UMLで記述された設計仕様書を対象としたレビュー手法CBRとPBRの比較評価実験

松川文一[†] Giedre Sabaliauskaite[††]
楠本真二[†] 井上克郎[†]

ソフトウェアレビューの目的は, ソフトウェアの開発過程で生成されるプロダクトを検査し, バグを検出, 除去することにある. 開発の初期の段階でバグを取り除くことによって, その後の開発段階でのコストを抑え, ソフトウェアの品質と生産性がより高いものとなる.

現在までに数多くのレビュー手法が提案されている. 最近では, Perspective-based Reading(PBR) 手法が注目されている. Laitenberger らは, オブジェクト指向開発で標準的に用いられている仕様記述言語 UnifiedModeling Language(UML) で記述された設計仕様書を対象として, PBR を, 従来手法の一つである Checklist-based Reading(CBR) 手法との間で有効性の比較実験を行った. しかし, この実験で用いられた UML 図は 2 種類であり, より様々な UML 図に対して比較実験を行う必要性が指摘されている.

本論文では, 5 種類の UML 図を用いた設計仕様書に対し, CBR と PBR の有効性を比較するための追証実験を実施した. 実験では情報科学科の 3 年生 59 人を被験者とし, レビュー対象として 2 つのシステムの UML 図を用いた. 被験者はあらかじめ複数のバグを埋め込んだ UML 図を, 指定されたレビュー手法でチェックし, 検出したバグ, 検出時刻等を報告した. 収集データを用いて, バグの検出率, 検出時間に関して, PBR を使用した被験者と CBR を使用した被験者の間で比較を行った. 結果として, 構文的なバグの検出に関しては CBR に, 意味的なバグ, 他の図と関連したバグの検出に関しては PBR に有効性が認められた.

Experimental comparison of Checklist-based reading and Perspective-based reading for UML Design Documents

Fumikazu Matsukawa ,[†] Giedre Sabaliauskaite ,[††]
Shinji Kusumoto [†] and Katsuro Inoue[†]

The purpose of software review is to detect and fix the defects in software products. By fixing the detected defects in earlier phases, we can reduce software development cost, and improve the quality and productivity of software.

Many review methods have been already proposed. Recently, Perspective-based Reading (PBR) have become to be noticed. O. Laitenberger et al. compared PBR and Checklist-based Reading (CBR) for review activities in design documents written in Unified Modeling Language (UML). However, since their experiment used only two types of UML diagram, it is pointed out that the other types of UML diagram should be considered.

In this paper, we performed experimental comparison between PBR and CBR for defect detection in five types of UML diagram. The subjects were fifty nine students in the Department of Informatics and Mathematical Science, Osaka University, and the objects were UML diagrams of two system. Students were divided into two groups, one group used CBR method and another group used PBR method. They reviewed UML diagrams in which some defects have been previously injected using CBR or PBR method. We collected the information about detected defects and their detection time. After the experiment, we compared two methods with respect to defect detection rate and detection time. The experimental results show that CBR is more effective in detecting syntax defects than PBR, and PBR is more effective in detecting semantic defects and the defects related to consistency between several UML diagrams.

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†† 大阪大学大学院基礎工学研究科
Graduate School of Engineering Science, Osaka University

1. はじめに

ソフトウェアの品質向上に有効な手段の 1 つとして, ソフトウェアレビューがある. ソフトウェアレビューとはソフトウェアの開発工程において作成されたプロダクト (設計ドキュメント, プログラムコードなど) の内容について, プロダクト作成者と他の開発者の間で議論し, 最終的にプロダクト内にバグが残らないようにする作業のことである[11]. ソフトウェアのプロダクトに対するレビューによって開発プロセスの初期の段階でバグを除去すれば, ソフトウェアの開発コストを削減することができ, その結果としてソフトウェアの品質と生産性を向上させることができる[11].

ソフトウェアレビューは一般的にいくつかの作業 (計画, バグ検出, 会議 (バグ収集), バグ修正等) から成る[8]. そのうちの 1 つであるバグ検出において, これまでにさまざまな手法が提案されている. 最も一般的な手法として, Ad hoc 手法と Checklist-based Reading(CBR) がある. Ad hoc 手法は, レビュー者に対して特に何の指針も無しでレビューを行う. それゆえこの方式ではレビュー者の経験や知識に依存する部分が大きい[7]. CBR はプロダクトと共にいくつかのチェック項目が記されたチェックリストを用いて, そのチェック項目を yes/no 形式でチェックしながらレビューを進めていくという手法[5] であり, バグ検出の効率を上げることができる. さらに, Basili によって Scenario-based Reading(SBR) が提案されている[2]. この手法は, シナリオと呼ばれる, レビューの進め方や着目すべき情報等が記されたものを用いる[7]. SBR にはさらに様々な形式があり, その中の 1 つである Perspective-based Reading(PBR) は, プロダクトに関わるいくつかの立場による視点からレビューを行うという手法である[1),9)].

