ドメイン参照モデルに基づく ITS 画像交換システム開発

早瀬 健夫 *1，安東 孝信 *1，丸尾 秀史 *1，新舘 秀和 *2，榎本 秀明 *1
*1　株式会社東芝 e−ソリューション社　　*2　東芝 IT ソリューション株式会社
〒183-8512 東京都府中市片町 3-22
E-mail：takeo.hayase@toshiba.co.jp

あらまし

本稿では，ITS(Intelligent Transport Systems)画像交換システム開発におけるソフトウェアエラーを削減するためのオブジェクト指向開発アプローチについて述べる．本アプローチは，ドメインオブジェクトをモデル化するための基本的なモデルであるドメイン参照モデルに基づいている．エラーを削減するためには，リスク要因に対して影響を受ける箇所（ホットスポット）を明らかにする必要がある．各開発フェーズでホットスポットをドメイン参照モデルのモデル要素ごとに分類し，リスクアセスメントを確保する．本アプローチにより，本システムの接続テスト時に約 10 件程度のエラーに抑えることができた.

キーワード：オブジェクト指向，ドメイン参照モデル，開発手法，品質，ITS

Developing Video-Exchange Systems on ITS Based on Domain Reference Models

Takeo Hayase *1, Takanobu Ando *1, Hideshi Maruo *1,
Hidekazu Niitate *2, and Hideaki Enomoto *1
*1　e-Solutions Company, Toshiba Corporation　　*2　Toshiba IT-Solutions Corporation
3-22, Katamachi, Fuchu-shi, Tokyo 183-8512, Japan
E-mail：takeo.hayase@toshiba.co.jp

Abstract

This paper describes an object-oriented methodology for reducing software errors in developing video-exchange systems on ITS (Intelligent Transport Systems). Our approach is based on Domain Reference Models (DRM) that a basic model for modeling domain objects. To reduce software errors, we need to clarify hot spots, which we redefine as the parts of software that is customized to all risk factors. Software engineers classify hot spots into related components of the DRM each development phase, and maintain risk assessment. We could reduce to approximately ten software errors in connecting test of this system.

Keywords: object-oriented, domain reference model, methodology, quality, ITS

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，これまでに多くの適用事例があり，設計・実装の再利用により開発工数を短縮することが報告されている．一方，オブジェクト指向開発においても，開発したソフトウェアの品質を向上することは重要である．ソフトウェアのライフサイクルで考えれば，分析から実装で開発作業が終了ではなく，その後にテストなどの重要なフェーズが控えている.

本稿では，オブジェクト指向を用いてエラーの少ないソフトウェアを開発するための技術を明らかにする．我々は，オブジェクト抽出のための基本的なモデルとしてドメイン参照モデル[1]という概念を既に提案しており，適用事例を通じてその有効性を確認している[1][2][3]．ドメイン参照モデルとは，あるドメインのオブジェクトをモデル化するために参照する基本的なモデルである．本稿の目的は，

ITS(Intelligent Transport Systems)画像交換システム開発における，このドメイン参照モデルを用いた，ソフトウェア開発の早い段階でソフトウェアのエラーが発生する原因となるリスクを洗い出し，その結果としてテスト工程でのバグ曲線の立ち上がりを早め，かつバグ件数を減らすためのアプローチを示すことである．さらに，開発を通じての考察を行い，本アプローチの効果を明らかにする．

2. オブジェクト指向開発の現状

オブジェクト指向開発を進めるためのガイドとして，これまでに OMT 法[4]，Booch 法[5]などの方法論が提案され，これらの方法論を統合した RUP(Rational Unified Process)[6]がある．現在のオブジェクト指向開発では，UML(Unified Modeling Language)[7]と呼ばれるモデリング言語を用いて，RUP などの開発方法論をベースに進めることが多い．しかし，いずれの方法論においても，分析･設計フェーズの記述が中心になっている．

しかし，実際の開発では，再利用性を考慮することも重要であるが，エラーの少ないソフトウェアを実現するために品質を考慮することも重要である．従来のオブジェクト指向方法論では，この観点は充分明らかにされていない．

3. ドメイン参照モデル

まず，ドメイン参照モデルとは何かついて示す．3.1節でドメインおよびドメイン参照モデルの定義を明らかにし，3.2節にドメイン参照モデルの構築方法を示し，3.3節に例として監視ドメイン向けの PERS モデルを説明する．

