4L-5

モデルとパターンに基づくソフトウェア変更分析方法論

田中 光† 青山 幹雄‡

南山大学 大学院 数理情報研究科† 南山大学 数理情報学部 情報通信学科‡

1. はじめに

近年のソフトウェア開発は，既存システムへの機能追加・変更が主流である．ユーザ要求は複雑化する一方で開発期間は短縮している．そのため迅速に追加・変更できる技術が必要である．

本稿では，実際のソフトウェアの変更作業と内容を分析し，処理のパターン化と変更内容のモデル化，ならびに，モデルに基づく変更分析方法を提案する．実務の分析から，提案方法が開発者のスキルを補い，品質向上に有効であることを示す．

2. ソフトウェア変更開発の問題分析

2.1. ソフトウェア変更プロセスの分析

ある業務アプリケーションシステムの変更における工数を収集し，工程別工数配分を分析した結果を図 1 に示す．変更要求に対する設計工程の割合が最も多いので，ボトルネックになっている[2]．また，「分業開発」で効果をあげるためには次に割合を占めるテスト工程で，設計工程の改善方法が再利用できることが望ましい．

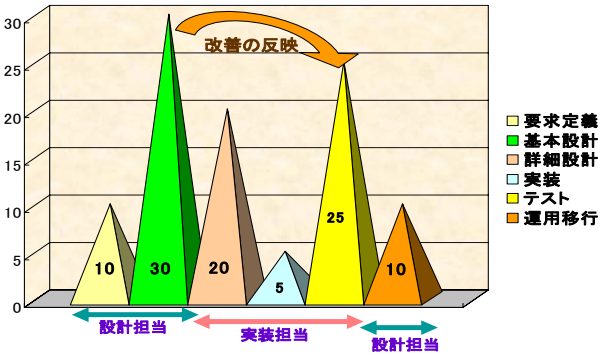

図 1 システム変更における工程別工数配分

2.2. ソフトウェア変更設計の問題分析

上記のシステム変更の設計工程における問題を分析し，図 2 の特性要因図で示す．

問題を四つの視点より分析した．**(1)ユーザの要求：**開発期間の短縮や変更の頻度．**(2)設計：**ドキュメント不足や調査不足．**(3)実装・テスト：**修正箇所の不明瞭やテスト不足．**(4)リソース：**人員不足やコミュニケーション不足．

A Software Change Analysis Methodology Based on Change Models and Patterns

†Hikaru Tanaka, ‡Mikio Aoyama

†Graduate School of Mathematical Sciences and Information Engineering, ‡Faculty of Mathematical Sciences and Information Engineering, Nanzan University

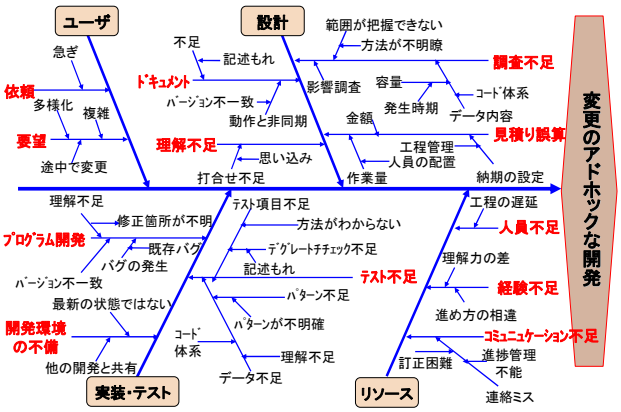

図 2 システム変更における問題点の分析

以上の原因より，アドホックな開発[1]を余儀なくされている．

3. ソフトウェア変更分析方法論

3.1. 変更要求に対するソフトウェア変更の分析

ユーザの変更要求には改善などの「内部要因」と法律改正などによる「外部要因」に分類できる．どちらも，ユーザの目に見える画面と帳票を中心に定義される．設計者はユーザには見えない内部処理の調査，分析を行い，機能追加・変更の内容を明らかにしなければならない．

3.2. 変更の分析方法

変更分析方法として，処理のパターン化とパターンに基づく変更内容の分析方法を提案する．処理をパターン化することで，変更要求に迅速に対応できる．さらに，提案する変更分析方法により変更内容を効率的に分析できる．

3.3. 処理のパターン化

業務処理の要素を「入力・抽出・編集・更新・帳票」に分類した．さらに，これを組み合わせた処理を 6 つのパターンにまとめた(図 3)．

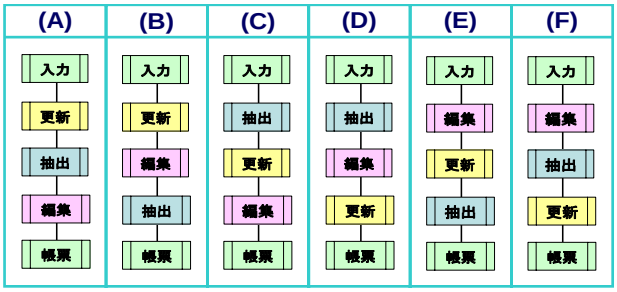

図 3 業務処理の流れのパターン

要求機能により処理順が異なるので異なるパターンとなる．処理の繰り返しもあるが，処理全体の流れは，基本的にこのパターンで表現できる．

3.4. モデルに基づく変更内容の分析方法

画面と帳票から変更内容を分析するために，変更内容をモデル化した．それぞれ，画面駆動モデル，帳票駆動モデルと呼ぶ．画面駆動モデルを 図 4 に示す．MVC(Model-View-Controller)アーキテクチャに基づき変更の構造をモデル化した．

