南山大学 大学院 数理情報研究科 2006 年度修士論文 要旨集

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

M2005MM034 田中 光

指導教員 青山 幹雄

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．ユーザ要求が多様化し，ソフトウェアが大規模で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，バグを未然に防ぐ品質向上に有効であることを示す[3]．

2. ソフトウェア変更開発の問題分析

2.1. ソフトウェア開発のプロセス

システム開発はユーザの要求から始まる．一般的に基本設計までが上流工程と呼ばれるが，詳細設計まで含まれる事もある．通常，開発は設計担当と実装担当に分業できる．

2.2. 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プロセス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プロセスで，設計プロセスの改善方法が再利用できることが望ましい．

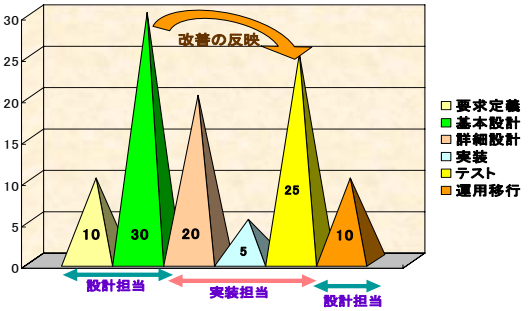

図 1 システム変更における工程別工数配分

2.3. 基本設計プロセスにおける作業分析

システムの基盤となる「基本設計」は次の手順で作業を行う．また，作業内容の割合を図 2 に示す．1)開発内容が新規追加で行うべきか修正で行うかの判断．2)変更（追加）部分の明確化，データ内容の確認，他のシステムの調査などの影響分析．3)開発方法の決定．4)基本設計書の作成．5)工程管理の考案（詳細設計以降のスケジュール）．6)基本設計段階のレビュー．

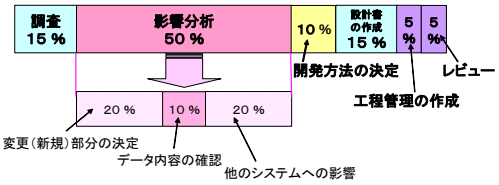

図 2 「基本設計」の作業時間の割合

2.4. 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 3 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．

