参照モデルに基づくオブジェクト指向フレームワーク開発手法

正員 早瀬 健夫 ((株) 東芝 情報・社会システム社 SI技術開発センター)

非会員 池田 信之 ((株) 東芝 情報・社会システム社 SI技術開発センター)

非会員 松本 一教 ((株) 東芝 情報・社会システム社 SI技術開発センター)

非会員 守安 隆 ((株) 東芝 情報・社会システム社 SI技術開発センター)

Developing Object-Oriented Frameworks Based on Reference Models

Takeo Hayase, Member, Nobuyuki Ikeda, Non-member, Kazunori Matsumoto, Non-member, Takashi Moriyasu, Non-member (System Integration Technology Center, Information and Industrial Systems & Services Company, TOSHIBA Corp.)

This paper describes a method for developing object-oriented frameworks based on reference models. Reference models are a reuse technique to represent directly the logical structure of a domain. A framework embodies both an abstract design that is common to all applications from the domain and hot spots that allow the framework to be customized to application-specific requirements. It is difficult for developers to extract the objects in a domain. The existing object-oriented methods do not have the guide of object extraction. Moreover, developers must specify the flexible parts of applications by hot-spot analysis. There is the difference of the abstraction level between domain models and object models of frameworks. Reference models can show the guide of object extraction and mediate between domain models and object models of frameworks. We apply our approach to develop the framework of an industrial system. We propose MVCP (Model-View-Controller-Proxy) model for monitoring domains. By applying this model to this system, we can limit the scope of the objects in relation to hot spots. We discuss the effects in our approach.

キーワード：オブジェクト指向フレームワーク，参照モデル，開発手法

1. はじめに

近年，オブジェクト指向技術は，再利用性や拡張性などのメリットから広く普及し，特に再利用技術としてオブジェクト指向フレームワーク（以下，フレームワークと呼ぶ）が注目されてきた。フレームワークとは，特定の目的のために再利用できるように設計された部品群⁽¹⁾である。フレームワークを適用することにより，生産性，再利用性，品質などの向上が期待できる。現在では，GUI(Graphical User Interface)⁽²⁾⁽³⁾，OS⁽⁴⁾といった汎用的に利用可能なものだけでなく，ドメイン向けの事例^{(5)~(8)}も報告されており，フレームワークの適用が活発に行われてきている。しかし，フレームワークを活用する際には，トレードオフとして考慮しなければならない課題がある。例えば，フレームワーク開発の難しさ，修得にかかるコスト，保守サイクルの継続などが挙げられる。これらの課題を解決しないかぎり，フレームワークの十分な効果を得ることは容易でない。

本稿では，フレームワーク開発の難しさを軽減すること

を目的として，参照モデルという新たな概念を提案し，これに基づくフレームワーク開発手法について示す。フレームワーク開発では，ドメインを分析しオブジェクトモデルを構築した上で，アプリケーションごとで柔軟に対応しなければならない箇所（ホットスポット）を特定し，それをオブジェクトモデルにマッピングする必要がある。参照モデルは，オブジェクト抽出の観点を与え，ホットスポットのオブジェクトモデルへのマッピングをサポートする技術である。また，本アプローチによるシステム開発の事例として，社会インフラで利用する監視システムのフレームワーク開発について紹介する。さらに，開発を通じての適用評価を行い，本アプローチの効果を考察する。

2. フレームワーク技術の現状

フレームワークが適用されているが，一方で課題もある。以下に，効果と課題を簡単に整理する。

〈2・1〉 効果 フレームワークの導入により，次に示すメリットがある。

生産性の向上 フレームワークは、アプリケーションの骨格を提供し、標準・共通機能を実現する。そのため、個々のアプリケーション開発では、個別にカスタマイズしなければならない箇所のみを実現すればよいため、生産性の向上が望める。

再利用性の向上 フレームワークの適用により、再利用される単位はクラス単体ではなく、ソフトウェア・アーキテクチャ（アプリケーションの基本的な構造）を含んだクラス構造となるので再利用範囲が広がる。また、フレームワークによりアーキテクチャを標準化でき、機能拡張が容易である。

品質の向上 フレームワークを何度もアプリケーション開発に適用することで、フレームワークそのものの品質が向上する。それに相関してアプリケーションの品質を向上することが期待できる。

〈2・2〉 課 題 フレームワークの適用により効果が期待できる一方、次に示す課題がある。

開発の難しさ フレームワークを構築するためには、対象となるドメインの特徴や仕様を充分吟味する必要がある。また、完成度の高いフレームワークを構築することは容易でない。一般に、フレームワーク開発では、ドメイン分析の視点からドメインの共通部分を示すフローズンスポットと可変部分を示すホットスポット⁽⁹⁾を特定し、それをオブジェクトモデルにマッピングすることになる。ホットスポットとなる部分に対応するオブジェクトを特定した後に、ホットスポットを満たすフレームワーク設計をサポートするための技術については、デザインパターン⁽¹⁰⁾などを適用したアプローチが試みられている。しかし、ホットスポットをオブジェクトモデルにマッピングする技術については触れていない。

修得にかかるコスト フレームワークを正しく利用しなければ、効果を得ることは難しい。フレームワークの利用方法を修得するための技術として、ドキュメントを提示する方法⁽¹¹⁾、ドキュメントの参照をツール化する方法⁽¹²⁾、ツールによりプログラミングを直接支援する方法⁽⁸⁾などがある。修得コストを最少にする適切な方法を選択する必要がある。

保守サイクルの継続 フレームワークを導入以後も新たな仕様追加などにより修正を繰り返す可能性がある。その際には、フレームワークのインタフェースの互換性を保つことに注意を払う必要がある。しかし、場合によってはインタフェースを変更しなければならない状況が考えられる。その場合には、フレームワークのバージョンの違いをうまく吸収するようなメカニズムを提供する必要がある。