3.1. ドメイン参照モデルの定義

ドメインとは，一言で言えば類似したシステムの集合を指す．我々は，事務処理システムなどの情報分野，監視システムなどの工業分野に大きく分け，工業分野では図 1に示すように制御，監視，組み込みといったドメインを定義している．さらに，監視ドメインには複数の類似したシステムが存在し，例えば図 1に示すように，化学プラント監視システム，駅務監視システムなどが含まれる．

ドメインごとで基本的なソフトウェア構造を共有するためにドメインのオブジェクトを抽出するための基本的なモデルを定義し，これをドメイン参照モデルと呼ぶ[1]．工業分野の三つのドメインごとに異なるドメイン参照モデルを提案している[1][2][3].

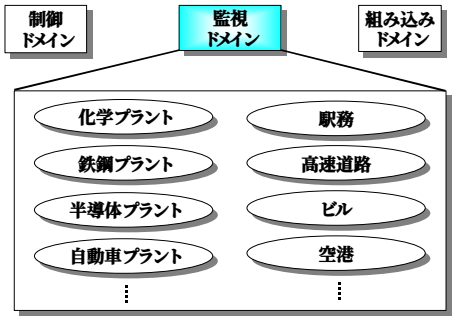

図1 工業分野のドメイン

3.2. ドメイン参照モデルの構築方法

既存のドメイン参照モデルに適切なモデルがなければ，次の手順で新たに構築する．これらの作業は，オブジェクト指向分析を始める前の準備段階であり，システム分析の作業の一環として行う．

[**ステップ 1：初期オブジェクトの抽出**]

ユースケースを実行するためのオブジェクトを抽出し，各オブジェクトの役割分担を決める．

[**ステップ 2：初期オブジェクトの分類**]

抽出したオブジェクトの役割に沿って，オブジェクトを分類する．オブジェクトの分類名を付ける．

[**ステップ 3：ドメイン参照モデルのモデル化**]

オブジェクト抽出と分類が固まったならば，システム構造をドメイン参照モデルとしてモデル化する．

3.3. ドメイン参照モデルの例

我々は，対象とするシステムが属するドメインに応じて，さまざまなドメイン参照モデルを整備している．例えば，監視ドメイン向けの PERS (Presentation-Entity-Relay-Service)モデル[1]，組み込みドメイン向けのデセセ（デバイス-制御-制約）モデル[2]，制御ドメイン向けの SMUGD (Service-Media-Unit-GUI-Data)モデル[3]がある．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（例えば，カメラ，ゲートなど）に対して遠隔の端末から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（数値データなど）を取得することで，監視業務を行うドメインである．監視ドメインは次の特徴を持ち，我々は図 2に示すモデルを提案している[1].

- 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相違から異なる場合がある．

Presentation：視覚的な情報を提供する．
Entity：ドメインのデータを表現する．
Relay：監視対象機器とデータを相互に通信する．
Service：システムのサービスを提供する．

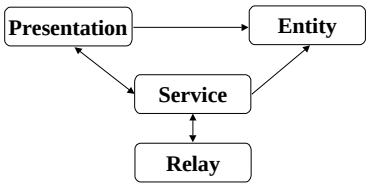

図2 PERS モデル

4. ドメイン参照モデルに基づく ITS 画像交換システム開発

ITS 画像交換システム開発における，エラーの少ないソフトウェアを開発するためのドメイン参照モデルに基づくオブジェクト指向開発プロセスについて示す．4.1節で対象システムについて述べ，4.2節でドメイン参照モデルに基づく品質向上アプローチについて示し，4.3節でドメイン参照モデルに基づくオブジェクト指向開発プロセスについて述べる．

4.1. 対象システム

ITS 画像交換システムは，クライアントの指示にしたがって，道路に設置されたカメラの動画像をクライアントに転送するためのシステムである．図 3 に，ITS 画像交換システムの一例を示す．本システムは，クライアント，画像交換サーバ，接続管理サーバの三つから構成される．また，各コンピュータノードは，アプリケーション部，通信ソフトウェア部，伝送部という三つのレイヤから構成される．