また近年, オブジェクト指向ソフトウェア開発が主流となり, その開発に UML(Unified Modeling Language)[4] が用いられていることが多い. UML は Booch らによって開発されたソフトウェア設計開発言語であり, 様々な種類の図を用いてソフトウェアの体系を視覚的に表現することができる.

Laitenberger らはこの UML を用いた設計仕様書に対し CBR と PBR を用いてレビュー実験を行い, 両手法の比較評価を行った. この実験では, バグの平均検出率 (CBR:43%, PBR:58%), バグ 1 個当りの平均検出コスト (CBR:85(分/バグ), PBR:43(分/バグ)) でそれぞれ PBR が有効であるという結果が報告されている[7]. しかしこの実験において用いられた UML 図は 2 種類 (クラス図, コラボレーション図) であった. UML 図には現在 9 種類の図が存在し, それらに対しても両手法の比較評価を行う必要があると指摘されている[7].

そこで本論文では, さらに多くの種類の UML 図を用いた設計仕様書に対し, CBR, PBR の両手法でのレビュー実験を実施し, 得られたデータを分析することで, 両手法の比較評価を行うことを目的とする. 実験では情報科学科の 3 年生 59 人を被験者とし, レビュー対象として 2 つのシステムの UML 図を用いた. 被験者はあらかじめ複数のバグを埋め込んだ UML 図を, 指定されたレビュー手法でチェックし, 検出したバグ, 検出時刻等を報告した. 分析では, 個人データでのバグ検出率の分析と, チームに分けてバグ検出数とレビュー時間の比較を行った.

以降, 2. では CBR, PBR の両手法について説明する. 3. では今回行ったレビュー実験について述べる. 4. ではレビュー実験によって得られたデータに対し分析及び考察を行う. 最後に 5. では, まとめと今後の課題について述べる.

2. CBR と PBR

2.1 Checklist-based Reading

Checklist-based Reading(CBR) は, レビュー対象プロダクトと共に, いくつかのチェック項目が記されたリストを使用し, そのチェック項目に yes/no 形式で答えながらレビューを行う手法である.

CBR のチェックリストは基本的に以下の項目から成る[7].

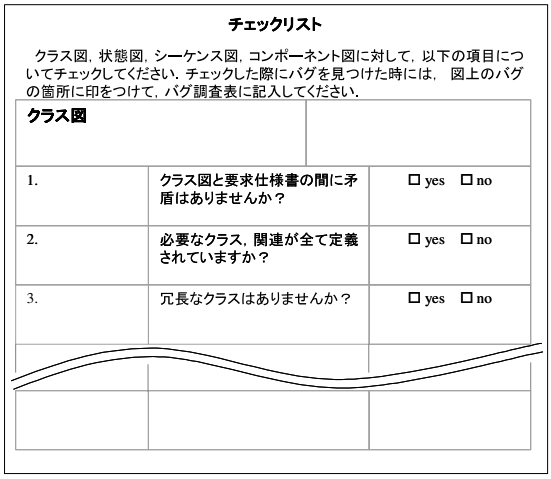
- *"Where to look"* - 可能性のある問題点のリスト
- *"How to look"* - 各問題点についてバグを特定する手がかりとなる

CBR では, チェック項目に yes/no で答えながら進めていくので, レビューは容易に行うことができる反面, プロダクトの全体の情報を網羅する必要があるため, 1 回のレビューに費やす時間や負担は大きくなってしまうという欠点がある[7].

なお, 本論文における実験では, Chernak[5] によって提案された構造に基づき, UML 図のレビュー用に独自のチェックリストを作成した. 実験で使用したチェックリストの一部を図 1 に示す. UML の各図に対して設定された各チェック項目に対し yes/no で答え, その際に発見したバグをバグ調査票に記入するという形式になっている.

