3W-07 組み込みソフトにおけるオブジェクト設計法*

加藤 丈治　　長谷川 賢一　　村松 昭男

富士通株式会社 情報処理事業推進本部 ファームウエア開発推進統括部 開発部

1 はじめに

近年, 組み込みソフト (以下, ファームウエアと記す.) の規模の増加に伴い, ファームウエアの部品化 (コンポーネント化)/再利用が不可欠なものとなりつつある.

しかし, 現在, 最も普及している部品化/再利用化技術の一つであるオブジェクト指向設計手法に基づくコンポーネント設計/構成手法は, ファームウエア設計に適用するには不十分である.

本論文では, 既存のオブジェクト指向設計手法における構成設計面の問題を明らかにし, その解決法を提案する.

2 オブジェクト指向設計手法の問題点

現状のファームウエア開発を考慮すると, 既存のオブジェクト指向設計手法には以下のような問題が存在する.

頻発する仕様変更に対応できない 開発過程において制御対象となるハードウエアの仕様が頻繁に変更される. しかし, 既存のオブジェクト指向設計手法の多くは, オブジェクト抽出からトップダウンに設計を進めていくため, 仕様変更にともなう構成変更は困難である.

装置固有の最適化を記述できない 装置に搭載されている資源 (メモリなど) が非常に限られるため, 各装置の実行環境に合わせた設計の最適化が必要となる. しかし, 既存のオブジェクト指向設計手法では, 機能分割に基づく設計しか考慮していないため, 装置に固有な設計の最適化を明示的に記述できない.

3 提案する設計手法

本論文では, 第 2 節で述べた問題を解決するために**構成ルールに基づくオブジェクト構築法**を提案する. これは, まずファームウエアの構成を手続きやデータ群の集合として表し, それらの手続き/データに対して**構成ルール**と呼ばれるルールを適用することでオブジェクトを抽出/構築する設計手法である.

3.1 構成ルールに基づくオブジェクト構築法

提案手法では, オブジェクトを以下の手順で構築する.

1. **プログラム関連図作成**
   開発対象となるプログラム内のプログラム関連図[1]を作成する.
2. **ドメイン分析**
   プログラム関連図を基に各データや手続きと各ドメイン[2]との依存関係を調査する.
3. **仮オブジェクト抽出**
   プログラム関連図内の各手続き, 大域データと一対一に対応するようなオブジェクト群 (これを**仮オブジェクト**と呼ぶ.) を定義する.
4. **構成ルール策定**
   3.で定義された仮オブジェクトを最終的なオブジェクト (コンポーネント) に変換する手順を構成ルールとして定義する.
5. **構成ルール適用**
   4.で策定したルールを仮オブジェクトに適用し, オブジェクト (コンポーネント) を構築する.

この手順を図示すると, 図 1のようになる.

3.2 構成ルール

各構成ルールは, 次の 2 種類のメタクラスによって表される.

オペレーション オブジェクトの定義を変更する操作 (メソッドの実装部を置き換えるなど) を表すメタクラスである.

*An Object Design Method for Firm-ware Developments, Takeharu KATO, Kenichi HASEGAWA, Akio MURAMATSU, Fujitsu Limited

[1] 手続き呼び出しの流れや手続き間のデータの流れを表す図のこと.

[2] 装置の構成要素を表す概念的な空間のことを指す.