クライアント：カメラの動画像に対する画像交換サーバへの配信要求機能などを持つ．
画像交換サーバ：クライアントの要求を受信し，接続管理サーバへの画像配信指示機能などを持つ．
接続管理サーバ：画像交換サーバの配信指示に従い，クライアントへの画像配信機能などを持つ．

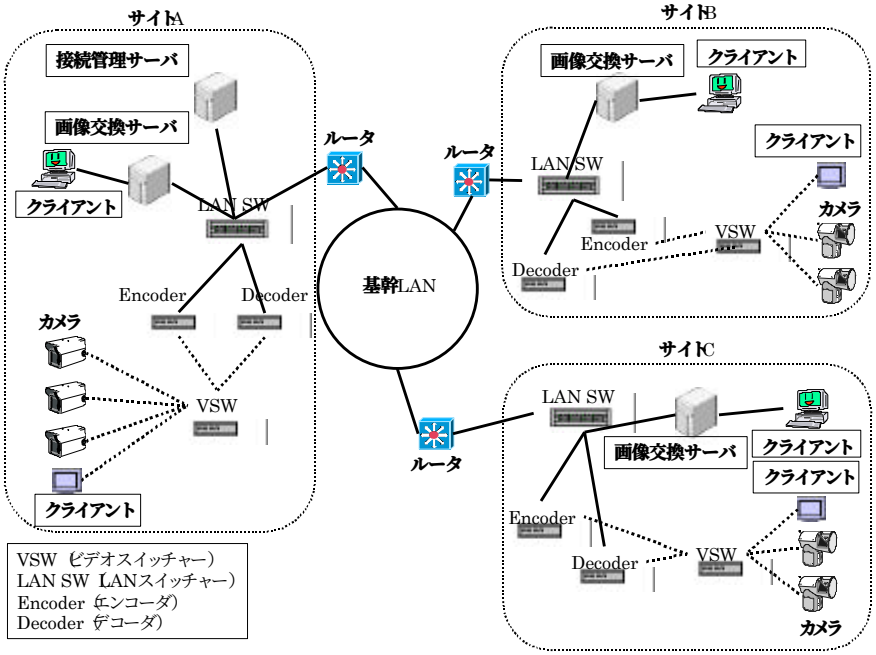

図3 ITS 画像交換システム

4.2. ドメイン参照モデルによる品質向上

ソフトウェア開発フェーズにおいてできるだけ早い段階で，リスク要因を洗い出しその施策について検討することは重要である．その結果として，バグ曲線の立ち上がりを早め，かつバグ件数を減らすことが可能である．そこで，我々はドメイン参照モデルに基づいて品質を作り込むアプローチを提案する．

4.2.1. 対象とする開発フェーズ

ソフトウェア開発のフェーズを「システム分析，オブジェクト指向分析，オブジェクト指向設計，オブジェクト指向実装，オブジェクト指向テスト」とし，品質の作り込みプロセスを定義する．

4.2.2. 対象とするリスク要因

ソフトウェアのエラーとは，ソフトウェアの中に障害を作り込む人間の動作である[8]．このエラーを発生させるリスク要因の中で，ソフトウェア構造に関わる要因を対象に品質の作り込みを行う．したがって，開発者の技術不足や開発者のコミュニケーションの問題などプロジェクトに関わる要因については対象としない．ソフトウェア構造に関わる要因とは，例えば次に示す点を指す．

- ユーザの要求が少しずつ変わるなどの仕様変更
- サブシステムなどのインタフェース変更
- 実行速度などの実装上制約に伴うチューニング

これらの影響を受けるソフトウェアの部位をホットスポットと呼ぶことにする．ホットスポットという言葉は，文献[9]によれば「再利用の観点から，ドメインの中で予測できないことに柔軟に対処しなければならない部分」と定義されている．本稿では，広い意味としてとらえ「ソフトウェア構造に関わる柔軟に対処しなければならない部分」と再定義する．

4.2.3. ドメイン参照モデルに基づく理由

ドメイン参照モデルを活用する最大の理由は，オブジェクトのカテゴリ（似た位置づけを持つオブジェクト群の特徴を表した分類の観点）を設定できることにある．ドメイン参照モデルに基づく有効性について，従来のソフトウェア開発フェーズにおけるホットスポット分析の問題点を示し，ドメイン参照モデルとホットスポット分析の関係を明確にする．