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．

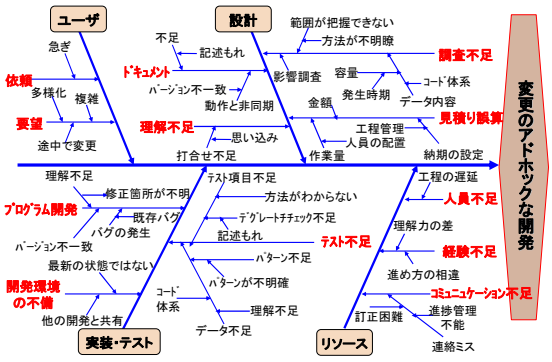

図 3 システム変更における問題点の分析

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

本研究では，業務処理ソフトウェアの構成を「要素」と「フロー」の2つに分類し，パターン化する．

データを入力して帳票が出力される一連のシステムの流れ部分を「フローパターン」と定義し，それを構成する各々の処理部分を「要素パターン」として定義する(図 4)．

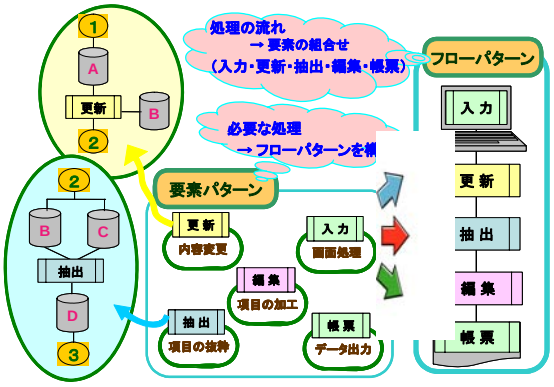

図 4 パターン化の定義

業務処理ソフトウェアの構成要素となる主要な処理を抽出し「入力」「更新」「抽出」「編集」「帳票」の5つに分類して「要素パターン」として定義した．

「要素パターン」を組合せて，データ入力より帳票が出力されるまでの処理の流れをパターン化し，図 5 に示す「フローパターン」とした．

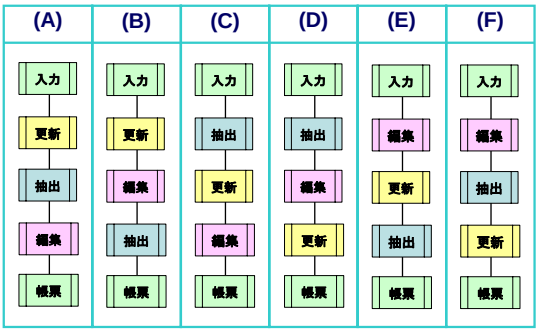

図 5 フローパターン

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分析方法論

設計プロセスにおけるモデル定義に当り，どのような視点で手順をモデル化するかが鍵となる．本研究では，業務処理ソフトウェアのパターン化の定義とユーザ要求の関係の整理という視点から設計プロセスにおいて手順をモデル化した．ユーザ要求が画面と帳票から発生するため，次の二つのモデルを考える．さらにモデルを用いて，変更分析を行う手法を示し，ドキュメント化をおこなう．

以上の手順をまとめて図 6 に示す．モデルに基づく変更分析方法論を提案する．

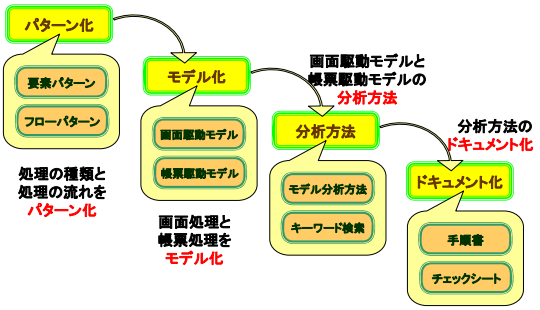

図 6 モデルに基づく変更分析方法論

3.5. モデル化

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 7 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．

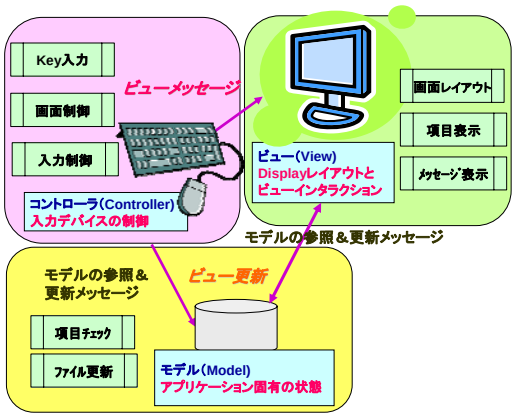

図 7 画面駆動モデル

3.6. 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図 8 に示す．

【View 部分】(1)画面レイアウト:項目の追加変更の有無．**(2) 項目表示:**表示項目の分析．**(3)メッセージ表示:**メッセージの確認．

【Controller 部分】(4) key 入力:key となる項目の分析．**(5) 画面制御:**画面動作の分析．**(6) 入力制御:**入力動作確認．

【Model 部分】(7) 項目チェック:入力データの内部チェックの分析. **(8)ファイル出力・更新:**出力項目の分析.

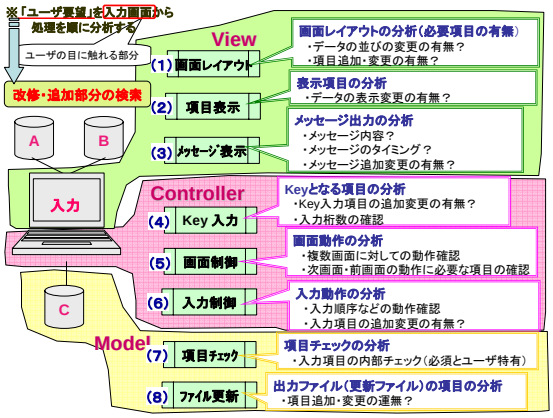

図 8 画面駆動モデルに基づく分析方法

3.7. 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図 9 に示す. 帳票駆動型モデルでは処理の流れの逆順に追加・変更の有無を分析する.

