3ビュー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

早瀬健夫[†] 松本一教[††]

本稿では，オブジェクト指向フレームワーク開発のための3ビューモデル，および3ビューモデルを用いたフレームワーク開発プロセスを提案する．本手法では，対象システムを多くのユーザへ提供することが求められ高いリピート性を持つシステム開発向けのフレームワークを対象とする．このようなフレームワークに特に求められる品質特性は，再利用性，移植性，保守性である．3ビューモデルは，これらの品質特性を向上するために，ドメイン分析ビュー，レイヤビュー，メカニズムビューから構成される．ドメイン分析ビューでは，同種のシステム群全体の性質や知識をドメイン参照モデルと呼ぶ基本的なモデルで明確にし，これに基づきフレームワークを開発するので，再利用性を向上することが可能となる．レイヤビューでは，3つのレイヤ（インフラレイヤ，ジェネリックレイヤ，ドメインレイヤ）に分割することで，実装上の制約を考慮しフレームワークの一部を交換することが容易であるので，移植性を向上することが可能となる．メカニズムビューでは，開発が容易なホワイトボックスフレームワーク，あるいは利用が容易なブラックボックスフレームワークのどちらのメカニズムを選択するかの判断基準を提示しているので，システムの特徴にあわせて保守性を向上することが可能となる．また，本手法を監視システム向けのフレームワーク開発に適用した事例について示し，本手法の有効性を示す．

Object-oriented Framework Development Method Based on a Three-view Model

Takeo Hayase[†] and Kazunori Matsumoto[††]

This paper describes a three-view model for developing object-oriented frameworks and processes for developing frameworks based on this model. Our approach focuses on frameworks for developing systems that are released for many users and has high repeatability. The quality factors that are especially required for developing such a framework are reusability, portability, and maintainability. The three-view model consists of a domain analysis view, a layer view and a mechanism view. The domain analysis view is used to clarify all information and domain knowledge by using a basic model that we call it domain reference model, so that a framework has high reusability. The layer view is used to divide a framework into three layers that are piled up vertically: an infrastructure layer, a generic layer, and a domain layer. Because software engineers can replace a part of the framework for restriction of implementation, the framework has high portability. The mechanism view is used to choose mechanisms of a whitebox framework, which is relatively easy to develop, or a blackbox framework, which is relatively easy to use. By using this guideline, the framework has high maintainability. Moreover, we describe an example of applying our approach to the framework development for an industrial monitoring application, and demonstrate the effectiveness of our approach.

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，単にフレームワークと呼ぶ）が注目されてきた．フレームワークは，アプリケーション間に共通した部分の実装を共有可能なスケルトンとしてモジュール化し，アプリケーションに固有の部分のみを拡張することにより，安定した品質のソフトウェアを効率良く作り上げることをねらいとする[1]．フレームワークを適用することにより，生産性，再利用性などの向上が期待できる．現在では，Graphical User Interface（GUI）[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)]もいくつか報告されており，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

† 株式会社東芝 e-ソリューション社 SI 技術開発センター
Systems Integration Technology Center, e-Solutions Company, Toshiba Corporation
†† 神奈川工科大学情報工学科
Department of Information and Computer Sciences, Kanagawa Institute of Technology

レームワークを開発することは容易ではない．

本稿では，フレームワーク開発のための3ビューモデル，および3ビューモデルを用いたフレームワーク開発プロセスを提案する．3ビューモデルでは，フレームワークを開発する際に着目するべき観点として，ソフトウェアの品質特性である再利用性，移植性，保守性を向上することを重視する．これらの品質特性を考慮するために，“ドメイン分析”，“レイヤ”，“メカニズム”という3つのビューを示す．さらに，これらのビューに基づいたフレームワーク開発プロセスを示す．

2章では，従来の開発手法の課題を整理する．3章では，フレームワーク開発のための3ビューモデルについて示す．4章では，3ビューモデルに基づくフレームワーク開発プロセスについて示す．5章では，本手法によるシステム開発事例として，社会インフラとして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．6章では，適用事例を通じて本手法を考察する．7章では，本研究と関連する研究を取り上げて比較する．8章では，本研究をまとめ，今後の課題を示す．

2. 従来のフレームワーク開発技術

フレームワークを開発するための技術という観点から，従来のオブジェクト指向分析・設計手法，次にフレームワーク開発手法の課題について示す．

2.1 オブジェクト指向分析・設計手法

従来のオブジェクト指向分析・設計手法として，OMT法[9]，Booch法[10]などがあり，これらを統合したRational Unified Process（RUP）[11]がある．RUPでは，“4+1”ビューモデル[12]というモデルがあり，異なる担当者が異なる点に着目するというビューに立ち，論理ビュー，プロセスビュー，実装ビュー，配置ビュー，ユースケースビューという5つのビューを定義している．しかし，単一のシステムを開発することを前提としており，数多くの類似システムの仕様の相違を考慮して開発することを前提としていない．

2.2 フレームワーク開発手法

フレームワーク開発手法として，ホットスポット駆動手法[13]，ホットスポットサブシステム導入による手法[14]，データ中心アプローチとユースケースに基づく手法[15]，大規模システム向けのレイヤ構造導入による手法[16]がある．

ホットスポット駆動手法[13]では，既存のオブジェクト指向方法論を拡張し，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．また，ホットスポットサブシステム導入による手法[14]では，ホットスポットサブシステムと呼ばれるホットスポットに応じた汎化したクラス構成を提供している．これらの手法は，ホットスポットを満たすフレームワーク設計に関して焦点を当てている．しかし，ドメイン分析の結果を反映し，数多くの類似システムを統一的にどのようにモデリングするかといったフレームワーク分析に関して着目していない．

データ中心アプローチとユースケースに基づく手法[15]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あ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．しかし，このアプローチ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．その他の特徴を持つドメインについては触れていない．

