス、属性、操作の戻り値、操作の引き数の宣言に必要なクラスIFpublicは、クラスをcomposeするモジュールIMPにcomposeされているか、そのモジュールIMPに対応するモジュールIFがimportするモジュールIFにおいてexportされていなければならない。

- **関係の宣言に必要なモジュールIFのimport:** クラスIFpublicの関係の相手先クラスは、クラスIFpublicをcomposeするモジュールIMPにcomposeされているか、そのモジュールIMPに対応するモジュールIFがimportするモジュールIFにおいてexportされていなければならない。

- **inverse宣言のある関係の整合:** クラスIFpublicがinverse宣言のある関係を持つならば、相手のクラスIFpublicもそのクラスとinverse宣言のある関係を持たなければならない。

4 部品

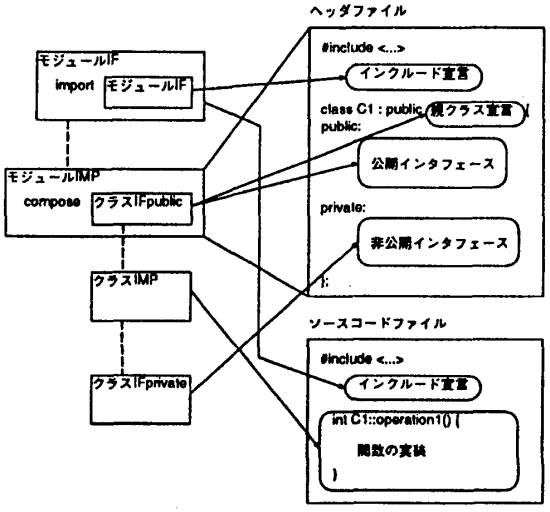
あるモジュールIFを起点とする明確な構成が存在する時、そのモジュールIFとモジュールIMPが表すモジュールをモジュール部品とすることができる。上記手順においてimport/composeするモジュールIFがモジュール部品のモジュールIFであれば、それに対して上記手順を適用する必要はない。

また、関係に関係部品[2]を対応付ける場合には、関係管理クラスをexportしているモジュールIF、ユーティリティモジュールとして必要なモジュールIFをimportしなければならない。

5 生成されるコードとの対応

明確な構成からコードを生成する。C++のコードを生成する場合、設計モデルの構成要素とコードとの対応関係例を図2に示す:

図 2: 設計モデルと生成されるC++コード



6 おわりに

部品の管理空間のモデル化等は、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).

[2] 野田夏子、他: オブジェクト指向開発アプローチ Crossover (2) - 生成系 -. 情報処理学会第49回全国大会 (1994).