2.2 Perspective-based Reading

Perspective-based Reading(PBR) は, レビューの


チェックリスト

クラス図，状態図，シーケンス図，コンポーネント図に対して，以下の項目についてチェックしてください．チェックした際にバグを見つけた時には，図上のバグの箇所に印をつけて，バグ調査表に記入してください．

クラス図		
1.	クラス図と要求仕様書の間に矛盾はありませんか？	□ yes □ no
2.	必要なクラス，関連が全て定義されていますか？	□ yes □ no
3.	冗長なクラスはありませんか？	□ yes □ no

図 1 チェックリスト

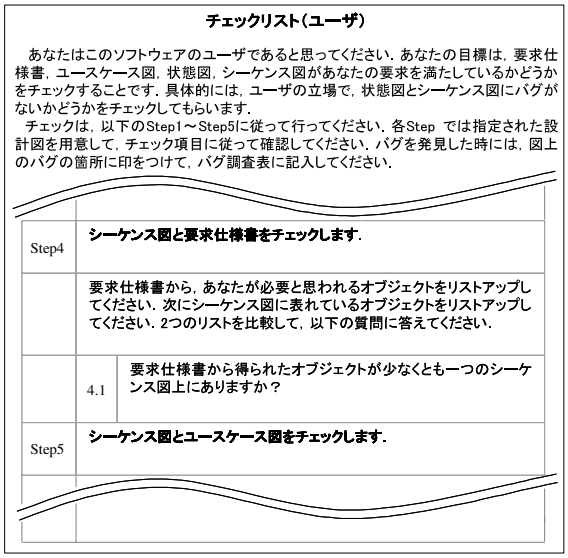
実行方法等を記したシナリオを用いたレビュー手法 Scenario-based Reading(SBR) のさまざまな種類の内のひとつである．対象ソフトウェアに関わるいくつかの異なった視点 (ユーザ, 設計者, テスタ等) から，それぞれのシナリオを用いてプロダクトをレビューし，バグを検出する．

PBR のシナリオは主に 3 つの部分から成る[7]．

- *Introduction* - 対象プロダクトの要求品質を記述した部分
- *Instruction* - どのドキュメントを使用するか等，レビュー全体の指針となる部分
- *Questions* - レビュー者がレビュー中に答える質問の部分

PBR では各視点ごとでレビュー対象となるドキュメント数が異なっているので，レビューにかかる時間や負担を抑えることができる．

本論文における実験では，ユーザ，設計者，プログラマという 3 つの視点を設定し，各役割でシナリオを作成し，レビューを行った．[7),9)] 例として，今回の実験でユーザ視点でレビューを行った者が使用したシナリオの一部を図 2 に示す．各 Step ではオブジェクトのリストアップなどの作業等が記されており，それを行った上で各質問に答えるという形式になっている．


チェックリスト（ユーザ）

あなたはこのソフトウェアのユーザであると思ってください．あなたの目標は，要求仕様書，ユースケース図，状態図，シーケンス図があなたの要求を満たしているかどうかをチェックすることです．具体的には，ユーザの立場で，状態図とシーケンス図にバグがないかどうかをチェックしてもらいます．

チェックは，以下のStep1～Step5に従って行ってください．各Step では指定された設計図を用意して，チェック項目に従って確認してください．バグを発見した時には，図上のバグの箇所に印をつけて，バグ調査表に記入してください．

Step4	**シーケンス図と要求仕様書をチェックします．**	
	要求仕様書から，あなたが必要と思われるオブジェクトをリストアップしてください．次にシーケンス図に表れているオブジェクトをリストアップしてください．2つのリストを比較して，以下の質問に答えてください．	
	4.1	要求仕様書から得られたオブジェクトが少なくとも一つのシーケンス図上にありますか？
Step5	**シーケンス図とユースケース図をチェックします．**	

図 2 シナリオ

3. 評 価 実 験

3.1 概　　要

先に挙げた 2 つの手法によるレビュー実験を，情報科学科 3 年生の学生 59 人に対して行った．この学生達は，UML とレビューについて講義で学んでおり，それぞれに対する予備知識は持っていた．

レビュー実験は，後述する 2 つのシステムをレビュー対象とし，各手法，役割に応じた UML 図，チェックリスト (シナリオ)，バグ報告書を配布し，CBR, PBR の 2 つの方法でレビューを行った．その後，配布した UML 図に検出したバグの該当箇所に印をつけたものと，検出したバグに該当する UML 図名，そのバグを発見したチェックリスト (シナリオ) の項目，そしてそのバグの発見時刻を記録したバグ報告書を回収した．