大規模システム向けのレイヤ構造導入による手法[16]では，大規模システム向けとしてフレームワークをApplication，Business Section，Business Domain，Desktop，Technical Kernelの5つのレイヤに分割することを提案している．しかし，再利用性とリアルタイム性といったトレードオフとなるシステム要件を調整することについては触れていない．

3. フレームワーク開発向け3ビューモデル

本章では，2章で述べた従来の手法の課題を解決するために，フレームワーク開発のための3ビューモデルを提案する．なお，3ビューモデルに基づくフレームワーク開発プロセスについては4章に提案する．本手法が対象とするフレームワークは，GUI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は，対象システムを多くのユーザへ迅速に提供することが求められ高いリピート性を持つシステムへ適用される．

まず，3.1節で本手法で対象とするフレームワークで主に求められる再利用性，移植性，保守性というソフトウェアの品質特性の必要性を明らかにする．さらに，個々の品質特性を向上するための観点である“ドメイン分析”，“レイヤ”，“メカニズム”という3つのビューについて，それぞれ3.2節〜3.4節に示す．さらに，ビューどうしの関係を3.5節に示す．

3.1 フレームワーク開発で求められる要件

フレームワークに求められる要件として次の3点を

あげる．2 章で示した従来の手法には，これらのすべての要件について触れておらず，また 1 つの要件を取り上げても不十分な点がある．

再利用性 フレームワークを構築するためには，対象ドメインの特徴を十分吟味する必要がある．そのためには，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検討を重ねて，適切なオブジェクトを抽出しモデルを洗練するためのフレームワーク分析に関する議論が必要である．従来の手法では，フレームワーク設計に関する議論に焦点を当てている．

移植性 システムによっては，リアルタイム性などの実装上の制約を考慮しなければならない場合もある．フレームワークの性能を向上させるために，フレームワークの一部を交換することも考えられる．従来の手法では，フレームワークの一部を交換可能な構造について十分議論してない．

保守性 フレームワークのメカニズムとして，一般にホワイトボックス方式とブラックボックス方式がある[17]．前者は開発が容易で利用が困難であり，後者は開発が困難で利用が容易である．仕様変更などでフレームワークを修正する可能性が考えられるが，そのメカニズムにより修正のしやすさやシステムに与える影響が異なる．従来の手法では，メカニズム方式を選択する指針が述べられていない．

3.2 ドメイン分析ビュー

ドメイン分析ビューは，再利用性に焦点を当てたビューである．フレームワーク開発では，ドメイン分析を十分に行いモデリングすることが重要である．一般に，ドメイン分析とは，対象システムが属するドメインが本来持つあらゆる性質や開発上の多くの知識を十分に分析することである[18]．ドメイン分析の結果を反映して，数多くの類似システムを統一的にモデリングすることが必要である．ドメインごとに基本となるソフトウェア構造を定義するために，我々はドメイン参照モデルという概念をすでに提案している[19]．ドメイン参照モデルに基づき，フレームワークの分析モデルを構築した後で，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[13]を特定し，これをフレームワークの設計モデルに反映する．

3.2.1 ドメイン参照モデルの定義

ドメイン参照モデルとは，ドメインのソフトウェア構造をシステムレベルで示したものである．ドメイン参照モデルを適用することで，数多くのシステムを統一的にモデリングすることができる．ドメイン参照モデルは，次の条件を満たす．

- システムの静的な側面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ドメイン参照モデルの各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，Unified Modeling Language（UML）[20]のステレオタイプとして表現できる．

3.2.2 ドメイン参照モデルの例

対象とするドメインに応じてドメイン参照モデルを定義する．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に分ける．工業分野では，“監視”，“制御”，“組み込み” という 3 つのドメインに分ける．それぞれのドメインに応じて次のドメイン参照モデルを定義する．

監視 PERS（Presentation-Entity-Relay-Service）モデル[19]：監視業務を実現する Service と，監視対象機器を模擬する Relay を分離する特徴を持つ．

制御 SMUPE（Service-Media-Unit-Presentation-Entity）モデル[8]：制御業務を実現する Service，制御対象機器を物理的に写像した Unit，および論理的に写像した Media に分離する特徴を持つ．

組み込み デセセ（デバイス–制御–制約）モデル[21]：デバイスを動作させる制御と，制御の際の制約を分離する特徴を持つ．

5 章で適用事例として監視システムを取り上げるため，ここでは PERS 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（たとえば，カメラ，ゲート，シャッタ，センサなど）に対して遠隔の端末から動作指示やモード設定を与えたり，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）を取得したりすることで，監視対象機器を通じて監視業務を行うドメインである．このドメインは，次の特徴を持つ．

- 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監視業務を実行する．
- 監視対象機器の通信プロトコルが，同機能の監視対象機器であっても機種の相違で異なる場合がある．

これらの特徴をふまえたモデリングを実現するために，**図 1** に示す PERS モデルを定義する．以下に，各モデル要素の位置付けを示す．

Presentation 情報を操作者へ視覚的な情報として提供するもの．

Entity ドメインのデータとそのデータに対する操

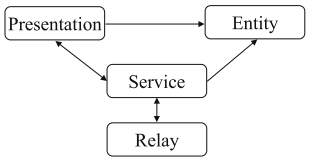

図 1 PERS（Presentation-Entity-Relay-Service）モデル
Fig. 1 PERS (Presentation-Entity-Relay-Service) model.