従来のソフトウェア開発でホットスポット分析を行う趣旨は，基本的には再利用性を向上させるためであり，ソフトウェアの品質を向上させることについて焦点を置いているわけではない．また，ホットスポット分析においてリスク要因に対応するオブジェクトを洗い出したとしても，二つの問題点がある．

一つ目は，ホットスポット分析実施上の問題点である．ホットスポット分析においてリスク要因に対応するオブジェクトを洗い出すことは可能である．しかし，リスク要因は基本的には該当するオブジェクト固有の問題としてとらえることになる．似た位置づけを持つオブジェクトがあったとしても，リスク要因における似た位置づけを持つ他のオブジェクトへの影響を見過ごす可能性があった．

二つ目は，ホットスポット分析実施による品質確保の問題点である．開発を進める中でリスクアセスメントのためにホットスポット分析を何度か行うことが考えられる．しかし，リスク要因に該当するオブジェクトのみに焦点を当てているため，開発を進める中での時間的変化に対してシステム全体としての傾向を捉えることが困難であった．

上記の問題を解決するために，似た位置づけを持つオブジェクトを分類の観点として捉えることが重要である．我々は，似た位置づけを持つオブジェクト群の分類の観点をドメイン参照モデルとして定義している．そのため，ドメイン参照モデルをベースにホットスポット分析を進めることが有効と考える．

4.3. 開発プロセス

本システムにおけるドメイン参照モデルによる品質の作り込みのための開発プロセスを示す．ただし，ここでは通信ソフトウェアの開発について示す．

4.3.1. システム分析

システム分析では，ユースケース分析，ドメイン参照モデル構築，ホットスポット分析の三つの作業を実施した．

■ ユースケース分析

システムで提供するサービス（機能）を明らかにした．ユースケース分析の一例を図 4に示す．図 4は，「クライアントが画像交換サーバに接続する」というユースケースを表している．

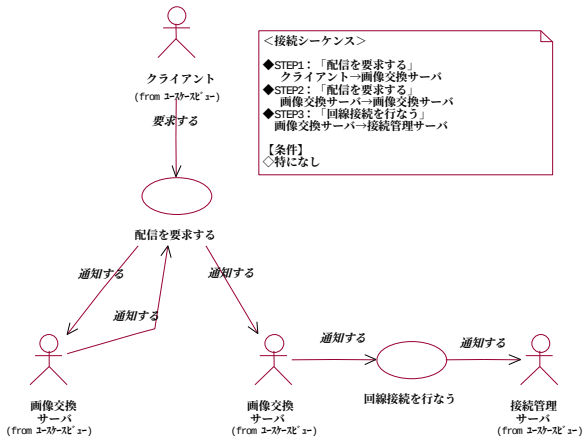

図4 ユースケースの一例

■ ドメイン参照モデル構築

ITS 画像交換システムは，画像交換サーバにとってみれば，クライアントや接続管理サーバとデータを相互に通信し，画像を交換するためのサービスを提供する．この特徴は，監視システムと似ている．そこで，3.3節で示した PERS モデルを ITS 画像交換システムに適用することを考えた．

■ ホットスポット分析（システム分析レベル）

PERS モデルに基づきホットスポット分析を行った．ホットスポット分析結果の一部を図 5に示す．図 5では，ユースケースに対する仕様変更などのリスク要因の影響を受けるホットスポットを分析した．影響の大きさによって 0〜3 までの重み付けを行い，PERS モデルの詳細な分類ごとに影響度の値を合計した．なお，通信ソフトウェアでは，GUI を持たないため Presentation モデルの結果はない．

図 5では，本来ならばリスク要因となる仕様変更そのものを記述するべきであるが，分かりやすさを考慮し仕様変更の対象となるユースケース名を記述している．例えば，「コネクションの確立を行う」というユースケースでは，「コネクションの確立方法が変更される」というサービス仕様の変更がリスク要因として考えられる．これに影響を受けるホットスポットは Service と Relay である．

