Vol.2012-SE-177 No.9
2012/7/20

過去の欠陥データの深刻度と修正工数に着目した重点検出欠陥の選択手法

小川　睦子[1]　森崎　修司[1]

概要：修正コスト低減の効果の大きな欠陥やリスク低減が大きな欠陥の検出を目的とし，ソフトウェアレビューにおいて検出すべき欠陥種別を選定する手法を提案する．提案手法では，過去のレビュー指摘記録や不具合情報から，修正コストやリスクの大きかった欠陥を要約し抽象化し，レビューで検出すべき欠陥種別を得る．欠陥種別の間で優先順位をつけ，優先順位の高い欠陥種別をレビューアに割当てる．また，提案手法を具体的なシステムに適用した場合の利用シナリオを示す．

キーワード：　レビュー，観点，修正工数，影響度

1. はじめに

ソフトウェアレビューは，プログラムが完成する前に実施できるソフトウェアの品質改善のための技法である．これまでリーディングテクニックと呼ぶ欠陥検出の具体的手順やガイドラインが数多く提案されている [1][2][3][4]．ソフトウェアレビューのメリットの１つは，早期に欠陥修正をすることで，テストで発見，修正されたときよりも修正工数を低減できることにある．しかしながら，レビューによって得られる修正工数低減の効果は欠陥によって様々である．早期発見により，大きな修正工数低減効果がある欠陥が存在する一方で，それほど修正工数低減効果のない欠陥もある．例えば，誤字脱字は修正工数低減効果の低い欠陥の１つといえる．また，既存部分との整合性が十分にとれていない機能追加はレビューでの検出により，大きな修正工数低減効果をもたらす．誤字脱字よりも修正工数が大きい欠陥を優先的に検出することにより，より大きな修正工数低減効果を得ることができる．

欠陥がシステムに与える影響も様々である．早期に検出して修正をしなければ大きな影響を与えてしまう欠陥もあれば，修正を遅らせてもそれほど影響はでない欠陥もある．修正工数と同様に，大きな影響を与える欠陥を早期に見つけ修正するほうが早期発見，修正によるリスク低減の効果が大きい．

レビューの効果を大きくするためには，修正工数が大きい欠陥やシステムに大きな影響を与える欠陥を多く検出する必要があり，そのような欠陥種別を特定し，欠陥を検出できるような着眼点からレビューを行うことが必要である．本稿では，そのような欠陥種別の特定を目的として，過去のレビュー記録と不具合管理データを用いてレビューで特に検出することにより，より大きな効果をもたらす重点検出欠陥を選択する手法を提案する．

[1] 静岡大学　情報学部
Faculty of Informatics, Shizuoka university

2. 提案手法

2.1 前提

過去データから導き出した観点に沿ってレビューを行う方法を提案する．そのため，過去に類似のソフトウェア開発プロジェクトが行われていることを前提とする．過去のプロジェクトで蓄積された過去データとして，レビューで指摘されたレビュー指摘記録とテストで検出された不具合記録を用いる．

本稿では，レビューで検出されたバグは「レビュー指摘」・テストで検出されたバグは「不具合」として区別している．また、レビュー指摘と不具合を抽象化し要約したものを「欠陥」と呼ぶ．

2.1.1 レビュー指摘記録

レビューを行ったときに検出された欠陥の情報は，レビュー指摘記録表などに記録される．表１は、レビュー指摘記録表の例を示したものである．本手法を利用するにあたり，レビュー指摘記録表のなかで最低限必要となる情報は “指摘種別”，“指摘内容”，“修正工数” の 3 つである．

指摘種別とは，指摘された内容の大まかな分類を示すものである．たとえば，例外処理の漏れやデータベース管理の誤り，といった大まかな分類とする．指摘内容では，指

摘種別よりも詳しい説明を記述する．例えば，「入力文字は英数字のみと制限しているが，日本語を入力できる欄がある」などである．修正工数は，指摘されたことを修正するのにどの程度の工数を必要としたかを表す．