3.2 実 験 詳 細

3.2.1 レビュー対象システム

レビューの対象としたシステムには，「セミナ情報システム」と「医療情報システム」の 2 つを用いた[6]．

各システムは要求仕様書，5 種類の UML 図 (ユースケース図，クラス図，アクティビティ図，シーケンス図，コンポーネント図) により構成されている．また，CBR 手法を用いた者には全てのドキュメントを配布したのに対し，PBR 手法を用いた者には各役割毎にレビューを行ったドキュメント数が異なっている．各システムについて，それぞれのドキュメント数と役割ごとの割当てを表 1 に示す．表における丸印は，そのドキュメントが対応する役割に割当てられていることを示す．例えば，セミナ情報システムを PBR 手法のユーザ視点でレビューを行った者は，要求仕様書，ユースケース図，アクティビティ図，シーケンス図の 4 種類を用いてレビューを行った．

3.2.2 バ　　グ

2 つのシステムそれぞれには，クラス図に 3 個，アクティビティ図に 4 個，シーケンス図に 5 個，コンポーネント図に 3 個の合計 15 個のバグを混入した．バグ

表 1 各システムのドキュメント数及び割当て

	ドキュメント数		割当			
				PBR		
	セミナ	医療	CBR	ユーザ	設計者	プログラマ
要求仕様書	1	1	○	○	○	○
ユースケース図	1	1	○	○	○	○
クラス図	1	1	○		○	○
アクティビティ図	8	7	○	○		
シーケンス図	12	7	○	○	○	○
コンポーネント図	1	1	○			○

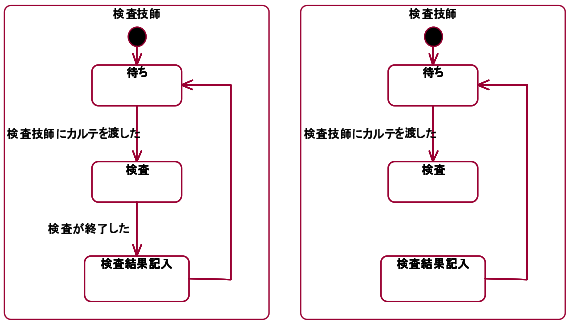


図 3 構文的バグ (状態の要素の欠如)

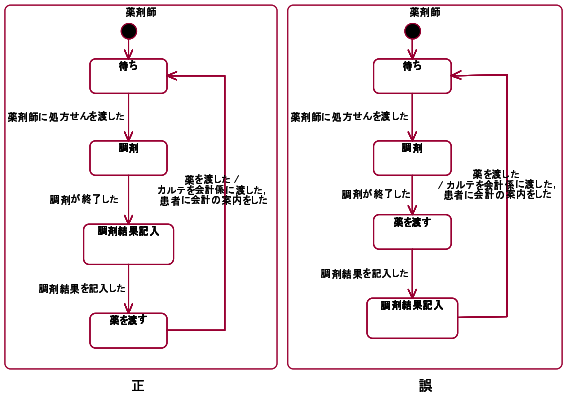


図 4 意味的バグ (状態の順序の違い)

には 3 種類あり, 構文的なもの, 意味的なもの, 他の図との関連によるものがある. 図 3 から図 5 に 3 種類のバグの一例を示す.

また, PBR については各視点で用いるシナリオによって検出できるバグ数が異なり, それについて以下に示す.

- ユーザ視点 - 7 個 (アクティビティ図 4 個, シーケンス図 3 個)

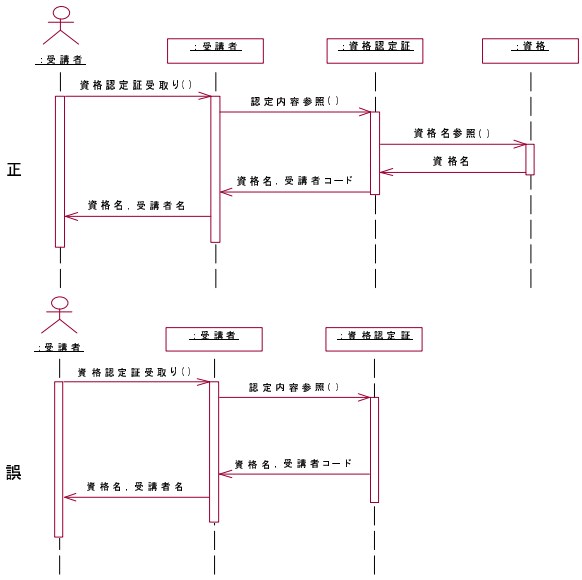


図 5 他の図との関連によるバグ (必要なオブジェクトの欠如)