3：影響大，2：影響中，1：影響小，0：影響なしあるいは限りなく小さい

ユースケース	Relay		Service			Entity			
	通信 IF	TCP/IP インタフェース	要求応答系サービス	状態遷移管理マネジャー	受信キュー管理マネジャー	受信キュー	DATEX ヘッダ	メッセージセット	データセット
コネクションの確立を行う	1	1	0	1	0	0	0	0	0
コネクションの開放を行う	1	1	0	1	0	0	0	0	0
ログイン要求を送信する	1	0	0	1	0	0	1	0	0
ログイン要求を受信する	1	0	0	1	0	0	1	0	0
ログイン応答を送信する	0	0	0	1	0	0	0	0	0
ログイン応答を受信する	0	0	0	1	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…
T1タイムアウトを通知する	0	0	1	1	1	0	1	1	1
T2タイムアウトを通知する	0	0	1	1	1	0	1	1	1
T3タイムアウトを通知する	0	0	1	1	1	0	1	1	1
T4タイムアウトを通知する	0	0	1	1	1	0	1	1	1
合計	22	2	12	26	4	4	14	12	12

図5 ホットスポット分析（システム分析レベル）

4.3.2. オブジェクト指向分析・設計

PERS モデルに基づきオブジェクトモデルを構築した．分析と設計フェーズを分けて開発を進めたが，これらのフェーズにおいてホットスポット分析で指摘したリスク要因の種類と数に大きな違いがなかったのでここでは併せて示す．ただし，リスク要因の詳細度の違いがわかるようにする．

■ オブジェクト分析・設計

PERS モデルに基づきオブジェクトを抽出し，クラス図を作成した．特に工夫した点は，Service モデルのオブジェクトとして，状態遷移管理マネージャオブジェクトと，実行するサービスオブジェクトを分離したことである．また，通信ソフトウェアの状態をオブジェクトとして実現した（State Pattern[10]の利用）．

■ ホットスポット分析（オブジェクト分析・設計レベル）

オブジェクト指向分析・設計フェーズにおいて，ホットスポット分析で指摘したリスク要因の種類と数に大きな違いがなかった．しかしながら，オブジェクト指向分析と設計では，オブジェクトモデルの詳細度が異なるので，その意味でのリスク要因の詳細度は異なっている．例えば，本ソフトウェアでは，複数クライアントのリクエストに対するサーバの実行に関するリスク要因がある．これについて，オブジェクト指向分析フェーズでは，「複数のクライアントからリクエストを受けたときのリクエストの管理とサービスの実行は状態遷移を用いることで仕様どおりに実行できるのか？」といったオブジェクトモデルそのものに対する指摘となる．一方，オブジェクト指向設計フェーズでは，「複数のクライアントからのリクエストに対するサービスの実行は状態遷移により判断するが，デザイン・パターンなどを適用した場合に性能面で不利にならないか？」といったオブジェクトモデルの実装上の指摘となる．

オブジェクト設計レベルでのホットスポット分析結果の一部を図 6に示す．図 6では，設計モデルに対して実装上でリスク要因となる項目を洗い出し，この項目ごとに PERS モデルに沿って影響を受ける部位を分析した．影響の大きさによって 0〜3 までの重み付けを行い，PERS モデルに基づくオブジェクト群ごとに影響度の値を合計した．なお，通信ソフトウェアでは，GUI を持たないため Presentation モデルの結果はない．

4.3.3. オブジェクト指向実装

C++言語を用いて実装し，綿密にコードレビューを実施した．コードレビューでは，「仕様としてのプログラミング」の観点と「言語としてのプログラミング」の観点を設定した．前者については，オブジェクト指向設計フェーズでのホットスポット分析で指摘されたリスク要因を参考に，コードレビュー

でのチェック項目を設定した．

■ コードレビュー

コードレビューは，ソースコードを実装した担当者以外の担当者約 4 名で行った．二つのグループに分かれ，一つのグループは「仕様としてのプログラミング」の観点で，もう一つのグループは「言語としてのプログラミング」の観点でレビューした．レビュー項目の例を以下に示す．

[システムとして]

PERS モデルのモデル要素ごとに注意すべき視点を設定した．全体で 80 項目程度のリスク要因を洗い出した．以下に一例を示す．

◆ Entity

・通信データが正しく生成されているか？

・通信データの初期化は正しくできているか？

◆ Relay

・リクエスト受信処理は正しく実装されているか？

・アプリケーションに対するインタフェースは正しく実装されているか？

◆ Service

・状態遷移は正しく記述されているか？

・エラー処理が正しく埋め込まれているか？

[言語として]