3. 参照モデルに基づくフレームワーク開発手法

〈2・2〉節で指摘したフレームワーク開発の難しさを軽減するために、参照モデルという新しい概念を提案し、これに基づくフレームワーク開発手法を示す。まず、フレーム

図1 フレームワーク構築の視点
Fig. 1. Viewpoints of framework development

ワーク開発で重要視する視点から本手法の位置づけを示す。次に、参照モデルの定義を明らかにし、参照モデルの例として監視ドメイン向けのMVCPモデルについて説明する。次に、参照モデルを導入する理由を示す。さらに、他の再利用技術との関連を示す。最後に、参照モデルに基づくフレームワーク開発プロセスについて述べる。

〈3・1〉 手法の位置づけ フレームワークを開発する際に考慮すべき視点として、「ドメイン分析」、「レイヤ」、「論理」の三つのビューが考えられる。図1にフレームワーク構築の視点を示す。フレームワークは、インフラストラクチャ（OSやミドルウェアを指す。以下インフラと略す。）上に実現され、複数のシステムで利用される再利用部品である。

従来のフレームワーク構築手法⁽⁹⁾⁽¹³⁾は、図1に示す「ドメイン分析」の視点を出発点として、「論理ビュー（静的・動的側面）」を定義する立場と考えられる。すなわち、ドメイン分析の観点からホットスポットを特定し、その上で静的・動的側面を定義する手法である。

一方、我々は図1に示す「レイヤ」の視点を出発点として、再利用性や拡張性だけでなく、実装上の制約を持つシステム要件に柔軟な対応を可能とする手法を既に提案している⁽⁸⁾。レイヤ構造として、インフラ（インフラを隠蔽した機能）、ジェネリック（汎用的に用いる機能）、ドメイン（ドメインに依存した機能）の三つを設けている。

本稿で提案する参照モデルに基づくフレームワーク手法は、ドメイン分析の観点を重視している。したがって、対象とするフレームワークは、GUIフレームワークなどの汎用的に利用可能なフレームワークではなく、ドメイン向けのフレームワークである。ドメイン向けのフレームワークでは、対象システムを多くのユーザーに迅速に提供することを目的としている。

〈3・2〉 参照モデルの定義 我々は、フレームワーク開発において、オブジェクトを抽出しやすくすること、ホットスポットを特定しオブジェクトモデルにマッピングしやすくすることを目的として、参照モデルという新たな概念を提案する。

参照モデルとは、ドメインのソフトウェア構造を端的に表現した論理構成を示す再利用技術である。参照モデルを

適用してアプリケーションを構築することで、一貫性・整合性のある構造を保持することができる。参照モデルは、次の条件を満たすものと定義する。

- システムの静的な側面（全体構造）を実装方式とは非依存で論理的に表現できる。
- システムの動的な側面を参照モデル要素間のシーケンスとして表現できる。
- オブジェクト抽出の観点、分類の観点を与えることができる。
- オブジェクト指向分析・設計フェーズで、Booch 法⁽¹⁵⁾のカテゴリやUML(Unified Modeling Language)⁽¹⁶⁾のステレオタイプとして表現できる。

このような性質を持つ参照モデルは、ドメインごとに整備する。フレームワーク開発でドメイン分析は不可欠である⁽⁹⁾。ドメイン分析は、ドメイン固有の性質を分析・モデル化しておくことで、システム開発を効率的に行うことを目的としている⁽¹⁷⁾。ドメイン分析で得られた情報を有効に活用するためには、参照モデルもドメインごとに定義することが望ましいと考える。

〈3・3〉 参照モデルの例 我々は、対象とするシステムが属するドメインに応じて、さまざまな参照モデルを整備している。例えば、監視ドメイン向けの MVCP(Model-View-Controller-Proxy) モデル、制御ドメイン向けの SMUGD(Service-Media-Unit-GUI-Data) モデル⁽⁸⁾、小規模組み込み制御ドメイン向けのデセセ（デバイス-制御-制約）モデルなどがある。ここでは、MVCP モデルについて説明する。

一般に監視ドメインとは、監視対象機器（例えば、カメラ、ゲート、シャッター、センサなど）に対して遠隔の端末から動作指示やモード指示を与えたり、監視対象機器の状態や監視データ（数値データ、画像データなど）を取得することで、監視対象機器を通じて監視業務を行うドメインである。このような機能を持つ監視ドメインは、次のような特徴を持つ。

- GUI を持っていることが多い。
- 監視対象機器を遠隔制御する、監視対象機器から監視データを受信する機能があり、監視対象機器と協調してサービス（監視業務）を提供する。
- 監視対象機器との通信プロトコル（通信データ構造や通信シーケンス）が、同機能の監視対象機器であっても機種の違いから異なる場合がある。

我々は、MVCP(Model-View-Controller-Proxy) モデル[†]と呼ぶ参照モデルを提案する。MVCP モデルを図 2 に示す。図 2 の矢印は、依存関係（矢印元が矢印先を利用する関係）を示す。以下に、各モデル要素の位置付けを示す。

Model ドメインのデータとそのデータに対する操作を提供するもの。

図 2 MVCP(Model-View-Controller-Proxy) モデル

Fig. 2. MVCP(Model-View-Controller-Proxy) model

View Model の情報を操作者へ視覚的な情報として提供するもの。

Controller Model を管理し View と Model を結び付け、システムのサービスを提供するもの。Proxy と監視データの通信を行う。

Proxy 監視対象機器と監視データの通信を行い、通信プロトコルに依存する部分を隠蔽するもの。

〈3・4〉 参照モデルを導入する理由 参照モデルを導入する理由について、オブジェクト指向分析、ホットスポット分析の二つの観点から述べる。

〈3・4・1〉 オブジェクト指向分析の観点 フレームワーク開発で前提となるのは、オブジェクトモデルが充分整備されていることである。しかし、従来のオブジェクト指向方法論⁽¹⁴⁾⁽¹⁵⁾では、オブジェクト指向分析と設計の開発フェーズに関する記述が中心になっている。実際の開発では、ドメイン分析やシステム分析といったシステムレベルでの視点でソフトウェアの静的側面や動的側面を検討するプロセスも重要である。従来のオブジェクト指向方法論では、オブジェクトを抽出することから分析を始めることとしている。しかし、オブジェクト抽出に関する指針は充分示されていない。そのため、分析者によってオブジェクト抽出の粒度にばらつきがあったり、オブジェクトの役割が異なっていたりすることが考えられる。