表 2 人数配分

	PBR			CBR
	ユーザ	設計者	プログラマ	
セミナ情報システム	7	6	6	11
医療情報システム	7	6	6	10

- 設計者視点 - 6 個 (クラス図 3 個, シーケンス図 3 個)
- プログラマ視点 - 9 個 (クラス図 3 個, シーケンス図 3 個, コンポーネント図 3 個)

3.2.3 人数配分

実験に参加した 59 人の配分は表 2 の通りである.

3.2.4 実験データ

この実験により収集したデータをまとめたものを表 3 に示す.

これは, 提出された UML 図, バグ報告書に基づいて各値を算出したものである. 各システム, 各手法毎に, バグについては検出可能数, 実際の検出数の平均, 最大値と最小値を記し, 時間についてはレビュー時間の平均, 最大値と最小値を記した.

4. 分析と考察

実験で得られたデータを個人データ, チームデータの 2 つの方向から分析を行った.

4.1 個人データ分析

個人データについては, バグの検出率 [(検出バグ

表 3 実験データ

		人数	バグ			時間(分)	
			検出可能数	平均発見数	max/min	平均	max/min
セミナ情報システム	ユーザ	7	7	4.43	6/3	60.43	90/46
	設計者	6	6	5.00	6/4	65.50	80/51
	プログラマ	6	9	6.50	9/5	76.67	95/40
	CBR	11	15	10.55	13/8	74.64	90/62
医療情報システム	ユーザ	7	7	4.43	6/3	48.29	70/25
	設計者	6	6	3.83	5/3	59.17	73/30
	プログラマ	6	9	6.33	7/5	63.30	77/44
	CBR	10	15	10.50	12/8	70.10	94/60

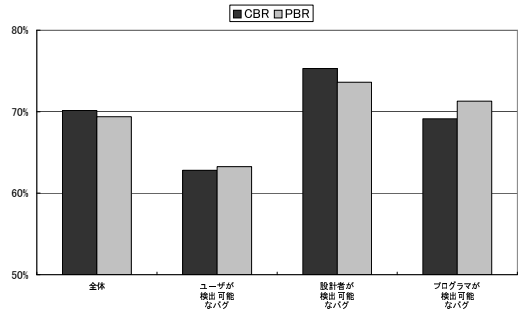


図 6 バグ検出率 (手法)

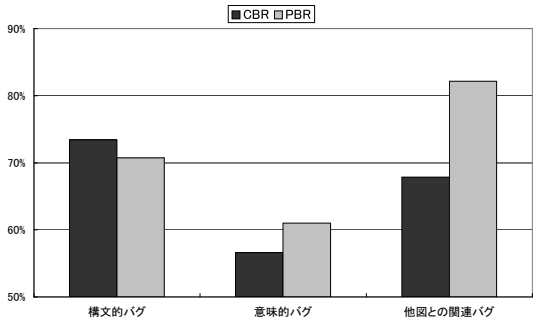


図 7 バグ検出率 (バグの種類)

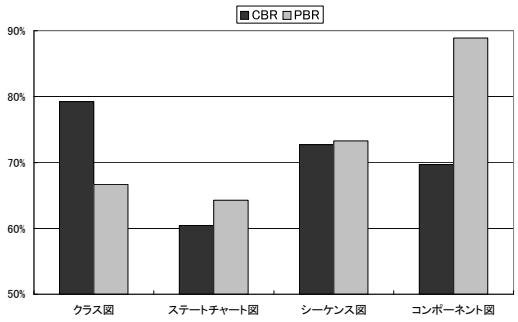


図 8 バグ検出率 (UML 図)

数)/(検出可能バグ数)] を以下の 3 つの観点から計算し, 分析を行った.

4.1.1 手法による違い

これは両手法におけるバグ検出率を算出したものである. その結果を図 6 に示す. グラフについては, 全体の検出率, そして PBR の各視点での検出率を示している. また, CBR についての各視点での検出率は, 各視点で検出可能なバグについて算出したものである. 例えば, 「ユーザが検出可能なバグ」という項目については, ユーザ視点でレビューを行った場合での検出可能なバグについて, PBR 手法と CBR 手法での検出率を示している. CBR 全体での平均バグ検出率が 70.2%, PBR 全体での平均バグ検出率が 69.4%となっており, 手法別に見ると今回の実験では, CBR と PBR に特に明確な差は見られなかった.