PERS モデル全体に関連する．全体で 30 項目程度のリスク要因を洗い出した．以下に一例を示す．

・ ポインタや変数などの初期化はされているか？

・ 基底クラスのデストラクタは virtual か？

・ パラメータ値渡しについて理解されているか？

・ スレッド関連の処理を正しく実装しているか？

4.3.4. オブジェクト指向テスト

コードレビューによる修正を行った後，通信ソフトウェアとアプリケーションの単体テスト・組合せテスト後，システムとしての接続テストを行った．

3：影響大，2：影響中，1：影響小，0：影響なしあるいは限りなく小さい

リスク要因	Relay							Service										Entity																		
	通信I/F	通信I/Fファクトリ	アナライザI/F	…	TCP/IP I/F	コンバータ	ソケット	状態遷移管理マネージャ	状態遷移クラス群	MS送受信サービス群	受信キュー管理	デコード	初期化	イベント	…	セレクト	ヘルスチェック	配信制御	配信状況	構成追加	構成情報	…	装置ID	配信OP	位置情報	関連情報	号機情報	装置管理者情報	配信状況	構成要素	障害情報	受信待ち	セッション情報群	固定情報群	…	受信キュー
MS/DDの再設計	1	1	0	…	0	0	0	1	0	0	0	0	0	0	…	0	0	0	0	0	0	…	1	1	1	0	1	1	0	0	0	0	0	1	…	0
EasyIFの変更	2	1	0	…	0	1	0	1	1	0	0	0	1	0	…	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0	0	0	…	0
ASN実装の修正大	0	0	1	…	0	2	0	0	0	0	0	2	0	0	…	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0	0	0	…	0
ASN部分自作	2	1	3	…	0	3	0	1	0	0	0	2	0	0	…	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0	0	0	…	1
ﾂｰﾙに依存する	1	0	1	…	0	1	0	1	1	0	0	2	0	0	…	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0	0	0	…	0
EasyIFの変更	1	0	0	…	0	0	0	1	1	0	0	0	0	1	…	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0	0	0	…	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
機関 ･組織ｺｰﾄﾞ関連修正	1	1	0	…	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	0	0	…	0	0	1	0	2	2	0	0	0	0	0	0	…	0
ﾄﾞﾒｲﾝ処理の実装追加	1	1	0	…	0	0	0	1	1	1	0	0	1	0	…	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0	0	0	…	0
ｺｰﾃﾞｯｸのｱﾗｰﾑ処理追加	1	0	0	…	0	0	0	1	0	1	0	1	0	0	…	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	1	0	0	0	…	0
認証処理の追加	1	1	0	…	0	0	0	1	1	1	0	0	1	1	…	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0	0	0	…	0
合計	39	28	19	…	4	20	3	44	23	28	8	24	11	10	…	3	3	5	4	4	4	…	5	9	7	2	7	8	2	3	8	2	4	8	…	9

図6 ホットスポット分析（オブジェクト設計レベル）

5. 考察

PERS モデル活用による有効性について，ホットスポット分析の効果，品質確保の効果という二つの観点で考察する．

5.1. ホットスポット分析の効果

ホットスポット分析効果を評価するために，オブジェクト指向設計フェーズでのホットスポット分析で指摘されたリスク要因に対応するバグが，実装・テストフェーズでどの程度含まれたかを評価する．

オブジェクト実装フェーズでの単体テスト終了後のコードレビューでのバグが約 340 件，組合せテストおよび総合テストでのバグが約 40 件で，全体で約 380 件のバグがあった．これをオブジェクト指向設計フェーズでのホットスポット分析で指摘されたリスク要因に対応するバグと，プログラミング上のケアレスミスによるバグとに分類した結果を図 7に示す．ホットスポット分析を実施することにより，全体の 3 割程度のバグについては事前にリスク要因として洗い出すことができた．また，約 90%のバグはコードレビュー段階で洗い出すことができた．これらのことから，ホットスポット分析を実施したことで，コードレビュー，あるいはテストフェーズで，明確にバグとなるリスク要因を意識し，テスト後半でのバグ件数を削減することができた．