我々は、この課題を解決するために参照モデルを導入する。参照モデルは、システムレベルの論理構成を表現しており、モデル要素はオブジェクト抽出の観点を表現している。参照モデルのモデル要素を使ってシーケンスを記述することで、システムの動的な側面について、基本的な流れを把握することができる。分析者に対して、オブジェクト抽出の指針を与えることができ、一貫性・整合性のある構造を構築できる。

〈3・4・2〉 ホットスポット分析の観点 フレームワークを構築するためには、ドメイン分析の視点からホットスポットを特定し、これをスムーズにフレームワーク設計に反映することが重要である。しかし、ホットスポットをオブジェクトにマッピングすることは容易でない。複数のホットスポットが一つのオブジェクトに対応する場合や、一つのホットスポットが複数のオブジェクトに対応する場合が考えられる。図 3 では、1 番のホットスポット（サービスにより扱う情報が異なる）と、2 番のホットスポット（エラー情報

[†]Smalltalk-80 の MVC(Model-View-Controller) アーキテクチャとは異なる概念である。

図 3 従来のフレームワーク開発

Fig. 3. Traditional framework development

が異なる)が, CMonitoringSetData オブジェクトに関係しており, 複数のホットスポットが一つのオブジェクトに対応する例を表している。

また, 図 3では五つのオブジェクトしかないが, 一般にオブジェクト数は数十～数百個程度になることが多く, ホットスポットを多数のオブジェクトを対象にマッピングすることを強いられていた。これらは, ホットスポットがシステム要求仕様の視点に立っているのに対し, オブジェクトがソフトウェア内部構造の視点に立っていることに起因する。両者にギャップがあるために, ホットスポットをオブジェクトにマッピングすることは容易でない。従来のフレームワーク開発では, この点に関する議論が充分でなかった。

我々は, 上記の課題を解決するために参照モデルを導入する。参照モデルは, システムレベルのドメインモデルと, オブジェクトレベルのオブジェクトモデルのギャップを埋め, 両者を接続することができる。そのため, ホットスポットを比較的容易に参照モデルのモデル要素にマッピングすることができる。ここでマッピングしやすいとは, ホットスポットに対応するオブジェクト候補の範囲が狭い(オブジェクト候補数が少ない)ことを表している。図 4に示すように, 参照モデルに基づくフレームワーク開発では, 対象とするアプリケーションにおいて登場するモデル要素の数は多くても数個程度(それに対してオブジェクトの場合には数十～数百個程度)であり, 比較的容易にマッピングできる。さらに, 参照モデルからオブジェクトモデルにマッピングする際には, ホットスポットに対応する参照モデルのモデル要素が限定され, 対象となるモデル要素内部のオブジェクト群のみがマッピング候補となる。図 4では, 1番と2番のホットスポットが MVCP モデルの Model にマッピングされていることから, 一つのオブジェクトに限定されることになる(図 4では, 参照モデルの要素名をクラスのステレオタイプとして表現)。図 4では, 非常に簡単例を示しているが, 実際のシステム開発で多くのオブジェクトが存在した場合には, 参照モデルを導入することによりマッピング作業がさらに容易になる。

〈3・5〉 他の再利用技術との関連 本稿で提案する参照モデルという概念は, 再利用技術である。参照モデル以外の再利用技術には, ドメインモデル, パターン(アナリシスパターン, デザインパターンなど), フレームワークな

図 4 参照モデルに基づくフレームワーク開発

Fig. 4. Framework development based on reference models

図 5 参照モデルと他の再利用技術との関連

Fig. 5. Relationships between reference models and others reuse technique

どがある。これらの技術と参照モデルについて, 開発工程と抽象度を軸に配置したものが図 5である。ドメインモデルは, ドメインの知識をモデル化したものであり, システムレベルの再利用技術である。参照モデルは, システムレベルの再利用技術であり, オブジェクト指向分析前のシステム分析の段階で活用される。一方, オブジェクトレベルの再利用技術には, パターン, フレームワークなどがある。パターンは, 分析あるいは設計段階で利用され, 粒度は中程度で実装を持たない。フレームワークは, 主に設計・実装段階で使われ, 比較的粒度が大きく実装を持つ。これらの再利用技術に対し, 参照モデルは主にシステム分析段階で利用されるもので, 粒度が比較的大きくそれ自体は実装を持たない。

また, 参照モデルとフレームワークの関係を図 6に示す。参照モデルは, 後で再利用や拡張できるようにカタログ化しておく。ドメインやシステムの特徴に合わせて複数の参照モデルが存在する(図 6の参照モデル A～Cに相当)。さらに, 参照モデルはフレームワークを実現するメタモデルとして位置づけられる。例えば, 図 6の「参照モデル B」をフレームワークとして実現したのが「フレームワーク B」に相当する。さらに, 「フレームワーク B」を活用して, 「アプリケーション B1～B3」を開発する。

図 6 参照モデルとフレームワークの関係

Fig.6. Relationships between reference models and frameworks

図 7 参照モデルに基づくフレームワーク開発プロセス

Fig.7. Framework development processes based on reference models

〈3・6〉 フレームワーク開発プロセス 図 7に参照モデルに基づくフレームワーク開発プロセスを示し、以下に概要を示す。

Process1：ドメイン分析 ドメインの知識や情報を整理し、開発対象となるアプリケーションの要求や制約を考慮した上で、ドメインモデルを構築する。

Process2：参照モデル構築 既存の参照モデルのカタログを参照し、システム要求をドメインモデルから参照モデルにマッピングし整理する。必要ならば新しいモデル要素を加えて、既存の参照モデルを拡張・改良する。