作を提供するもの．

Relay 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．

Service Entity を管理し Presentation と Entity を結び付け，システムのサービスを提供するもの．Relay と監視データの通信を行う．

3.2.3 他の再利用技術との関連

ドメイン参照モデルという概念は，再利用技術の一種である．ドメイン参照モデル以外の再利用技術には，ドメインモデル[18]，アナリシスパターン[22]，デザインパターン[23]，フレームワークなどがある．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．ドメイン参照モデルは，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，ドメイン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．

3.3 レイヤビュー

レイヤビューは，移植性に焦点を当てたビューである．システムによっては，プラットフォームが変更されたり，リアルタイム性などの実装上の制約からフレームワークの一部を交換することが考えられる．そこで，複数のシステム要件やインフラの変化などに対応するためにフレームワークを次の 3 つのレイヤに分割する．

インフラレイヤ インフラを隠蔽する機能を持つ．

ジェネリックレイヤ 汎用的に用いる機能を持つ．

ドメインレイヤ ドメインに依存する機能を持つ．

たとえば，開発初期のフレームワークでは，再利用性や拡張性を重視していたが，システムの性能を満足できなかったとする．システムの性能は，インフラを効率的に利用しているかに依存する場合が多い．したがって，インフラフレームワークのみを調整することで，システムの性能を満足させることが可能となる．

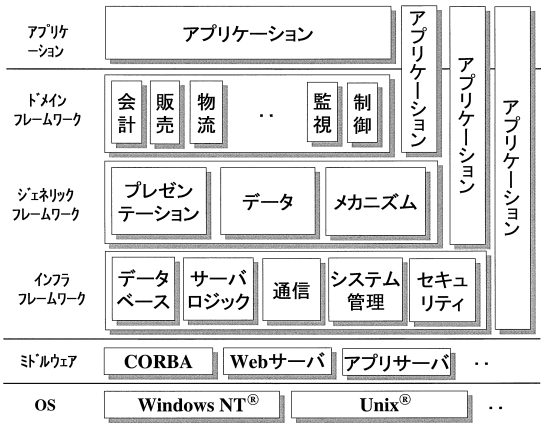

図 2 レイヤフレームワークのアーキテクチャ
Fig. 2 Architecture of layered frameworks.

我々は，すでにレイヤビューを重視した手法を適用した事例[8]により効果を確認している．

さらに，3 つのフレームワーク内部を**図 2** に示すカテゴリに分類する．体系的なフレームワークアーキテクチャを定義することで，フレームワークを部分的に交換したりするなどの柔軟な対応を狙いとする．なお，図 2 は，レイヤフレームワークのカテゴリ体系を表しているであり，すべてのシステムでまったく同じフレームワークを活用するという意味ではない．たとえば “データベース” カテゴリのフレームワークは，すべてのドメインで利用可能なフレームワークを意味しているのではない．インフラレイヤフレームワークには，カテゴリとして “データベース” があるという意味である．

3.3.1 インフラレイヤフレームワーク

Windows NT☆や UNIX☆☆などの OS や，Web サーバなどのミドルウェアに依存した機能を含むレイヤである．図 2 に示すように，“データベース”，“サーバロジック”，“通信”，“システム管理”，“セキュリティ” カテゴリから構成される．

データベース データベースあるいはそれに相当する永続的なデータにアクセスする機能を備えている．

サーバロジック Web サーバやアプリケーションサーバ上で実行するロジック機能を備えている．

通信 分散環境での通信として，イベント通知やメッセージ通知などの機能を備えている．

システム管理 機器構成情報などのシステム管理上必

☆ Windows NT は，米国 Microsoft Corporation の米国およびその他の国における登録商標である．

☆☆ UNIX は，X/Open Company Limited が独占的にライセンスしている米国ならびに他の国における登録商標である．

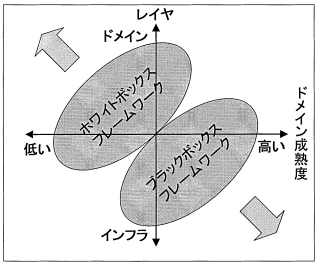

図 3 フレームワークのメカニズムの選択基準
Fig. 3 Guidelines for choosing the framework mechanism.

要な機能を備えている．

セキュリティ 認証やアクセス制御といったセキュリティ上必要な機能を備えている．

3.3.2 ジェネリックレイヤフレームワーク

ドメインにかかわらず汎用的に用いる機能を含むレイヤである．図 2 に示すように，“プレゼンテーション”，“データ”，“メカニズム” カテゴリから構成される．

プレゼンテーション GUI など視覚的な表現形式にかかわる機能を備えている．

データ データ変換など，データの表現形式にかかわる機能を備えている．

メカニズム 状態遷移，オブジェクトメッセージ通信など，振舞いにかかわる機能を備えている．

3.3.3 ドメインレイヤフレームワーク

ドメインに依存した機能を含むレイヤである．図 2 に示すように，情報分野として “会計” や “販売” などのドメインが存在し，工業分野として “制御” や “監視” などのドメインが存在する．

3.4 メカニズムビュー

メカニズムビューは，保守性に焦点を当てたビューである．一般に，フレームワークのメカニズムとして，ホワイトボックス方式とブラックボックス方式がある．適用するシステムの特徴に応じたメカニズムを選択することが重要である．選択基準を 3 ビューモデルの他の 2 つのビューを用いて示したものが，**図 3** である．以下に，それぞれの観点に関して示す．

3.4.1 ドメインの成熟度

ドメインの成熟度は，システムの機能の安定性に依存する．機能の安定性とは，システムに対する仕様変更の範囲や頻度を意味する．図 3 に示すように，ホワイトボックス方式は未成熟なドメイン向きであり，ブラックボックス方式は成熟したドメイン向きであると考える．

未成熟なドメインでは，フレームワークで提供される機能をベースに，必要に応じてカスタマイズする必要がある．ホワイトボックス方式では，フレームワークで提供されるクラスをシステム開発者にすべて公開する形態であるので，仕様変更に応じて自由にカスタマイズすることができる．一方，成熟したドメインでは，フレームワークとして安定した機能を提供し，厳密に正しく利用可能なことが求められる．ブラックボックス方式では，フレームワークで提供されるコンポーネントのインタフェース，セマンティックスを厳密に定義し，加えて品質の保証がされているため，システム開発者にとってフレームワークを正しく利用できる．

3.4.2 レ イ ヤ

レイヤとは，3.3 節で述べたインフラ，ジェネリック，ドメインレイヤを指す．図 3 に示すように，インフラレイヤよりは，ブラックボックス方式向きであり，ドメインレイヤよりは，ホワイトボックス方式向きであると考える．