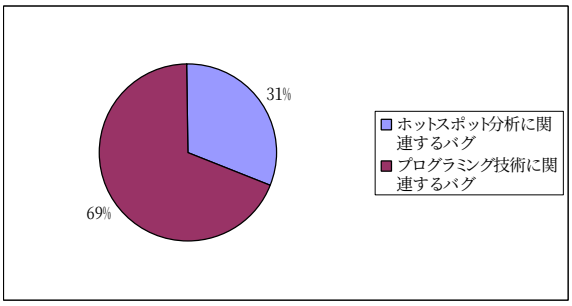

図7 ホットスポット分析の効果

5.2. 品質確保の効果

ホットスポット分析を実施することによる品質確保の効果について，バグ曲線を評価することで示す．

コードレビュー開始から組合せテストおよび総合テストまでのバグ曲線を図 8に示す．コードレビューでの指摘事項もバグ件数として数えた．試験日数の内訳は，コードレビューの期間は約 20 日間，組合せテストの期間は約 10 日間，総合テストの期間は約 10 日間である．

図 8を見ればわかるように，組合せテストを開始して以降（試験日数 20 日目以降）は，総合テストも含めてバグ発生件数は非常に少なく抑えることができた．まず，ソースコードレビューにより全部で約 380 件の指摘事項を洗い出した（試験日数 1～20 日目）．これらを修正した後に組合せテストを実施した．組合せテストで発覚したバグは約 30 件であった（試験日数 21～30 日目）．さらに，組合せテスト時のバグを修正した後に，総合テストを開始した（試験日数 31 日目以降）．総合テストでは，約 10 件程度のバグしか発見されなかった．以上の結果から，ホットスポット分析の実施，ホットスポット分析に基づいたコードレビューの実施により，バグの立ち上がりを早くし，テスト後半でのバグ件数を削減できた．

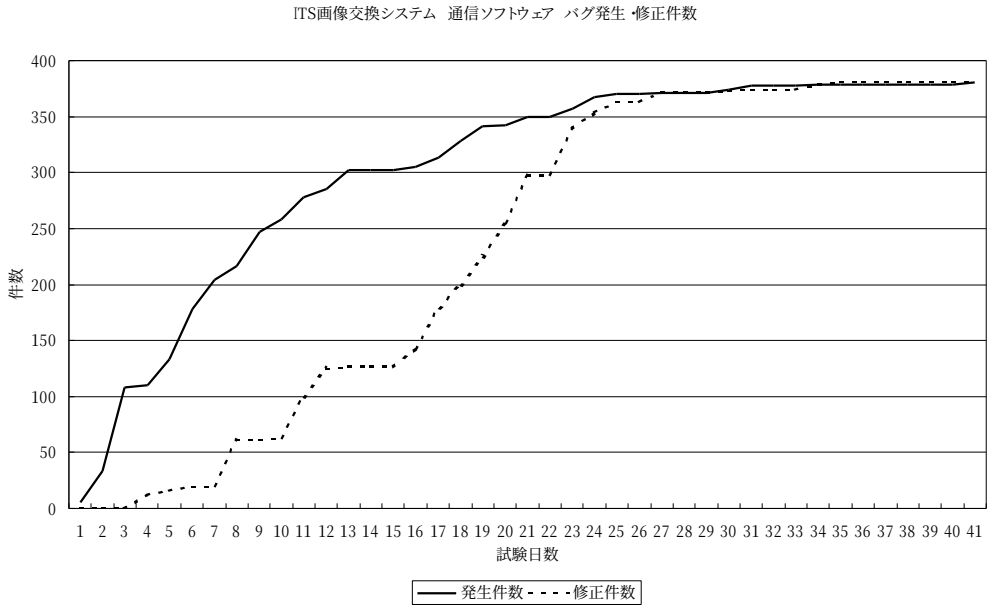

図8 バグ曲線

6. 関連研究

ソフトウェアの品質に関連する研究は数多く行われている．しかし，それらの多くはソフトウェアの品質特性（ソフトウェアの品質を決定する要因）を測定するための手法に関するものが多い．