Process3：オブジェクト指向分析 参照モデルのモデル要素は、オブジェクト抽出の観点を示す。モデル要素にしたがってオブジェクトを抽出する。

Process4：ホットスポット分析 ドメインの中で、仕様変更や予測できないことに柔軟に対応しなければならないホットスポットを特定する。

Process5：ホットスポットマッピング 特定したホットスポットが参照モデルのどのモデル要素に影響を与えるかをマッピングする。

Process6：フレームワーク設計 ホットスポットをマッピングした参照モデルを基に、フレームワークの構造を静的モデル（クラス図）として構築する。さらに、フレームワークの振る舞いを動的モデル（シーケンス図）として定義する。

Process7：フレームワーク実装 オブジェクト指向言語を用いてフレームワークを実装する。

上記の各プロセスのうち、本手法で特に重要である Process1:ドメイン分析, Process2:参照モデル構築, Process6:フレームワーク設計の三つのプロセスの詳細を以下に示す。

〈3・6・1〉 ドメイン分析 ドメインとは、複数の類似した問題やアプリケーションが属する対象領域のことを示す⁽¹⁷⁾。一つのドメインには複数のアプリケーションが存在する。例えば、監視ドメインには、高速道路監視アプリケーション、ビル監視アプリケーションなどが存在する。ドメイン分析では、ドメインのあらゆる情報を列挙して用語を整理し、ドメインの開発制約を明らかにする。ドメインの

情報とは、業務知識、業務ロジックなどドメインを理解する上で必要な情報である。さらに、これらの情報を理解する上で必要な用語を整理する。一方、インフラや性能条件など実現する上で制約となる点を洗い出す。また、システム要件をピックアップし優先順位を付けておく。

厳密に言えばドメインの性質により重要視する点は異なるが、ここでは大きく事務処理システムなどの情報分野と、監視システムや制御システムなどの工業分野とに分けて、ドメイン分析のポイントをまとめる。

情報分野 ドメインとしては、比較的成熟している場合が多い。システムに関わるアクター（システムに影響を与えるシステム外部の人間または事物）が多い場合がある。特に、アクターは人間の場合が多い。主に事務処理業務を実現しているので、画面や帳票類が非常に多い。このような特徴を持つ情報分野では、画面や帳票類に表れるデータと、それを操作する業務ロジックとの関連を明らかにすることが重要である。情報分野に属するドメインとしては、会計、金融、販売、運輸などが挙げられる。

工業分野 ドメインとしては、比較的未成熟な場合が多い。システムに関わるアクターは人間よりも事物である場合が多い。画面を持つことが少なくないが、自律的に稼働する機器を監視・制御することに重点を置いている。したがって、システムの性能制約などの開発制約が厳しい場合が多い。また、新しい種類の機器が接続されたりなど、ドメインが将来どのように変化していくのかなどの予測が困難な場合が多い。このような特徴を持つ工業分野では、リアルタイム性や再利用性といったトレードオフとなるシステム要件を満足することが重要である。工業分野に属するドメインとしては、大きくは制御ドメイン（プラント制御、組み込み制御など）と、監視ドメイン（電力監視、機器監視など）が存在する。

〈3・6・2〉 参照モデル構築 ドメイン分析で得られた業務知識や業務ロジックに関連する情報からドメインをどのような論理構成でとらえればよいかを参照モデルとして表

す。主に次のような点に注意して参照モデルを構築または選定する。

- 仕様変更に対してどの部分で対応するか。
- アクターに依存する部分をどこで隠蔽するか。
- インフラの変更に対してどのように対応するか。
- 業務ロジックの流れと、データの流れをどのように表現するか。

参照モデルは、開発制約やシステム開発で活用する方法論（構造化分析・設計、オブジェクト指向、コンポーネント指向など）に依存するものではない。ドメインの状況が変わらない限り、固定的なものである。

参照モデルはドメインごとにカタログ化しておき、後で再利用や拡張できるようにしておく。参照モデルを構築する際には、カタログ化された既存の参照モデルを参考にしたり、必要があれば新たにモデル要素を追加したりする。図8では、監視ドメイン向けのMVCPモデルを特定の監視アプリケーションで再利用するレイヤ構造を持つ参照モデルを示している。各レイヤの位置づけを以下に示す。

ドメインレイヤ 監視、制御ドメインなど、特定のドメインに適用可能なレイヤ。対象となるドメインの複数のアプリケーションに適用する。例えば、図8では、監視ドメイン向けのMVCPモデルがこれに相当する。

アプリレイヤ 特定のアプリケーションに適用可能なレイヤ。ドメインレイヤの参照モデルを再利用・拡張する場合がある。例えば、図8ではドメインレイヤのMVCPモデルを再利用し、監視アプリケーションA向けの参照モデルがこれに相当する。ドメインレイヤでModel、View、Controller、Proxyと呼んでいたモデル要素を、対象とするアプリケーションに合ったData、GUI、Manager、I/Oという名称に置き直している。

〈3・6・3〉フレームワーク設計 参照モデルのモデル要素ごとにホットスポットに対応するクラス（オブジェクト）を特定する。ホットスポットに該当しないクラスは、そのまま再利用するクラスとして実現する。ホットスポットに該当するクラスは、カスタマイズが必要なクラスとして実現する。フレームワーク設計では、ホットスポット分析を反映するだけでなく、〈3・6・1〉節で示したドメイン分析の結果を反映し、フレームワークにどのような仕様を盛り込むかということが重要である。

ホットスポットは、アプリケーションごとに異なる仕様（ユーザの要求仕様）を表している。要求仕様はユーザによってばらつきがある場合があり、豊富な仕様を要求するユーザもあれば、最低限の仕様を満足すれば良いというユーザも存在する。フレームワークでは、このように要求仕様に違いのあるすべてのユーザに柔軟な対応ができることが望ましい。

例えば、すべてのユーザに共通な仕様のみをフレームワークに組み込んだとすると、フレームワークで対応できる範囲が狭くなり大きな効果を得ることは困難である。また、仕様変更が頻繁な場合には、当初ユーザ共通であった仕様が

図8 レイヤ構造を持つ参照モデル
Fig. 8. Layered reference models

将来共通でなくなる可能性が考えられる。このようなことからフレームワークに実現する仕様は、ユーザ共通であるかという観点ではなく、対象とするドメインにおいて標準的かどうかという観点で定義することが望ましいと考える。標準仕様から外れる仕様については、オプション仕様という形で区別する。個々のユーザから要求されたオプション仕様の中で、特に多くのユーザで要求されているオプション仕様については、フレームワークで実現した方が再利用の観点で効果的と考える。