インフラレイヤよりでは，ドメインに依存せず汎用的に利用する場合が多く考えられる．内部構造が固定的であり，どのクラスを利用するかが決められているホワイトボックス方式では，汎用性を得ることが容易でない．むしろ，コンポーネントを組み合わせることが可能であり，必要なコンポーネントのみを利用することが可能なブラックボックス方式が対処しやすい．また，インフラレイヤよりでは，個々のクラス単位で見ると仕様が安定しており普遍的な機能を実装するので，コンポーネントとして実現しやすい．一方，ドメインレイヤよりでは，ドメインに依存した機能を実装する．ドメインの成熟度に依存するが，一般にドメインが将来どう変化するかを特定することは難しく，初期の適用ではホワイトボックス方式が向いている．ドメインが成熟した時点で，ブラックボックス方式へと移行することは考えられる．

3.5 ビューどうしの関係

3.2 節〜 3.4 節で示した 3 つのビューは相互に影響し合う．以下にビューどうしの関係を示す．

3.5.1 ドメイン分析ビューとレイヤビュー

ドメイン分析により開発制約を明らかにする．開発制約とは，インフラや性能条件など実現するうえで制約となる点を指す．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ことも重要である．一方，ドメイン分析により得られた開発制約をもとに必要なレイヤを定義する．レイヤ定義により，実装するフレームワークのスコープが明らかになる．レイヤごとにドメイン分析の結果により洗い出されたホットスポットを分類する．

3.5.2 レイヤビューとメカニズムビュー

レイヤ定義により実装するフレームワークのスコープが明らかになる．個々のレイヤあるいはレイヤ内部のカテゴリごとに，適切なメカニズムを選択する．一方，個々のレイヤあるいはレイヤ内部のカテゴリごとにメカニズムを決定することにより，レイヤ間あるいはレイヤ内部のカテゴリ間の結合方式が明らかになる．ホワイトボックスフレームワークの場合には，継承型あるいは委譲型の結合方式となる．ブラックボックスフレームワークの場合には，プラグイン型の結合方式となる．

3.5.3 メカニズムビューとドメイン分析ビュー

ドメイン分析によりドメインの成熟度を明らかにする．ドメインの成熟度に応じて，適切なメカニズムを選択する．一方，メカニズムを決定することにより，カスタマイズ方式が明らかになる．ホワイトボックスフレームワークの場合には，フレームワークで提供されるクラスをシステム開発者が参照しカスタマイズする．ブラックボックスフレームワークの場合には，フレームワーク提供されるクラスのインタフェースをシステム開発者が参照しカスタマイズする．

4. 3ビューモデルに基づく開発プロセス

3 章で示した 3 ビューモデルを活用したフレームワーク開発プロセスを提案する．ドメイン分析ビュー，レイヤビュー，メカニズムビューの順で開発を進める．システム分析およびオブジェクト指向分析フェーズでドメイン分析ビューを利用し，オブジェクト指向設計フェーズでレイヤビューおよびメカニズムビューを利用する．以下に個々の作業項目の概要を示す．

4.1 ドメイン分析フェーズ

ドメイン分析作業の目的は，システムが属するドメインが本来持つあらゆる性質や開発上のさまざまな知識を十分に分析することである．

Process1：ドメインモデル構築 ドメインの知識や情報を整理し，ドメインモデルを構築する．ドメインモデルの記述は，既存のドメイン分析手法の 1 つである D-AME[18]などを参考にする．

4.1.1 システム分析フェーズ

システム分析作業の目的は，システム内部の構造と振舞いをユーザの視点から規定することである．

Process2：ユースケース分析 システム化の範囲を決定し，利用者から見たシステムの機能であるユースケースを列挙し，利用者または外部システムとの具体的なインタラクションを規定する．

Process3：システム構成決定 ユーザの視点からシステムの構成方法を決定する．ハードウェア，ミドルウェア，プログラミング言語などの構成を決定する．また，再利用可能なソフトウェアがあるかどうか，独自に開発するソフトウェアは何かを洗い出す．

Process4：ドメイン参照モデル構築 既存のドメイン参照モデルを参照し，システム要求をドメインモデルからドメイン参照モデルにマッピングし整理する．必要ならば既存のドメイン参照モデルを拡張・改良する．なお，既存のドメイン参照モデルに適切なモデルがなければ，新たにドメイン参照モデルを次のような手順で構築する．

Process4.1：初期オブジェクトの抽出 Process2 で列挙したユースケースを実行するためのオブジェクトを抽出する．この時点で完全なオブジェクトをすべて列挙する必要はない．したがって，すべてのユースケースについて検討する必要はなく，典型的な基本的なユースケースについてオブジェクトを抽出すればよい．また，ユースケースに対してシステムが行うべき振舞いを分析し，各オブジェクトの役割分担を決める．

Process4.2：初期オブジェクトの分類 オブジェクトの役割を吟味しオブジェクトを分類する．

Process4.3：ドメイン参照モデルのモデル化 オブジェクトの分類が固まったならば，システムの構造をドメイン参照モデルとしてモデル化する．

4.1.2 オブジェクト指向分析

オブジェクト指向分析作業の目的は，システム要求を基にシステムの構造と振舞いのモデルをオブジェクトとして明らかにすることである．

Process5：オブジェクト分析 ドメイン参照モデルのモデル要素が示すオブジェクト抽出の観点に従ってオブジェクトを抽出する．

Process6：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．

4.1.3 オブジェクト指向設計

オブジェクト指向設計作業の目的は，オブジェクト指向分析で得られたモデルをどのようにして実現するかを明らかにすることである．

Process7：アーキテクチャ決定 ドメイン分析の結果から，開発するフレームワークに必要なアーキテクチャを決定する．アーキテクチャとは，図 2 で示すレイヤ構造とレイヤ内部のカテゴリを決定することを意味する．また，再利用可能なフレームワークのカテゴリや，独自に開発するフレームワークのカテゴリを図 2 で明らかにする．

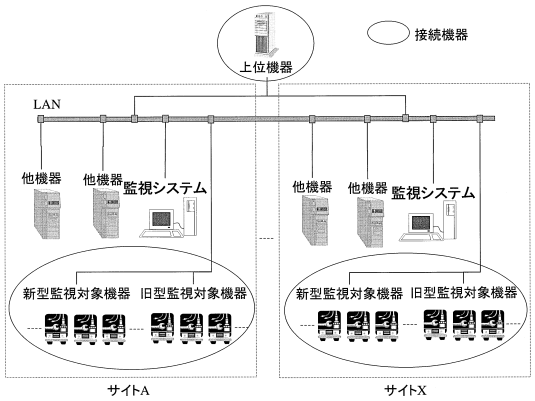

図 4 監視システム
Fig. 4 Monitoring systems.