ステップ 1 (帳票部分):レイアウト, 項目, 合計, 計算方法, ブレイク, 内容表示, 見出し, 明細, 合計, 外部処理

ステップ 2 (編集部分):計算方法, 変換, 移送項目, コード体系, 名称, 出力, 外部処理

ステップ 3 (抽出部分):条件, 内容, コード体系, 出力, 外部処理

ステップ 4 (更新部分):更新項目, 条件, 内容, 更新対象, 計算方法, 出力

ステップ 5 (入力部分):「画面駆動モデル」の分析内容

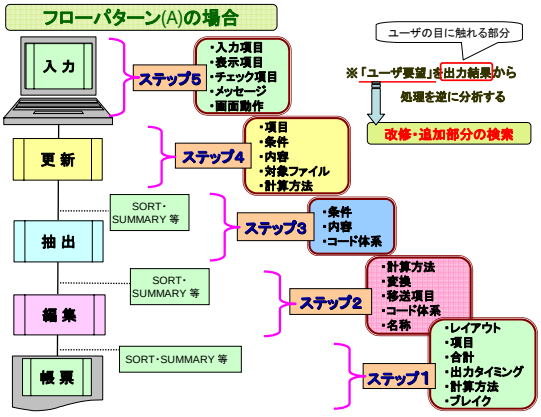

図 9 帳票駆動モデルに基づく分析方法

3.8. 修正項目の影響範囲の調査方法

設計者の思い込みで分析範囲が限定されないように, 分析範囲を適切に見極めるように, 可能な限り機械的に範囲を特定し, 分析作業の効率化を図る必要がある. まず, 分析の前段階として, 次に述べる調査を行ってから前節で定義した分析方法に着手するのが望ましい. 手法としては, 使用するマシンや言語等によって異なるが「キーワード検索機能」のツールを用い, 追加・修正となる項目の使用箇所を検索することである. ツールの使用例を 図 10 に示す.

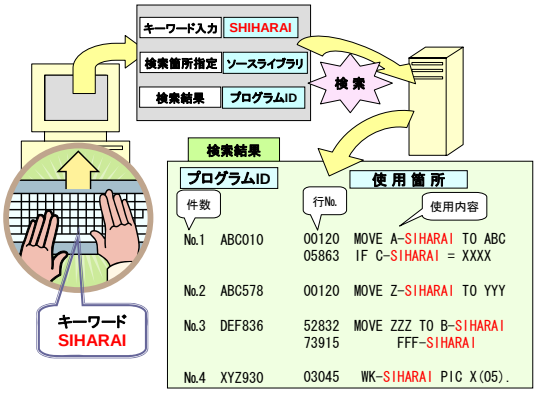

図 10 ツールを用いた「キーワード」の分析例

3.9. 変更・分析手順のドキュメント化

モデルに基づく分析方法を効率的に活用するためにドキュメント化する必要がある. 定義した分析方法に従い, 例を記入した手順のドキュメントを 表 1 に示す.

表 1 分析手順のドキュメント

赤文字 → 例

No.	作業手順		内 容							備考欄
1	修正内容		抽出条件の追加(支払いコード=ナンザン を追加							
2	対象項目の特定		SHIHARAI							
3	対象項目の検索(対象プログラム検索)		ABC010 ABC578 DEF836 XYZ930							XYZ930 は修正なし
4	ユーザ依頼		画面	帳票					その他	
5	処理の流れ		画面	A	B	C	D	E	F	その他
			Key入力	入力	入力	入力	入力	入力	入力	
			画面制御	更新	更新	抽出	抽出	編集	編集	
			入力制御	抽出	編集	更新	編集	更新	抽出	
			項目チェック	編集	抽出	編集	更新	抽出	更新	
			更新	帳票	帳票	帳票	帳票	帳票	帳票	
6	影響箇所		対象箇所	詳 細						
		画面	レイアウト							
			項目表示							
			ﾒｯｾｰｼﾞ表示							
			Key入力							
			画面制御							
			入力制御							
			項目チェック							
			更新							
		帳票	抽出	SHIHARAI = NANZAN を抽出の判定条件に追加						＊1
			更新							修正なし
			編集	計算方法を変更(別紙参照)						抽出されたデータが全件出力される
			帳票	＊1						
		その他	SUMMRY	SHIHARAI = NANZAN の金額集計を追加						
7	開発方法(確認方法)		プログラム修正・デグレートチェック・外部処理の追加							
8	その他									

プログラムID記入

○印で囲う

変更箇所を○印で囲う

赤色一例

ドキュメント化は, 修正内容に沿って分析手順状態が一目瞭然となる. さらに, 修正内容を熟知していない第3者がレビューを行う場合や, テストプロセス担当者にとって, 修正内容や処理の流れが把握しやすくなる. また, 設計者がどのような手順で分析を行ったかを伺うことが可能となる.