4.1.2 バグの種類別

これは混入したバグのその種類 (構文的, 意味的, 他図との関連) 別に検出率を算出したものである. 結果を図 7 に示す. 構文的なバグについては CBR(73.4%) が, 意味的なバグや他の図との関連に関するバグについては PBR(それぞれ 61.0%, 82.1%) がより高い検出率を示した.

この結果から, CBR は各チェック項目に対し単に yes/no で答える形式であることから, プロダクトを表層的にチェックしているため, 構文的なバグは見つけやすいと考えられる. また, PBR はチェックするプロダクトが限られていることと, シナリオの中で他の図との一貫性をチェックする作業が含まれていることから, プロダクトに対しより深くチェックを行うことができ, 意味的なバグ, 他の図との関連によるバグがより多く発見できたと考えられる. 従って, システムの表面的なレビューには CBR が, 内部的な部分のレビューには PBR が有効であると考えられる.

4.1.3 UML 図別

これはバグの混入された UML 図それぞれについて検出率を算出したものである. 結果を図 8 に示す. クラス図においては CBR(79.2%) に, アクティビティ, シーケンス, コンポーネント図では PBR(それぞれ 64.3%, 73.3%, 88.9%) に高い検出率が見られた.

この結果は, クラス図に含まれていたバグに構文的なものが多かったことと, コンポーネント図に含まれていたバグに意味的なバグ, 他の図との関連によるバグが多かったためである. 従って, 前述と同様, システムの表面的なレビューには CBR が, 内部的な部分のレビューには PBR が有効であるといえる.

4.2 チームデータ分析

一般にレビューは, 対象となるプロダクトに対して

複数の人間がチームで行う．そこで, 今回の実験では被験者を組み合わせて仮想的にチームを作り[1),3),13)], その上でバグ検出数, レビュー時間, バグ検出コストについて分析を行った．

4.2.1 チームの組み方

チームを作るにあたっては, 2 つの基準を設け, その基準に従って組み合わせた．

(1) 検出可能バグ数 (基準 A)
今回行った実験では, 検出可能なバグ数がそれぞれ異なっていた．CBR は 1 人あたり設定したバグ 15 個すべてを検出可能であるのに対し, PBR ではユーザ, 設計者, プログラマの 3 人を合わせることで初めて 15 個検出できるように設定していた．このことから, 各システムにおいて以下の方式でチームを組ませた．
- CBR - 1 チーム 1 人
- PBR - 1 チーム 3 人 (ユーザ, 設計者, プログラマから 1 人ずつ無作為に選出)

(2) 人数 (基準 B)
これは単に 1 チームあたりの人数をそろえて組ませたものである．
- CBR - 1 チーム 3 人 (無作為に選出)
- PBR - 1 チーム 3 人 (ユーザ, 設計者, プログラマから 1 人ずつ無作為に選出)

4.2.2 グループ化

前述のようにチームを組ませた後, さらにグループ化を行った．

まず基準 A の組み方では, CBR についてセミナ情報システムでは 11 チームで 1 グループとし, 医療情報システムでは 10 チームで 1 グループとした．PBR については 1 つの組み合わせ方でチームを組ませた場合に 6 チーム出来るので, それを 1 グループとした．全ての組み合わせのグループ化を考えると, CBR と PBR のグループ間の総比較回数は,

$$1 \times \frac{{}_7P_6 \times 6! \times 6!}{6!} = 3,628,800$$

となる．

次に基準 B の組み方では, CBR は 1 つの組み合せ方でチームを組ませた場合に 3 チーム出来るので, それを 1 グループとした．PBR については基準 A の場合と同様である．全ての組み合わせのグループ化を考えると, CBR と PBR のグループ間の総比較回数は, セミナ情報システムでは,