Process8：メカニズム決定 ドメイン分析とレイヤ分割の結果から，レイヤごとあるいはレイヤ内部のカテゴリごとにメカニズムを決定する．メカニズムとは，継承型のホワイトボックスフレームワークか，プラグイン型のブラックボックスフレームワークかを表す．

Process9：フレームワーク設計 ホットスポットがオブジェクトモデルのどこに影響を与えるかを整理し，構造と振舞いのモデルを定義する．Process7 のアーキテクチャ決定で明らかになったレイヤとカテゴリに対して，個々の構造と振舞いを定義していく．また，Process8 のメカニズム決定で明らかになったメカニズムを実現するために構造を見直す．

5. 監視システム開発への適用

3 章で示したモデル，および 4 章で示した開発プロセスを適用した監視システム向けフレームワークのプロトタイプ開発事例について示す．

5.1 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 4** 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．サイト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，ソースコードの規模は約 10 万ステップであり，我々の数多くのドメインのシステム開発の中では中規模システムである．

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．たとえば，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，監視対象機器で収集したデータは，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挙する．

- 監視対象機器の状態を表示し，必要に応じてモード設定を行う．
- 監視対象機器で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．

5.2 フレームワーク開発プロセス

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，顧客ごとの仕様の違いについて十分吟味した．したがって，フレームワーク開発のための Process1〜5 についてのモデリング作業において，これまでのシステム開発でのモデリング結果を見比べることで，フレームワークとしてのモデルの質を高めることができた．

5.2.1 ドメイン分析

Process1：ドメインモデル構築 ドメインそのものの把握と，システム要件の整理という 2 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，新型の監視対象機器が設置されたり，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．

5.2.2 システム分析

Process2：ユースケース分析 システムの機器構成や仕様によってシステムが提供する機能は若干異なるが，本システムが提供する全機能を列挙し，ユースケース図に表現すると**図 5** のようになる．システムが提供する機能については，すでに 5.1 節で述べた．

Process3：システム構成決定 ハードウェアは通常の PC を用い，OS には Windows NT を採用し，プログラミング言語は効率を考慮し C++言語を選択した．

Process4：ドメイン参照モデル構築 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，3.2.2 項で示した PERS モデルを構築した．本システムに必要なオブジェクトを洗い出し，シー

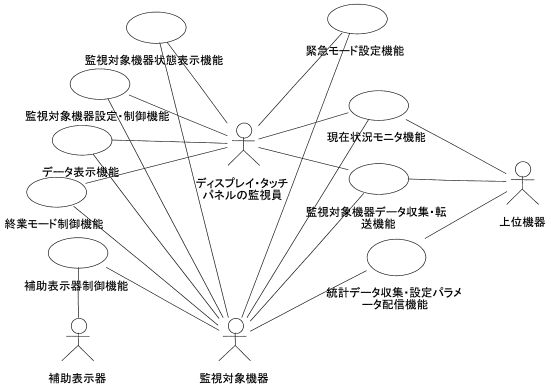

図 5 ユースケース図
Fig. 5 Use case diagram.

ケンス図の記述により，ドメイン参照モデルを検討した．以下に初期オブジェクトの役割と PERS モデルのモデル要素との対応を簡単に示す．

GUI 画面オブジェクト 監視状況を GUI 表示する．たとえば，設定画面オブジェクトなどである．これは PERS モデルの Presentation に相当する．

監視業務にかかわるデータオブジェクト 監視対象機器に関連する一連のデータ群を指す．たとえば，監視対象機器から収集したデータオブジェクトなどである．これは PERS モデルの Entity に相当する．

外部機器との通信を行うオブジェクト 外部機器の機種の相違から異なる通信プロトコルを扱う．たとえば，新型監視対象機器向け通信オブジェクトなどである．これは PERS モデルの Relay に相当する．

監視業務サービスを行うオブジェクト 監視対象機器の動作モードを設定したり，監視データを収集したりする．たとえば，監視対象機器から履歴データを収集するサービスオブジェクトなどである．これは PERS モデルの Service に相当する．

5.2.3 オブジェクト指向分析

Process5：オブジェクト分析 PERS モデルに従いオブジェクト指向を適用すると考えた Service のオブジェクト分析を行った．基本的には，監視システムで提供するサービスごとにオブジェクトを抽出した．

Process6：ホットスポット分析 すでにオブジェクト指向技術を導入した製品開発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るべき仕様，システムごとに異なる仕様を吟味した．その結果，主に考慮するべき点として次に示す項目があがった．

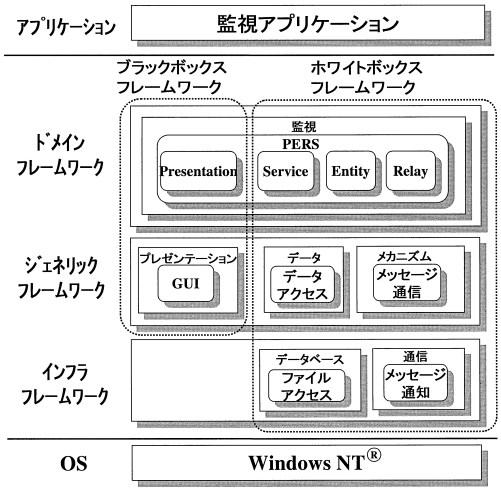

図 6 監視システムのアーキテクチャ
Fig. 6 Architecture of monitoring systems.