オブジェクト指向を用いたシステム向けのメトリクスの研究がいくつかされている．例えば，オブジェクト指向で開発されたソフトウェアの複雑さのためのメトリクスについては，既存のメトリクスである Halstead のメトリクス[11]と McCabe のメトリクス[12]をオブジェクト指向用に拡張した提案[13]がされている．また，オブジェクト指向設計のためのメトリクスについては，カプセル化，継承，ポリモーフィズムなどのオブジェクト指向の特徴を考慮した MOOD と呼ばれる評価方法[14]が提案されている．また，オブジェクト指向テストに関する研究も行われている．オブジェクト指向テスト技法として，コストを考慮に入れた最適なリスクのレベルを

設定し，そのためのテスト技法，環境開発，プロジェクト管理に関する提案がされている[15]．これらはいずれも，ソフトウェア開発の下流工程（オブジェクト指向設計，オブジェクト指向テスト）で活用される技術である．

一方，我々の手法は，ドメイン参照モデルを活用しソフトウェア開発の上流工程（システム分析，オブジェクト指向分析，オブジェクト指向設計）で品質を作り込むアプローチである．したがって，我々のアプローチと上記のいくつかのメトリクスやテスト技法は，対象とする開発工程が異なるため相互に補完可能である．

7. おわりに

本稿では，ITS 画像交換システム開発における，ドメイン参照モデルに基づく品質向上アプローチを示した．また，開発を通じての考察を行い，本アプローチの有効性を明らかにした．

まず，同一のドメインで共通に利用するオブジェクト抽出のためのモデルとして提案しているドメイン参照モデルに基づき，ソフトウェア開発の早い段階から仕様変更などのリスク要因となるホットスポットを洗い出すアプローチを提案し，これに基づく開発プロセスを示した．さらに，適用事例を通じて，開発の早い段階でホットスポット分析を実施し，開発フェーズが移行することでホットスポットが変化するかどうかを常にチェックし，ドメイン参照モデルに基づきホットスポットの分布を監視することにより，ソフトウェアの品質向上に非常に貢献することが明らかになった．また，バグの立ち上がりを早め，テスト後半でのバグ件数を削減する効果があることがわかった．オブジェクト指向による開発では，再利用性が重視されがちであるが，実際の開発では品質の問題も同様に重要である．

今後は，次の三項目について検討することが課題である．第一に，同種のシステムを多くのユーザに展開するリピート開発を通じての本アプローチによる効果を評価することである．第二に，他のドメインのシステム開発に本アプローチを適用し，その効果を確認することである．第三に，本アプローチでは，ソフトウェア構造に関わる要因を対象とするリスク要因としたが，開発者の技術不足や開発者同士のコミュニケーションの問題などプロジェクトに関わる要因を含めた開発プロセスの検討が必要である．

参考文献

[1] 早瀬健夫，池田信之 他：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，電気学会論文誌 C（電子・情報・システム部門誌），Vol.121-C，No.3，pp.642—652 (2001).

[2] 池田信之，早瀬健夫 他：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），Vol121-C，No.6，pp.1049—1058 (2001).

[3] 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).

[4] Rumbaugh, J. et al.: Object-Oriented Modeling & Design, Prentice Hall (1991).

[5] Booch, G.: Object-Oriented Analysis and Design with Applications. Second Edition, Benjamin Cummings Publishing Company (1994).

[6] 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).

[7] Booch, G.: The Unified Modeling Language User Guide, Addison-Wesley (1998).

[8] 東 基衛：ソフトウェア品質評価ガイドブック，日本規格協会，1994.

[9] Pree, W.: Design Patterns for Object-Oriented Software Development, Addison-Wesley, 1994.

[10] Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns, Addison-Wesley (1994).

[11] Halstead, M.: Elements of Software Science, Elsevier North-Holland, New York (1977).

[12] Thomas J.: A Complexity Measure, IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, pp.308—320 (1976).

[13] Coppick, J., Cheatham, T.: Software Metrics for Object-Oriented Systems, Proceedings of the 1992 ACM annual conference on Communications, pp.317—322 (1992).

[14] Harrison, R. et al.: An Evaluation of the MOOD Set of Object-Oriented Software Metrics, IEEE Transactions on Software Engineering, Vol. 24, No. 6 (1998).

[15] Siegel, S.: Object-Oriented Software Testing: A Hierarchical Approach, John Wiley & Sons (1996).