このような背景から、フレームワークには「標準仕様」、「オプション仕様」の両方を実現する。フレームワークで実装すべき「標準仕様」、「オプション仕様」とも、設計上はフレームワークの基底クラスのメソッドとして定義するため、フレームワーク設計イメージを示すと図9のようになる。それぞれの部位の説明を以下に示す。

フレームワークのフローズンスロット フレームワークのフローズンスロットでは、アプリケーションで固定的な標準仕様とオプション仕様とが含まれる。オプション仕様は、アプリケーションによって利用しない場合もある。これらのクラスは、図9に示すように継承を前提としない。フレームワークのホットスポット フレームワークのホットスポットでは、アプリケーションで流動的な標準仕様とオプション仕様とが含まれる。オプション仕様は、アプリケーションによって利用しない場合もある。これらのクラスは、図9に示すように継承を前提とし基底クラスとする。

アプリケーション アプリケーションでは、フレームワークのホットスポットに相当する箇所をカスタマイズする。これらのクラスは、図9に示すようにフレームワークのクラスを継承し派生クラスとする。

4. 監視システム開発への適用

3章で示したアプローチを適用した監視システムでのフ

図 9 フレームワークアーキテクチャのイメージ
Fig. 9. An image of the framework architecture

フレームワーク開発事例について示す。現在、フレームワークを実システム開発で適用中の段階である。

〈4・1〉対象システム 工業分野の社会インフラとして活用する監視システムであり、図 10 に示すように、複数の監視対象機器と上位機器に対して専用 LAN で接続されている。グループごとに監視システムが設置されており、監視業務として提供するサービスは 50～60 種類である。また、ソースコードのサイズは約 10 万ステップであり、我々のさまざまなドメインのシステム開発の中では中規模なシステムである。

監視業務において、監視対象機器との間で送受されるデータは、数値化されたバイナリデータである。例えば、監視対象機器の状態は、これを数値化したデータを監視対象機器から受信することで判断する。また、監視対象機器にて収集したデータは、監視対象機器を通じて顧客がサービスを受けた履歴に相当する情報を表しており、これも数値化されバイナリデータとして監視対象機器から送信される。また、一度に送信されるデータサイズは数百 K 程度であり小規模であるが、数秒から数十秒の間隔で送信される通信頻度の高いデータである。以下に、監視業務の分類を列举する。

- 監視対象機器の状態（正しく動作しているかなど）を表示し、必要に応じてモード設定を行う。
- 監視対象機器にて採取したデータを収集する。
- 収集したデータを直接あるいは中間集計した後に上位機器に転送する。
- 監視対象機器に必要なパラメータやデータを上位機器から受信し、監視対象機器に配信する。
- 監視対象機器と上位機器の通信状態を表示する。

〈4・2〉フレームワーク開発プロセス 本手法に基づき〈3・6〉節で示した開発プロセスを実践した。ただし、フレームワークを適用したシステム開発に先駆けて、オブジェクト指向を導入したシステム開発を数回実施し、ドメインのオブ

図 10 監視システム
Fig. 10. Monitoring systems

ジェクトモデルをフィックスした。したがって、Process1～Process3（ドメイン分析～オブジェクト指向分析）については、これまでのシステム開発の結果を流用した。

〈4・2・1〉ドメイン分析 ドメインの把握と、システム要件の整理という二つの観点で分析を進めた。まず、ドメインの理解について述べる。ドメインとしては未成熟であり、新しい監視対象機器が設置されたり、大きな単位でのサービスが追加されたりする可能性が高い。仕様変更は頻繁に行われ、随時アプリケーションをカスタマイズする必要がある。次に、特に重要なシステム要件を示す。

再利用性 アプリケーションにより仕様が少しずつ異なり、仕様の追加や変更が頻繁に行われている。アプリケーションごとに別々に対応すると開発コストが増大するので、共通に再利用可能な部品を提供することが望まれる。

拡張性 仕様変更については、新しいサービスや運用ルールが増えるなど、新仕様の予測が困難な状況にある。また、複数アプリケーションで同時に新仕様を追加することが多い。したがって、ドメインとして成熟しておらず、どのような仕様追加や変更に対しても、柔軟で迅速に拡張することが求められる。

上記のように再利用性や拡張性を最も重視し、非常に多数のアプリケーションを開発することが必要であり、開発リードタイムの短縮が望まれる。

〈4・2・2〉参照モデル構築 本適用事例で対象とするシステムは、監視ドメインのシステムであり、〈3・3〉節で示した MVCP モデルを活用した。特に、次の理由から MVCP モデルを採用した。

- 監視対象機器と協調してサービスを提供する。
- 仕様変更がサービス内容に集中している。
- 監視対象機器との通信プロトコルが、新型機器と旧型機器とで異なる。

〈4・2・3〉オブジェクト指向分析 システム開発の戦略として、従来技術、オブジェクト指向技術、コンポーネント

技術の三つの技術を混在させて進めた。オブジェクト指向技術やコンポーネント技術は、非常に有効な技術であるが、導入コスト（開発者への教育やスキルアップにかかるコスト）を考慮し、これらの新規技術を最も効果的な箇所に適用することで、少しでも導入コストを抑えようと考えた。

ドメイン分析の結果、GUI 画面と業務ロジックの部分に仕様変更が集中しており、このことから Controller（業務ロジック）にはオブジェクト指向技術を、View（GUI 画面）にはコンポーネント技術を適用した。GUI にコンポーネント技術を適用することは、さまざまな GUI 開発環境に代表されるように一般的と考えられる。残りの Model と Proxy については、オブジェクト指向技術やコンポーネント技術の導入コストと比べて大きな効果が期待できないと判断し、従来技術を適用した。