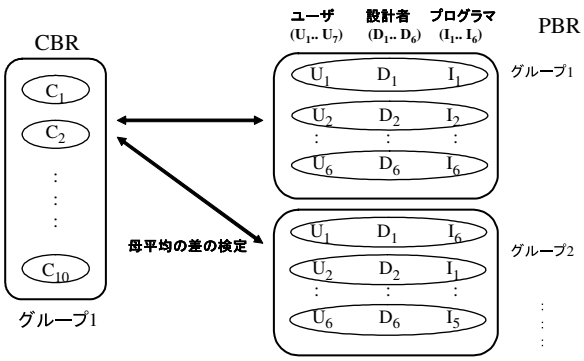


図 9 基準 A でのグループ化 (楕円, 四角で囲まれた部分がそれぞれ 1 チーム, 1 グループを表す)

$$\frac{{}_{11}C_3 \times {}_8C_3 \times {}_5C_3}{3!} \times \frac{{}_7P_6 \times 6! \times 6!}{6!}$$

$$= 55,883,520,000$$

となり, 医療情報システムでは,

$$\frac{{}_{10}C_3 \times {}_7C_3 \times {}_4C_3}{3!} \times \frac{{}_7P_6 \times 6! \times 6!}{6!}$$

$$= 10,160,640,000$$

となる．

例としてセミナ情報システムにおける基準 A でのグループ化を図式化したものを図 9 に示す．CBR 手法を用いた者 ($C_1..C_{10}$) 合計 10 人 (10 チーム) を 1 グループとし, PBR 手法のユーザ視点 ($U_1..U_7$), 設計者視点 ($D_1..D_6$), プログラマ視点 ($I_1..I_6$) から 1 人ずつ選んだ 3 人を 1 チームとし, 合計 6 チームを 1 グループとしている．

4.2.3 データ算出

前述のようにグループ化を行った後, バグ検出数, レビュー時間, バグ検出コストについてデータ算出を行った．詳しい算出方法を基準別で以下に示す．

(1) 基準 A
CBR では, 1 チーム 1 人なので, メンバーのそれぞれの検出数の平均値をグループのバグ検出数, レビュー時間, そしてレビュー時間をバグ検出数で割った値をグループのバグ検出コストとした．
PBR では, 15 個のバグについて重複する部分を 1 個と数えて 3 人の検出数を算出し, チームのバグ検出数とした．レビュー時間については 3 人のレビュー時間のうちの最長時間をチームの時間とした．バグ検出コストについては, チームのレビュー時間をチームのバグ検出数で割った

値をチームのコストとした. その後 6 チーム分の平均値をグループでのバグ検出数, レビュー時間, バグ検出コストとした.

(2) 基準 B

CBR では, 15 個のバグについて重複する部分を 1 個と数えて 3 人の検出数を算出し, チームのバグ検出数とした. レビュー時間については 3 人のレビュー時間のうちの最長時間をチームの時間とした. バグ検出コストについては, チームのレビュー時間をチームのバグ検出数で割った値をチームのコストとした. その後 3 チーム分の平均値をグループでのバグ検出数, レビュー時間, バグ検出コストとした.

PBR については基準 A と同様である.

4.2.4 データ比較

このようにして各グループでのデータを算出し, グループ間の比較を行った. 比較は単にグループでの検出数とレビュー時間, バグ検出コストを比べ, 検出数ではその数が多い方を, レビュー時間は短かった方を, バグ検出コストは値の小さい方を有効とし, その数をカウントした方法と, バグ検出数について有意水準 2.5%で母平均の検定を行った. 前者の比較方法の結果を基準 A は表 4, 表 5, 表 6 に, 基準 B は表 7, 表 8, 表 9 に示す. それぞれの表では, 比較の結果, その手法が有効であった比較の数と, 値が同じであった比較の数を示している.

表 4 比較結果 (基準 A その 1)

	バグ検出数		
	CBR	PBR	等しい
セミナ情報システム	893	3,615,154	12,753
医療情報システム	352	3,611,280	17,168

表 5 比較結果 (基準 A その 2)

	レビュー時間		
	CBR	PBR	等しい
セミナ情報システム	1,524,642	2,050,817	53,341
医療情報システム	1,209,600	2,384,640	34,560

また検定を行った結果, 基準 A では PBR 手法, 基準 B では CBR 手法を用いた方が, 多くのバグを検出していたことが確認された.

チームでの比較においては, Laitenberger らの実験のように, どちらの手法が有効であるかを結論付けることは出来なかった. これは実験環境の違いが原因のひとつではないかと考えられる. Laitenberger らが行った実験[7] では, 参加者に対して無作為に 3 人を選んでチームを作り, そのチームでミーティングを行った上でバグ検出を行っていたが, 本論文における実験では個々のデータを仮想的に組み合わせてチームのデータとしていた. そのため, Laitenberger らの実験では, ミーティングを行うことで新たにバグが発見された場合, それがバグ検出数に反映されていたということが考えられる.