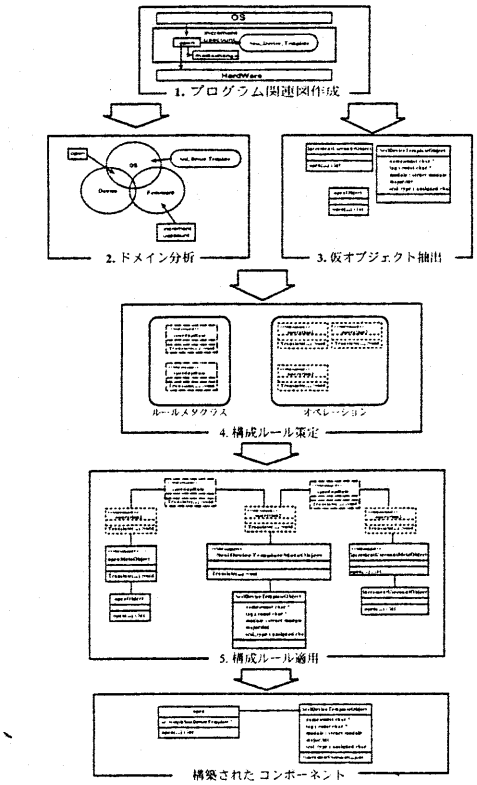

図 1: オブジェクトの構築手順

ルールメタクラス ルールメタクラスは, 仮オブジェクトをコンポーネントに変換するメタクラスであり, 各構成ルールごとに用意される. 各ルールメタクラスは, **変換メソッド** をもつ. 変換メソッドには, 仮オブジェクトからコンポーネントへの変換手順が定義される.

変換メソッドは, 構成ルールの適用条件と仮オブジェクトに対する変換操作を表すオペレーションのメソッド列とからなる. 変換メソッドの適用条件には, ドメインに基づく依存関係に関するルールを記述する. 変換メソッドの例を, 図2に示す.

```
void  TranslateClass(···){
        //適用条件 (AppCond 内の条件が真のときに実行する. )
        //双方のオブジェクトが"I/O"ドメインに属する場合に適用.
        AppCond  (inRelation(src,"I/ODomain") &&
                  inRelation(obj,"I/ODomain"))
          {
            //変換操作の定義
            //2 つの手続きの実装部を結合する
            //concatBody はオブジェクトのメソッドの実装部を
            //結合した結果を返すオペレーション
            body = concatBody->Concat( src, obj,
             srcMthd, apdMthd);
            //メソッド追加
            appendMthd("spdup_" + srcMthd);
                          :
            //メソッドの実装部を再定義
            setMember("spdup_" + srcMthd)-> setBody(body);
          }
        }
```

図 2: 変換メソッドの例

4 考察と結論

提案手法では, 構成ルールを適用することでオブジェクトを構成する. これにより, 適用するルールを変更するだけで, 仕様変更による構成変更に簡単に対応することが可能である.

さらに, 構成ルールをメタオブジェクトとして定義しているので, 仮オブジェクトのメソッドの実装を再定義することができる. これにより, 装置固有の最適化を構成ルールの一つとして記述することで, 機能的側面に基づくオブジェクトの定義と装置固有の最適化を考慮したオブジェクト定義とを構成ルールによって明示的に区別することが可能となる.

現在, 予期しない仕様変更に対応するために, 特定のルールにしたがってオブジェクトの構成設計を行う手法として, "Subject Oriented Programming"[1] が提案されている. また, 実行環境固有の最適化を考慮した手法として, "Aspect Oriented Programming"[2] が提案されている. しかし, これらの手法ではルールの策定者の意図に反したルールの適用を防ぐ手段を考慮していないため, 設計の変更に伴う設計品質の低下 (ドメインを無視したオブジェクト構築による再利用性の低下など) が起こり得るという問題があった.

一方, 提案手法では, 手続きおよびデータのドメインに対する依存関係に基づいて, 変換メソッドに構成ルールの適用条件を記述できるため, ルール策定者の意図に反するルールの適用を抑制することも可能である.

以上から提案手法は, ファームウエア設計に必要な特性を満たし, かつ, 構成ルールの適切な運用を促すことのできる今までにない特徴を持った設計手法であるといえる.

参考文献

[1] Matthew Kaplan, Harold Ossher, William Harrison, and Vincent Kruskal. Subject-oriented design and the watson subject compiler. In *OOPSLA96' Subjectivity Workshop*, oct 1996.

[2] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *Proceedings of ECOOP*, jun 1997.