〈4・2・4〉ホットスポット分析 既にオブジェクト指向技術を導入した製品開発（フレームワーク未適用）を数回実施しており、その経験からフレームワークとして実現すべき仕様、アプリケーションごとに異なる仕様の吟味を行った。その結果、主に考慮すべき点として次に示す項目が挙げた（実際にはより詳細な吟味を行った）。

- アプリケーションによって異なる部分とは、GUI 仕様、システムで提供するサービス、およびシステム内部で扱う情報の三種類に分類できる。
- GUI については、例えば扱うデータ種別の追加などに伴いボタンを追加する場合がある。
- サービスについては、例えば監視システムの上位機器が接続されるかどうかで提供するサービスが異なる場合がある。
- 扱うデータについては、例えばエラー情報の項目が異なる場合がある。

〈4・2・5〉ホットスポットマッピング ホットスポット分析の結果を、MVCP モデルのモデル要素にマッピングしたところ、Controller にホットスポットが集中していることがわかった。一方、View と Model は、多くはないがホットスポットが存在している。また、Proxy は、ほとんどホットスポットがなかった。

〈4・2・6〉フレームワーク設計 本システムでは、Controller にのみオブジェクト指向技術を適用するため、この部分をフレームワークの設計対象とした。また、従来技術において Model は共有ファイルや共有メモリなどのようなプロセスの境界に存在する。このことから、Controller のオブジェクト（監視システムの業務サービスを実現するオブジェクト）から Model にアクセスするためのオブジェクトが必要であることが判明した。

参照モデルへのマッピング結果から、ホットスポットに対応するオブジェクトを整理した。ここでは、Controller に登場するオブジェクトをマネージャオブジェクトとデータアクセッサオブジェクトに分ける。マネージャオブジェクトとは、監視業務サービスを実装するオブジェクトである。一方、データアクセッサオブジェクトは、Model にア

クセスするためのオブジェクトである。データアクセッサオブジェクトは、マネージャオブジェクトに対して、システムで扱うデータにアクセスするためのインタフェースを提供するものである。以下に、ホットスポットのパターンを示す。

業務ロジックが異なるパターン 例えば、監視対象機器に対してパラメータ設定を行うオブジェクトは、設定が不正の際に警報を鳴らす場合がある。

データの種別が異なるパターン 実際にはあまり多くないケースである。例えば、日報を作成しそれをジャーナル印刷する際に、印字フォーマットが変化する場合がある。印字フォーマットに関する仕様は、データアクセッサオブジェクト内部で閉じており、マネージャオブジェクトに影響を与えない。

データの種別と業務ロジックが異なるパターン 例えば、日報を作成する際に、対象機器から収集するデータ種別が異なるため、収集シーケンスが変化する場合がある。データ種別の変化はデータアクセッサオブジェクトに影響し、収集シーケンスの変化はマネージャオブジェクトに影響する。

これらの分類ごとにフレームワーク設計を進めた。例えば、あるホットスポットがマネージャオブジェクトとデータアクセッサオブジェクトとに関係する場合には、デザインパターン⁽¹⁰⁾で示されている Template Method パターンを活用して、図 11 に示すようなクラス構造を設計した。

図 11 に示すフレームワークの CBaseMonitoringManager オブジェクトは、フレームワークの CBaseDataAccessor オブジェクトと所有関係にある。CBaseMonitoringManager オブジェクトには、TemplateMethod というメソッドが定義されており、その内部で PrivateOperation1 メソッドと PrivateOperation2 メソッドを呼び出しており、アプリケーションの CMonitoringManager オブジェクトでそれぞれ再定義する。一方、TemplateMethod というメソッドでは、CBaseDataAccessor オブジェクトの GetData メソッドと SetData メソッドを呼び出している。これらのメソッドは、アプリケーションの CDataAccessor オブジェクトでそれぞれ再定義する。このように、マネージャオブジェクトで業務ロジックをカスタマイズするとともに、データアクセッサオブジェクトでデータ種別などをカスタマイズすることができる。

5. 適用評価

本手法を用いたフレームワーク適用によるプロトタイプ開発の結果からシステム要件について評価する。また、フレームワーク開発の容易さという観点で、定性的な評価を示す。現在、開発したフレームワークを適用したシステム開発を進めている。

〈5・1〉システム要件 本システムで重要視しているシステム要件は、「再利用率」と「拡張性」である。これらを判断する一つの方法として、開発工数について本手法に

図 11 フレームワーク設計の例
Fig. 11. An example of framework designs

よるフレームワーク適用による開発と未適用の開発についてプロトタイプ開発を通じて評価した。プロトタイプ開発では、典型的な仕様変更を想定し開発工数を比較した。フレームワークの初回適用では、フレームワーク自体の開発コストがかかるため、このコストも考慮した上でフレームワーク適用の効果の判断を必要がある。以上のことから、フレームワーク適用による開発コスト削減を確認するとともに、次の評価式によりフレームワーク開発コストが何回のリピートで回収できるかを判定した。

$$DfwDevCost < n(ObjDevCost - DfwDevCost)$$

ここで、DfwDevCost はフレームワーク開発工数、n はリピート回数、ObjDevCost はフレームワーク未適用の開発工数、DfwDevCost はフレームワーク適用の開発工数を表す。

その結果、フレームワーク未適用の場合と比べてフレームワークを適用した方が約 90% 程度の開発工数で済むことが判明した。また、フレームワーク開発コストは、5~6 リピート適用すれば回収できる見通しがたった。本システムでは、数十リピートを見込んでおり、本手法によるフレームワーク適用が効果的であると考えられる。

〈5・2〉 フレームワーク開発の容易さ

〈5・2・1〉 開発対象の明確化 従来のフレームワーク開発では、フレームワークを利用したアプリケーションの論理構成を明確に表現する手段がなかった。そのため、物理構成（プロセス構成）にマッピングして、フレームワーク開発対象を明確化していた。本来、論理構成と物理構成は明確に区別するべきである。

一方、参照モデルに基づいたフレームワーク開発では、参照モデルにより論理構成を表現する。参照モデルのモデル要素をベースに物理構成を検討することで開発対象を明確に表現することができる。