- GUI については，たとえば扱うデータ種別の追加などにともないボタンを追加する場合がある．Presentation に関連する．
- サービスについては，たと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．Service に関連する．
- 扱うデータについては，たとえばエラー情報の項目が異なる場合がある．Entity に関連する．
- 監視対象機器の変更などにより通信プロトコルが変更になる場合がある．Relay に関連する．

PERS モデルのモデル要素ごとにホットスポットの割合を求めると，Presentaion（約 20%），Entity（約 20%），Relay（約 10%），Service（約 50%）であり，ホットスポットは Service に最も集中している．

5.2.4 オブジェクト指向設計

Process7：アーキテクチャ決定 レイヤ構造を特定し各レイヤのカテゴリを定義した．**図 6** に，監視システムのアーキテクチャを示す．影の付いた四角はフレームワークのカテゴリを表す．また，影のついた角が丸い四角はフレームワークの構成要素を表す．

インフラレイヤフレームワーク 通信とデータベースという 2 つのカテゴリを定義した．通信カテゴリでは，プロセス間のメッセージ通信部品を実現する．また，データベースカテゴリでは，共有メモリや共有ファイルのデータへのアクセス部品を実現する．

ジェネリックレイヤフレームワーク プレゼンテーションとメカニズムという 2 つのカテゴリを定義した．プレゼンテーションカテゴリでは，監視画

面の GUI 部品およびオブジェクトレベルでのデータアクセス部品を実現する．一方，メカニズムカテゴリでは，オブジェクトレベルでのメッセージ通信部品を実現する．

ドメインレイヤフレームワーク PERS モデルとして機能する部品を実現する．

Process8：メカニズム決定 図 6 に示すように，Presentation に関連する構成要素については，ブラックボックス方式を採用し，それ以外のモデルに関連する構成要素については，ホワイトボックス方式を採用した．Presentation については，GUI 画面でありコンポーネント指向と親和性が高く，ブラックボックス方式が適切と考えた．一方，ドメインとしては未成熟であり，随時アプリケーションをカスタマイズする必要があるため，Presentation 以外のモデル要素については，ホワイトボックス方式が適切と考えた．

Process9：フレームワーク設計 本システムでは，Presentation をブラックボックスフレームワークとして，それ以外をホワイトボックスフレームワークとして設計した．

まず，個々のモデル要素ごとにプロセスを設け，プロセス間のメッセージ通信により監視サービスを進める方法を採用した．メッセージ通信の仕組みは，ジェネリックフレームワークのメカニズムカテゴリとして設計した．ジェネリックレイヤフレームワークのメカニズムカテゴリは，内部でインフラフレームワークの通信カテゴリを利用する関係となる．

また，ホットスポット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群を整理した．ここでは，ドメインレイヤフレームワークの Service を中心に述べる．このモデル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とに分けた．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，Entity 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，ジェネリックレイヤフレームワークのメカニズムカテゴリの部品を利用してシステム内部で取り扱うデータを提供する．

6. 考　　察

プロトタイプ開発を通じての本手法の考察を示す．まず，本手法適用による費用対効果，本手法適用の優位性について順に示す．

6.1 費用対効果

本システムで特に重要視しているシステム要件は，再利用性と拡張性である．これらを判断する 1 つの方法として，開発工数について本手法に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，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うえでフレームワーク適用の効果を判断する必要がある．

6.1.1 評価対象

プロトタイプ開発では，PRES モデル全体のうちホットスポットが集中している Service を評価対象とした．

6.1.2 評　価　式

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．

$$DfwDevCost < n(ObjAplDevCost - DfwAplDevCost)$$

ここで，DfwDevCost はフレームワーク開発工数，n はリピート回数，ObjAplDevCost はフレームワーク未適用のシステム開発工数，DfwAplDevCost はフレームワーク適用のシステム開発工数を表す．

6.1.3 想定する仕様変更

実際の仕様変更を参考に次のような仕様変更を想定した．最初の 2 つの仕様変更はサービスの追加にあたり，最後の仕様変更はサービス内容の変更にあたる．

- 上位機器向けサービス機能の追加上位機器との通信で発生する機能を実現する．ここでは，監視対象機器から収集したデータをまとめ，集計データを転送する機能を追加する．
- 状態通知先の追加状態通知先を画面だけでなく上位機器にも送信する．
- インターフォン機能の追加監視対象機器に対する異常通知処理の中でインターフォン機能を追加する．

6.1.4 開発工数の比較

オブジェクト指向分析までの作業内容は，フレームワーク未適用のオブジェクトモデルをほぼそのまま再利用できると考え，オブジェクト指向設計，実装，デバッグ，テストのフェーズについて，フレームワークを適用する場合とそうでない場合の開発工数を比較した．

その結果を**表 1** に示す．表中の “Dfw 未適用” とは，フレームワークを適用しない開発工数を表し，“Dfw 適用” とは，フレームワークを適用した開発

表 1 作業項目に対する開発工数の比較
Table 1 Comparison of development cost in each work-item.