3.10. 変更・分析のチェックシート

分析時に提案する方法をチェックシートとして活用できるように，各要素パターン別に分析内容の詳細をドキュメント化する．要素パターン毎の分析内容と確認事項を詳細に明記した．「抽出」を例としたチェックシートを表 2 に示す．

チェックシートによって，各処理で分析すべき内容が一目瞭然となる．設計者が分析時に，何から着手すべきか迷うことが無くなり効率的に分析ができる．また，確認すべき内容の漏れがなくなり分析の信頼性が向上する．

表 2 分析項目のチェックシート

処理	内容	確認事項	チェック	
			済	要修正
抽出	条件	抽出条件の変更		
		抽出方法		
	内容	抽出内容の変更		
	コード体系	コードの使用		
		他のコードとのリンク		
		コード内容		
	出力	項目の追加・変更		
		出力方法		
		タイミング		
		他ファイルとの整合性		
	外部処理	ソート・サマリーの変更		
	予備			

4. パターンとモデル化の適用と評価

4.1. 処理のパターン化の評価

本稿で提案した処理のパターン化を実システムの 67 件の変更事例に反映した結果，フローパターン毎の適用効果が確認できた(図 11)．

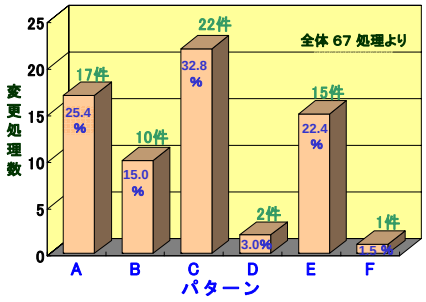

図 11 フローパターン別の変更処理数の分布

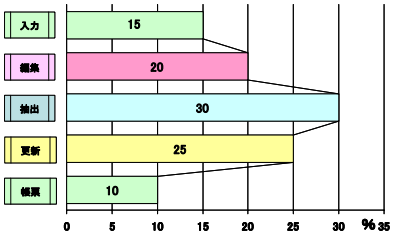

図 12 要素パターンの変更箇所の分布

パターン化したことで，処理の要素と流れが明確となり，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．さらに，開発者本人だけではなく，処理を理解していない第三者に対する内容説明が容易となった．さらに，要素パターンの出現分布を図 12 に示す．ユーザ要求に対してシステムの変更・追加箇所が「抽出→更新→編集→入力→帳票」の順に頻度が高いことがわかった．すべてのシステム変更に当てあまるとは断定できないが，設計プロセスの目安となった．

4.2. 変更分析方法のモデル化の評価

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．一例を表 3 に示す．

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．

表 3 パターンとモデルの適用効果

パターン	モデル	適用件数				パターン要素					適用内容
フロー		○	△	効果計	×	入力	更新	抽出	編集	帳票	(要因)
A	帳票	2	0	2	1		*	*	*		コード体系の分析，他ジョブへの影響分析(コーディングミス)
B	帳票	2	0	2	0		*			*	ファイル項目の確認 名称取得方法の確認
C	帳票	3	1	4	0			*	*	*	合計の分析，抽出方法の確認，変換方法の分析
D	帳票	1	0	1	0			*			フローパターンにて分析
E	帳票	1	1	2	1		*	*	*		キーワード検索，更新項目の確認 (データ量の分析)
F	帳票	1		1	0						外部処理の確認 レイアウトの桁位置確認
他	画面	4	0	4	0	*					項目チェックの分析，エラー処理の対応，出力先確認，キー項目分析
合計		14	2	16	2						

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，玉木 祐二，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, 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.

[3] 田中 光，青山 幹雄，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，情報処理学会第69回全国大会論文集，No. 4L-5, Mar. 2007(発表予定)．