〈5・2・2〉 論理構成要素間の関連の明確化 従来のフレームワーク開発では、アプリケーション全体をオブジェクト指向で分析してフレームワーク適用対象とすることが前提となっている。しかし、部分的にオブジェクト指向を導入しフレームワークを開発する場合には、フレームワークを適用する論理構成と他の論理構成の関係を明確に表現する手段がなかった。

一方、参照モデルに基づいたフレームワーク開発では、参照モデルがアプリケーション全体の論理構成を示しており、アプリケーション全体をオブジェクト指向で分析することに限定しない。実際、適用事例で示した監視システムの開発では、システム開発の戦略として、従来技術、コンポーネント技術、オブジェクト指向技術の三つの技術を混在させている。このような場合にでも比較的容易に対処することが可能となる。

〈5・2・3〉 オブジェクト抽出のガイド フレームワークを開発する以前に、オブジェクト指向分析においてオブジェクトモデルがフィックスしていることが前提である。したがって、オブジェクト抽出の観点からは、非常に重要な意味を持つてくる。

従来のオブジェクト指向開発手法では、要求仕様からオブジェクトを抽出する作業から始める。フレームワーク開発でも同様の作業を行う。しかし、オブジェクトを抽出する明確な指針は提示されていない。

一方、参照モデルに基づく開発では、モデル要素そのものがオブジェクトを抽出する指針を表現する。モデル要素の役割を理解していれば、比較的容易にオブジェクトを抽出することができる。

〈5・2・4〉 ホットスポット分析からのスムーズな移行 フレームワーク開発では、ドメイン分析の視点からフローズンスポットとホットスポットを特定する作業が必要である。ホットスポット分析の結果を吟味し、最終的にソフトウェア構造として表現する。

従来のフレームワーク開発では、ホットスポットとなる部分を分析しオブジェクトモデルにマッピングすることになる。アプリケーションにおいて、オブジェクトの数は数十~数百個程度になることが多く、非常に多くのオブジェクトを対象にホットスポットからマッピングすることを強いられていた。これは、ホットスポットがシステム要求仕様の視点に立っているのに対し、オブジェクトがソフトウェア内部構造の視点に立っていることに起因する。そのため、ホットスポットをオブジェクトにマッピングすることは容易でない。

一方、参照モデルを導入したフレームワーク開発では、対象とするアプリケーションにおいて登場するモデル要素の数は多くても 4 個（それに対してオブジェクトの場合には約 250 個）であり、比較的容易にマッピングできる。

また、参照モデルからオブジェクトモデルへホットスポットをマッピングする際には、参照モデルの個々のモデル要素内部に存在するオブジェクトのみを対象とすればよい。

表 1 Controller 内部のオブジェクト数
Table 1. The number of objects in Controller

オブジェクトの分類	オブジェクト数
マネージャ	約 25
データアクセッサ	約 85
メールデータ	約 130
その他	約 10
合計	約 250

例えば、ユーザにより収集するデータの種別が異なり、これに応じた業務ロジックを変更しなければならないというホットスポットを想定する。データ種別の変更については、MVCP モデルの Model に影響を与えるとともに、Model にアクセスする Controller 内部のデータアクセッサオブジェクトに影響を与える。一方、業務ロジックの変更については、Controller 内部のマネージャオブジェクトに影響を与える。以上のホットスポット分析に対して、表 1 の Controller 内部のオブジェクト数をもとに、ホットスポットに対応するオブジェクト候補がどの程度限定されたかを評価する。業務ロジックの変更については、Controller 内部のマネージャオブジェクト（約 25 個）を対象とすればよい。さらに、Controller のマネージャオブジェクトは業務ロジックごとに抽出することが明らかなので、収集サービスに関連するオブジェクトを（約 10 個）のみを対象とすればよい。

以上のことから、フレームワークの設計を比較的容易に進めることができた。

6. 関連研究

フレームワーク開発手法の観点から、本アプローチと他の関連研究を比較する。既存の手法として、ホットスポット駆動手法⁽⁹⁾、データ中心アプローチとユースケースに基づく手法⁽¹³⁾がある。

ホットスポット駆動手法では、既存のオブジェクト指向方法論を拡張し、ドメイン分析の観点からドメインの共通部分を示すフローズンスポットと可変部分を示すホットスポットを特定し、ホットスポットの柔軟性に基づいてメタパターンを選択している。この手法は、ホットスポットを満たすフレームワーク設計に関する議論に焦点を当てている。ドメイン分析の観点であるホットスポットを、ソフトウェア内部構造の観点であるオブジェクトモデルにどのようにマッピングするかについては着目していない。我々の提案するアプローチでは、参照モデルを導入し、ホットスポットをオブジェクトモデルにマッピングする手順を示している。ホットスポット駆動手法と我々の手法は、着目している点が異なるが相互に補完可能な技術である。

また、データ中心アプローチとユースケースに基づく手法では、システム群の分析とドメインの静的な特徴を抽出するためにデータ中心アプローチを取り入れ、動的な特徴を抽出し静的な特徴と合わせて一般化するためにユースケースを導入し、詳細なフレームワーク開発プロセスを示している。この手法は、フレームワーク設計のためのドメイン

分析を支援する技術という点で、我々と同じ立場を採っている。ただし、特に有効としているドメインが異なる。データ中心アプローチとユースケースに基づく手法は、事務処理系のデータベースへの処理が中心となるドメインに対して特に有効であるとしている。我々の提案するアプローチは、ドメイン分析によりドメインの特徴を参照モデルとして表現可能なドメインに対して特に有効である。データ中心アプローチとユースケースに基づく手法と我々の手法については、ドメインやシステムの特徴に合わせて選択することが適切であると考ええる。

7. おわりに

本稿では、参照モデルに基づくフレームワーク開発手法について述べた。また、本アプローチを監視システムへ適用し、効果を考察した。

フレームワーク開発の視点として、「レイヤ」、「ドメイン分析」、「論理」という三つのビューを指摘した。本アプローチは、ドメイン向けのフレームワークを主なターゲットとして、「ドメイン分析」を重視した手法である。ドメイン向けのフレームワーク開発において、オブジェクトの抽出を容易にすること、ホットスポットのオブジェクトへのマッピングを容易にすることに着目した。本アプローチでの参照モデルの導入により、これらの作業を容易にすることを監視システムへの適用事例を通じて評価した。