作業項目	Dfw 未適用 (h)	Dfw 適用 (h)	差分 (h)
設計	480	475	5
実装	762	754	8
デバッグ	480	475	5
結合テスト	120	39	81
総合テスト	120	39	81
合計	1,962	1,782	180

工数を表す．前者はこれまでの製品開発の経験から試算した値であり，後者はプロトタイプ開発での実測値である．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表 1 の結果から約 90%程度 (=1,782h/1,962h*100) 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 1,032h であり，表 1 の結果から 5〜6 リピート (n ≧ 1,032h/(1,962−1,782)h) 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費用対効果の点で有利であると考えられる．

6.2 3ビューモデル適用の優位性

3ビューモデルを適用することで，再利用性，移植性，保守性を同時に満足することができる．以下に，個々の品質特性における優位性を考察する．

6.2.1 再利用性

6.1.3 項で示した上位転送機器向けサービス機能の追加における旧型および新型監視対象機器の扱いを取り上げて説明する．本システムでは，監視対象機器を徐々に入れ替えるので，旧型と新型監視対象機器の構成が変化する．監視対象機器が旧型か新型かで収集データを集める機能の手順が異なる．手順の違いには，通信プロトコルレベルと業務ロジックレベルの違いがあるが，ここでは通信プロトコルレベルの違いを取り上げる．

ドメイン分析ビューにより，PERS モデルと呼ぶドメイン参照モデルを活用し，オブジェクトモデリングを行った．**図 7** に収集データを集める機能のシーケンスの概要を示す．旧型監視対象機器の場合は，図 7 の 2〜5 に示すように，監視システムからデータ収集指示を出すと，旧型監視対象機器は単純に収集データを送信する．一方，新型監視対象機器の場合は，図 7 の 7〜11 に示すように，監視システムからデータ収集指示を出すと，いったん指示受信通知を返した後，新型監視対象機器は収集データを返す．指示受信通知は，データ収集指示は受け取ったが，まだ収集データを返していない状態を表す．

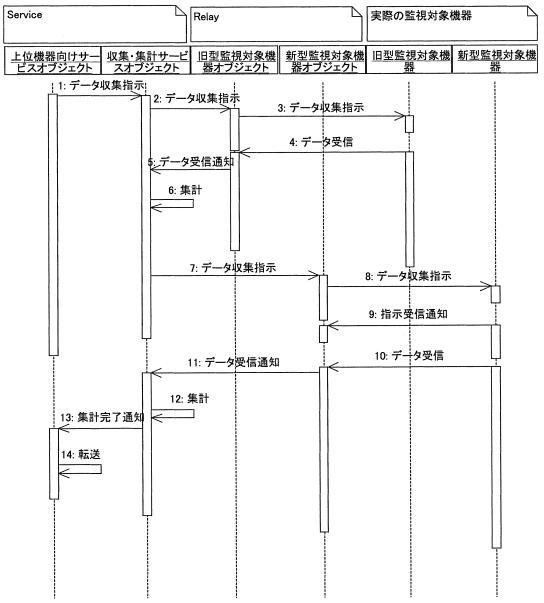

図 7 収集・転送機能のシーケンス図の概要
Fig. 7 Outline of the sequence diagram of collecting and sending service.

このような仕様に対して，PERS モデルを利用することにより柔軟に対応することができる．監視対象機器に対するシーケンスの違いについては，図 7 に示す旧型監視対象機器オブジェクトおよび新型対象機器オブジェクトに代表される Relay を設けることで対処する．一方，監視対象機器に対して収集指示を出す図 7 の収集・集計サービスオブジェクトは，旧型監視対象機器オブジェクトおよび新型監視対象機器オブジェクトからのデータ受信通知を受け取るだけである．したがって，監視対象機器に対するデータ収集機能を実現する際に，収集・集計サービスオブジェクトが監視対象機器が旧型か新型かを意識する必要はない．

以上のように，PERS モデルの導入により，新型と旧型の収集機能の違いは Relay 内部に局所化できるので，システム全体としての再利用性が向上する．

6.2.2 移植性

当初開発したフレームワークは，リアルタイム性については一部調整を必要とした．監視対象機器との通信処理時間は，監視対象機器のシミュレータを活用した結果，モード設定などの数値データについては数百 ms 程度のリアルタイム性を確保することができた．また，監視対象機器から受信した大量データの集計処理は，データ量が増えるとリアルタイム性に問題があった．ここで h，その改善について取り上げる．

レイヤビューにより，フレームワークにレイヤ構造を導入し，フレームワークを分割して開発した．イン

フラフレームワークでは，プロセス間のメッセージ通知やファイルアクセスといったインフラの資源を利用する部品を開発した．また，ジェネリックフレームワークでは，オブジェクトレベルでのメッセージ通信など他のシステムにも適用可能な部品を開発した．さらに，ドメインフレームワークでは，多様なユーザ仕様に対し柔軟に対応する部品を開発した．このようなレイヤ構造を設けることで，先に示した大量データの集計処理に対するリアルタイム性の問題は，インフラフレームワークのデータベースカテゴリの約 10 個のクラス部品のみをチューニングすることで改善できた．

以上のように，フレームワークの特定の部品を容易に入れ替えることが可能であるので，移植性が向上する．

6.2.3 保 守 性

本システムは，数十ユーザに展開するシステムである．ユーザによっては要求仕様のバラツキがあり，たとえば新型の監視対象機器が設置されたり，サービスが追加されたりする．また，同じユーザであっても数か月に 1 回の頻度で仕様変更がある．平均すると半年で 10 ユーザ以上開発することが通常となっており，非常にリピート性が高く，仕様変更が多いドメインである．この未成熟なドメインにおいては，フレームワークのリファインを余儀なくされ，その対処方法について取り上げる．

メカニズムビューにより，ドメインの成熟度に応じたメカニズムを選択した．約 400 個程度あるフレームワークのクラスのうち約 7 割のクラスは，基本機能を有するもののアプリケーションでカスタマイズしなければならないホットスポットとなるクラス群である．このような場合には，アプリケーション側で自由にカスタマイズすることができるように，ブラックボックス方式ではなくホワイトボックス方式で実現する方が有利である．

以上のように，フレームワークが対象とするドメインの成熟度に合わせた適切なメカニズムを選択できるので，保守性が向上する．

7. 関 連 研 究

開発手法の観点から，本手法と他の関連研究を比較する．他の関連研究の概要についてはすでに 2 章で示した．ここでは，本手法との差異のみを述べる．

オブジェクト指向分析・設計手法である RUP[11] では，“4+1” ビューモデル[12] が提案されている．本手法では，ソフトウェアの異なる品質要因である再利用性，移植性，保守性に着目するというビューにたっている．“4+1” ビューモデルと本手法とは，着目するビューが異なるが相互に補完可能な技術である．