表 6 比較結果 (基準 A その 3)

	バグ検出コスト		
	CBR	PBR	等しい
セミナ情報システム	0	3,628,800	0
医療情報システム	27,792	3,564,864	36,144

表 7 比較結果 (基準 B その 1)

	バグ検出数		
	CBR	PBR	等しい
セミナ情報システム	54,753,326,688	585,743,712	544,449,600
医療情報システム	10,160,482,176	8,064	149,760

表 8 比較結果 (基準 B その 2)

	レビュー時間		
	CBR	PBR	等しい
セミナ情報システム	27,809,497,152	27,965,566,656	108,456,192
医療情報システム	34,972	10,154,262,899	6,342,129

表 9 比較結果 (基準 B その 3)

	バグ検出コスト		
	CBR	PBR	等しい
セミナ情報システム	52,492,081,048	3,167,916,016	223,522,936
医療情報システム	5,393,963,535	4,644,066,395	122,610,070

5. まとめと今後の課題

本研究では, UML で記述された設計仕様書を対象とし, 2 つのレビュー手法 CBR と PBR を用いてレビュー実験を実施し, 得られた実験データの分析を行い, 両手法の比較評価を行った.

その結果, 構文的なバグ検出には CBR 手法, 意味

的なバグ検出や UML の図間の関連や一貫性をチェックするという点においては PBR 手法に, それぞれ有効性が確認された.

今後の課題としては,

- チーム間の比較におけるより詳細な分析
- 仮想的グルーピングではなく, 実際のチームレビューをした上での評価
- チェックリスト, シナリオ作成等の準備作業におけるコスト及びコストパフォーマンスについての評価
- 更なる追証実験によるデータ収集

などが挙げられる. チーム間のより詳細な分析については, 被験者にアンケートをとりその結果を分析に反映させることや, チームの組み合わせに新たな基準を設ける等, 現在も思案, 分析中である. また, 今後も学生を対象とし, 考察を踏まえた環境を設定した新たなレビュー実験等を予定している.

参 考 文 献

1) V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard, M. V. Zelkowitz, “The Empirical Investigation of Perspective-Based Reading”, Empirical Software Engineering, I: 133-164, 1996.
2) V. R. Basili, “Evolving And Packaging Reading Technologies”, Journal of Systems and Software 38(1), 3-12, 1997.
3) S. Biffl, M. Halling, “Investigating the Influence of Inspector Capability Factors with Four Inspectinon Techniques on Inspection Performance”, Proceedings of the Eighth IEEE Symposium on Software Metrics, 107-117, 2002.
4) G. Booch, J. Rumbaugh, I. Jacobson, “The Unified Modeling Language User Guide”, Addison Wesley Longman, Inc., 1999.
5) Y. Chernak, “A Statistical Approach To The Inspection Checklist Formal Synthesis And Improvement”, IEEE Transactions on Software Engineering, 22(12): 866-874, 1996.
6) 伊藤 潔, 廣田 豊彦, 冨士 隆, 熊谷 敏, 川端 亮, “ソフトウェア工学演習”, オーム社, 2001.
7) O. Laitenberger, C. Atkinson, M. Schlich, K. El Emam. “An Experimental Comparison Of Reading Techniques For Defect Detection In UML Design Documents”, The Journal of Systems and Software, 53: 183-204, 2000.
8) O. Laitenberger, J. M. DeBaud, “An Encompassing Life Cycle Centric Survey Of Software Inspection”, Journal of Systems and Software, 50: 5-31, 2000.
9) O. Laitenberger, C. Atkinson, “Generalizing Perspective-Based Inspection To Handle Object-Oriented Development Artifacts”, Proceedings of the 21st International Conference on Software Engineering, 2000.
10) J. Schmuller, 多摩ソフトウェア [訳], 長瀬 嘉秀 [監訳], “独習 UML”, 翔泳社, 2000.
11) S. Kusumoto, A. Chimura, K. Matsumoto, T. Kikuno, Y. Mohri, “A Promising Approach to Two-person Software Review in Educational Environment”, Journal of Systems & Software, Vol. 40, No. 2: 115-123, Feb., 1998.
12) 土田 昭司, “社会調査のためのデータ分析入門”, 有斐閣, 1994.
13) C. Wohlin, A. Aurum, H. Petersson, F. Shull, M. Ciolkowski, “Software Inspection Benchmarking - A Qualitative and Quantitative Comparative Opportunity”, Proceedings of the Eighth IEEE Symposium on Software Metrics, 118-127, 2002.