今後は、次の四項目について検討することが課題である。第一に、我々の手法を採用した効果を製品開発において定量的に評価することである。同一のフレームワークを別々の手法で開発することは、比較条件という意味でもシステム開発戦略上も考えにくいため、現時点では開発者に対してインタビューすることを考えている。現在、詳細なインタビュー項目を検討中である。第二に、我々の手法で核となる参照モデルをより充実させることである。現在、いくつかのドメイン向けの参照モデルを整備しているが、より体系的に参照モデルを充実させる必要がある。現在は、監視制御システムなどの工業分野を中心に適用を進めている。第三に、「論理ビュー」という視点について検討することである。この側面においては、メタパターンの概念が有効と考えている。既に提案されているメタパターンだけでは不充分であると考えており、開発者の経験や試行錯誤によって得られたノウハウをメタパターンとして抽出することを検討中である。第四に、システムあるいはドメインに応じてどの視点を重視するかなどの指針が必要であり、これら三つの視点を考慮した手法を検討することである。

（平成 12 年 4 月 25 日受付、同 12 年 10 月 27 日再受付）

文 献

(1) Rogers, G. F.: Framework-Based Software Development in C++, Prentice-Hall(1997).

- (2) Schmucker, K. J.: Object-Oriented Programming for the Macintosh, Hayden Book Company(1986).
- (3) Linton, M. A., Calder, P. R. and Vlissides, J. M.: InterViews: A C++ Graphical Interface Toolkit, Technical Report CSL-TR-88-358, Stanford University(1988).
- (4) Lewis, T. et al.: Object-Oriented Application Framework, Manning Publications(1993).
- (5) Franek, B., Gaspar, C.: SMI++ Object Oriented Framework for Designing and Implementing Distributed Control Systems, IEEE Transaction on Nuclear Science, Vol.45, No.4, pp.1946-1950(1998).
- (6) Alfredeisen, K., Saether, B.: An Object-Oriented Framework for Water Resource Planning, Proceedings of the Thirty-First Hawaii International Conference on System Science (Cat. No.98 TB100216), Vol.7, pp.441-450(1998).
- (7) 荒野高志, 青野博, 藤崎智宏: ネットワーク管理フレームワークとその開発に関する一考察, 情報処理学会論文誌, Vol.38, No.6, pp.1182-1191(1997).
- (8) 早瀬健夫: オブジェクト指向フレームワークと製品特化 CASE による組み込みシステム開発, 電気学会論文誌 C (電子・情報・システム部門誌), Vol.199-C, No.12, pp.1573-1581(1999).
- (9) Pree, W.: Design Patterns for Object-Oriented Software Development, Addison-Wesley, 1994 (佐藤他訳: デザインパターンプログラミング, トッパン, 1996).
- (10) Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns, Addison-Wesley(1994) (本位田真一, 吉田和樹 監訳: デザインパターン, ソフトバンク, 1995).
- (11) Johnson R. E.: Documenting frameworks using patterns, In Proceedings of OOPSLA '92, 1992.
- (12) e Silva, R. P., Price, R. T.: Tool support for helping the use of frameworks, Proceedings SCCC'98. 18th International Conference of the Chilean Society of Computer Science, pp.192-201, 1998.
- (13) 名取万里, 加賀谷聡, 本位田真一: データ中心アプローチとユースケースに基づくオブジェクト指向フレームワーク構築手法, 情報処理学会論文誌, Vol.38, No.3, pp.634-656(1997).
- (14) Rumbaugh, James et al.: Object-Oriented Modeling & Design, Prentice Hall(1991) (羽生田栄一 監訳: オブジェクト指向方法論 OMT, トッパン, 1992).
- (15) Booch, Grady: Object-Oriented Analysis and Design with Applications. Second Edition, Benjamin Cummings Publishing Company(1994) (山城明宏 他訳: Booch 法: オブジェクト指向分析と設計 第2版, アジソンウェスレイ, 1995).
- (16) Booch, Grady et al.: The Unified Modeling Language User Guide, Addison-Wesley(1998) (羽生田栄一 監訳, オージス総研 オブジェクト技術ソリューション事業部 訳, ピアソン・エデュケーション, 1999).
- (17) 伊藤潔 他著: ドメイン分析・モデリング これからのソフトウェア開発・再利用 基幹技術, 共立出版(1996).

早瀬 健夫 (正員) 1967 年生。1990 年 3 月上智大学理工学部機械工学科卒。1992 年 3 月同大学大学院理工学研究科機械工学専攻修士課程修了。同年 4 月 (株) 東芝入社。現在, 同社情報・社会システム社 SI 技術開発センターに所属。オブジェクト指向技術の研究・開発に従事。ACM, IEEE, 情報処理学会, 人工知能学会各会員。

池田 信之 (非会員) 1967 年生。1990 年 3 月筑波大学第三学群情報工学類卒。1992 年 3 月同大学大学院理工学研究科理工学専攻修士課程修了。同年 4 月 (株) 東芝入社。現在, 同社情報・社会システム社 SI 技術開発センターに所属。オブジェクト指向技術の研究・開発に従事。ACM, IEEE, 情報処理学会各会員。

松本 一教 (非会員) 1961 年生。1984 年 3 月九州大学理学部数学科卒。1986 年 3 月同大学大学院総合理工学研究科修士課程修了。同年 4 月 (株) 東芝入社。現在, 同社情報・社会システム社 SI 技術開発センターに所属。博士 (理学)。データマイニングなどの研究・開発に従事。ACM, IEEE, 情報処理学会, 人工知能学会各会員。

守安 隆 (非会員) 1954 年生。1978 年 3 月京都大学卒。1982 年 3 月同大学大学院博士課程修了。1983 年 4 月 (株) 東芝入社。現在, 同社情報・社会システム社 SI 技術開発センターに所属。分散オブジェクト技術などの研究・開発に従事。IEEE, 情報処理学会各会員。