次に，フレームワーク開発手法との差異について述べる．2 章で述べた手法は，いずれもドメイン分析あるいはレイヤという 1 つのビューを重視した手法である．本手法では，ドメイン分析，レイヤ，メカニズムという 3 つのビューに着目している．

ホットスポット駆動手法[13] およびホットスポットサブシステム導入による手法[14] では，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．本手法では，ドメイン参照モデルという概念を用いてフレームワーク分析に焦点を当てている．従来の手法と我々の手法とは，着目している点が異なるが開発プロセスの中で順に適用可能な技術である．

データ中心アプローチとユースケースに基づく手法[15] 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．しかし，このアプローチ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．本手法は，ドメイン分析によりドメインの特徴をドメイン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できると考える．

大規模システム向けのレイヤ構造導入による手法[16] は，フレームワークをレイヤ分割するという点で，我々と同じ立場を採っている．しかし，システム要件に対してフレームワークでどのように対処すればよいかについては触れていない．本手法は，レイヤビューにより再利用性とリアルタイム性といったトレードオフとなるシステム要件を調整することができる．大規模システム向けのレイヤ構造導入による手法で提案しているレイヤは，我々が提案するレイヤのカテゴリとして実現でき，融合することが可能であると考える．

8. お わ り に

本稿では，フレームワーク開発のための 3 ビューモデル，および 3 ビューモデルに基づくフレームワーク開発プロセスを提案した．また，本手法を監視システムへ適用し考察を述べた．

ソフトウェアの品質特性である再利用性，移植性，保守性を向上するために，ドメイン分析，レイヤ，メカニズムという 3 つのビュー，およびこれらのビューに基づくフレームワーク開発プロセスを提案した．ドメイン分析ビューでは，再利用性に着目し，ドメイン参

照モデルを用いて数多くの類似システムを統一的にモデリングすることを可能にする．次に，レイヤビューでは，移植性に着目し，インフラ，ジェネリック，ドメインという3つのレイヤを設け，トレードオフとなるシステム要件を調整することを可能にする．さらに，メカニズムビューでは，保守性に着目し，ホワイトボックス方式とブラックボックス方式の選択基準を明らかにし，ドメインの特徴に応じた選択をすることを可能にする．また，監視システム向けフレームワークのプロトタイプ開発を通じて，本手法が有効であることを明らかにした．

今後は，次の3項目について検討することが課題である．第1に，本手法の効果をより定量的に評価することである．開発したフレームワークを使ったシステム開発におけるフレームワークのシステム全体に対する再利用率を評価することを検討している．第2に，本手法を適用したフレームワークを実際の製品開発に適用し，何回のリピート回数で開発コストを回収することができたかを定量的に評価することである．第3に，ドメインに応じてどのビューを重視するかなどの指針を考えることである．

参考文献

1) Fayad, M.E. and Schmidt, D.C.: Object-Oriented Application Frameworks Introduction, *Comm. ACM*, Vol.40, No.10, pp.32–38 (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*, Stanford University (1988).
4) Lewis, T., et al.: *Object-Oriented Application Framework*, Manning Publications (1993).
5) Franek, B. and 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) Alfredsen, K. and Saether, B.: An Object-Oriented Framework for Water Resource Planning, *Proc. 31st Hawaii International Conference on System Science*, 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.119-C, No.12, pp.1573–1581 (1999).
9) Rumbaugh, J., et al.: *Object-Oriented Modeling & Design*, Prentice Hall (1991).
10) Booch, G.: *Object-Oriented Analysis and Design with Applications*, Second Edition, Benjamin Cummings Publishing Company (1994).
11) フィリップ・クルーシュテン：ラショナル統一プロセス入門，ピアソン・エデュケーション (1999).
12) Kruchten, P.: Architectural Blueprints—The 4+1 View Model of Software Architecture, *IEEE Software*, Vol.12, No.6, pp.42–50 (1995).
13) Pree, W.: *Design Patterns for Object-Oriented Software Development*, Addison-Wesley (1994).
14) Schmid, H.A.: Design patterns for constructing the hot spots of a manufacturing framework, *Journal of Object-Oriented Programming*, Vol.9, No.3, pp.25–37 (1996).
15) 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).
16) Baumer, D., Knoll, R., Gryczan, G. and Zullighoven, H.: Large scale object-oriented software development in a banking environment, *Object-Oriented Programming, 10th European Conference*, pp.73–90, Springer Verlag (1996).
17) Fayad, M.E., Schmidt, D.C. and Johnson, R.E.: *Building Application Frameworks*, Addison-Wesley (1999).
18) 伊藤 潔，杵嶋修三，田村恭久，廣田豊彦，吉田裕之：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).
19) 早瀬健夫，安東孝信，丸尾秀史，新舘秀和，榎本秀明：ドメイン参照モデルに基づくオブジェクト指向開発，電気学会論文誌C（電子・情報・システム部門誌），Vol.122-C, No.2, pp.296–308 (2002).
20) Booch, G.: *The Unified Modeling Language User Guide*, Addison-Wesley (1998).
21) 池田信之，早瀬健夫，松本一教：制御ソフトウェア開発のための参照モデル，電気学会論文誌C（電子・情報・システム部門誌），Vol.121-C, No.6, pp.1049–1058 (2001).
22) Fowler, M.: *Analysis Patterns: Reusable Object Models*, Addison-Wesley (1997).
23) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns*, Addison-Wesley (1994).

(平成13年5月 1 日受付)
(平成14年5月15日採録)

早瀬　健夫（正会員）

1990 年 3 月上智大学理工学部機械工学科卒業．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 e-ソリューション社 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，電気学会，人工知能学会各会員．

松本　一教（正会員）

神奈川工科大学情報工学科助教授．1984 年 3 月九州大学理学部数学科卒業．1986 年 3 月同大学大学院総合理工学研究科修士課程修了．1986 年 4 月から 2002 年 3 月まで（株）東芝に勤務．2002 年 4 月より現職．博士（理学）．