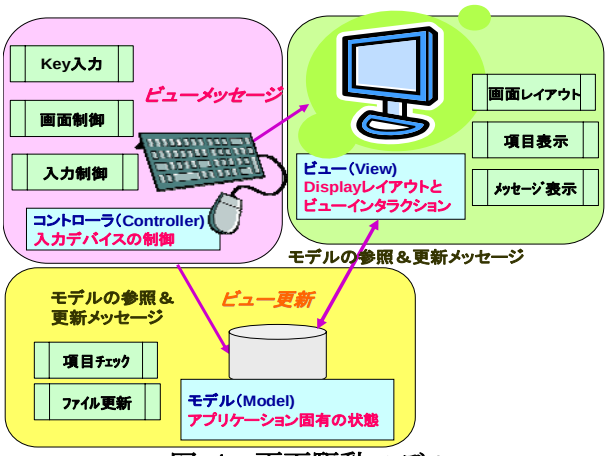

図 4　画面駆動モデル

3.5. 画面駆動モデルに基づく変更分析方法

画面駆動モデルに基づき，変更箇所を分析する方法を示す．**(1)画面レイアウト**：項目の追加変更の有無．**(2)key 入力**：key となる項目の分析．**(3)画面制御**：画面動作の分析．**(4)入力制御**：入力動作確認．**(5)項目表示**：表示項目の分析．**(6)項目チェック**：入力データの内部チェックの分析．**(7)メッセージ表示**：メッセージの確認．

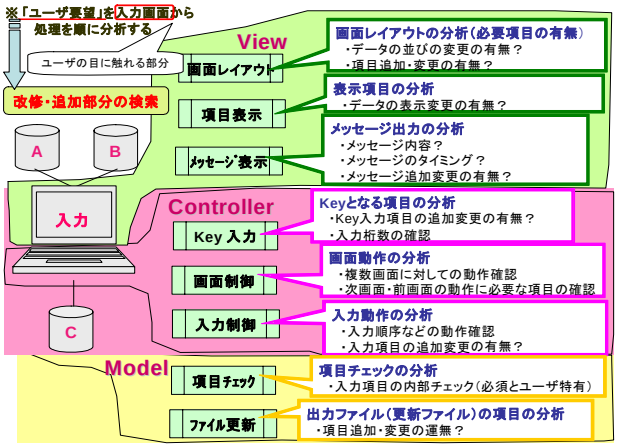

図 5　画面駆動型分析方法

3.6. 帳票駆動モデルに基づく変更分析方法

帳票駆動モデルに基づき，3.3 節の処理パターン(A)を用いた変更箇所の分析方法の例を示す．帳票駆動型モデルでは処理の流れの逆順に分析を行う．**(1)帳票レイアウト**：見出し，明細の変更追加の有無．**(2)ソート・サマリー**：プログラム外部での処理の分析として，データの並びや集約の変更の有無．**(3)入力ファイル**：必要項目に対し，ファイルレイアウトの変更の有無や他ジョブでの影響分析．**(4)編集・抽出・更新**：条件変更の有無，必要項目が網羅されているかの分析．**(5)入力画面**：必須項目の入力の確認や参照ファイル追加の有無．

4.提案方法の評価

(1)処理のパターン化の効果：67 件の変更事例とパターンとの対応を 図 6 に示す．パターン化したことで変更の全体像の把握が容易になり，変更箇所の分析が迅速に行えるようになった．

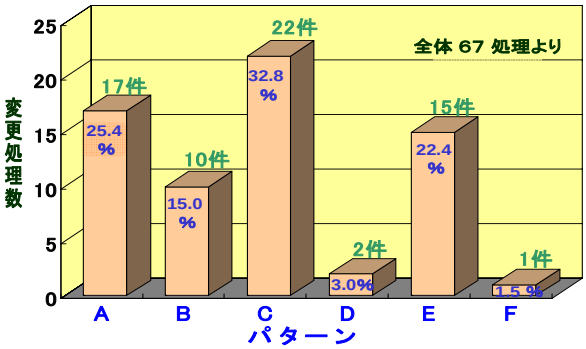

図 6　パターン別の変更処理数

(2)変更分析方法の効果：提案方法に基づき各処理において分析すべき内容を記述したチェックシートを作成した．これにより，一貫した手順に沿った分析が可能となり，経験の浅い技術者でも変更分析の手順が明確になり，変更漏れが減少した．実際の開発で発生した 18 件のバグを分析した結果，その 2/3 以上に本提案方法が適用可能であり，バグを未然に防ぐことができたと推察できる．

本方法を適用することにより，スキル不足で起こり得る分析の漏れや見落としが減り，設計が円滑に行われるようになった．さらに，変更分析の時間短縮効果も見られた．また，開発途中のバグ発生による後戻り作業が軽減され，設計プロセスの統一化が図られた．

5.まとめ

ソフトウェアの変更における生産性と品質の向上を目的とし，開発プロセスと変更内容を分析した．分析に基づき，処理の流れのパターン化とモデルに基づく変更分析方法論を提案し，実例に基づき評価した．開発者のスキルによらず，変更箇所の漏れのない統一された分析が可能となった．

参考文献

[1] 萱嶋 志門ほか, ソフトウェア設計方法論の開発と適用, 東芝レビュー, Vol. 61 No. 1, Jan. 2006, pp. 20-25.

[2] 渡辺 幸三, 業務システムのための上流工程入門, 日本実業出版, 2003.