2.1.2 不具合記録

テストで検出された不具合を記録したものを不具合管理表と呼ぶ．本研究で前提としている不具合管理表では，不具合と将来の機能要望，不具合かどうかが明確でないものは区別できることを前提とし，不具合のみを対象とする．表 2 は，不具合管理表の例を示したものである．

提案手法を利用するにあたり，不具合管理表のなかで最低限必要となる情報は “不具合種別”，“不具合の概要”，“混入工程”，“修正工数” の 4 つである．

不具合種別は，検出された不具合の分類を示すものである．セキュリティ上の問題や計算処理の誤りなど，大まかな分類とする．不具合の概要では，不具合種別よりも詳しく不具合の説明を記述する．例えば，「登録されていないユーザ ID とパスワードでもログインが出来た」などである．修正工数は，検出された不具合を修正するためにどの程度の工数を要したかを記録したものである．

2.1.3 役割

提案手法では，アナリスト，モデレータ，レビューアの 3 つの役割を想定している．アナリストとモデレータは通常 1 名ずつで，同一人物が担当してもよい．レビューアは 2 名以上である．

- アナリスト
  レビュー指摘記録表と不具合管理表のデータの要約を行い，要約したデータに対して修正工数の見積りと影響度と優先度の評価をする．そして，評価にもとづいて観点を選択する．
- モデレータ
  アナリストが選択した観点に対して，観点に該当する欠陥を検出するためのシナリオを作成する．そして，レビューアのスキルと経験を考慮して観点の割り振りを行う．
- レビューア
  モデレータによって割り当てられた観点に沿ってレビューを行う．

2.1.4 欠陥種別

欠陥種別とは，レビュー指摘記録表と不具合管理表を要約して得られた欠陥の種類を表すものである．欠陥種別は，レビューを実施するときに「どのような欠陥を検出するべきか，どこに着目してレビューを行えばいいか」ということをレビューアに示す．レビューの際に，優先的に見るべき欠陥種別として選択されたものを観点と呼ぶ．

2.2 手法の概要

図 1 は提案手法の概要を表したものである．まず，アナリストがレビュー指摘記録表と不具合管理表のデータを要約した欠陥データ要約表を作成し，欠陥データ要約表から観点となる欠陥種別を基準に従って選択する．次に，モデレータは，選択された各観点に対して欠陥を検出するための指針となるようなシナリオを作成し，レビューアの経験や能力に合わせて観点を割り当てる．そして，レビューアはモデレータから割り当てられた観点に沿ってレビューを行う．

過去データから観点を選択しレビューを実施するまでの流れは以下の通りである．

(1) レビューで指摘された指摘記録データの要約
  アナリストは，レビュー指摘記録表のデータを要約する．
(2) テストで検出された不具合データの要約
  手順 (1) と同様に，不具合管理表のデータを要約する．
(3) レビュー指摘要約表と不具合データ要約表の統合
  手順 (1),(2) で要約したデータを，さらに 1 つの表にまとめる．
(4) 影響度の評価
  手順 (3) で要約した各欠陥に対して、影響度を評価する．
(5) 優先度の評価
  手順 (4) で評価した影響度と修正工数を参考に，各欠陥の優先度を評価する．
(6) 観点の選択
  優先度の評価を基準に，レビューで優先的に見るべき観点として適当と考えられる欠陥種別を選択し，観点にする．
(7) 欠陥検出のためのシナリオ作成
  選択された観点に対して，どうすれば観点に該当する欠陥を検出できるかという読み方を定義したシナリオを作成する．
(8) レビューアへの観点の割り当て
  各観点をレビューアに割り当てる．個数や種類に制限はなく，レビューアに合わせて割り当てる観点を選ぶ．
(9) 観点に沿ったレビューの実施
  割り当てられた観点に沿ってレビューを実施する．シナリオに従い，観点に該当する欠陥を検出する．

提案手法の概要図で示したように，手順 (1)〜(6) がアナリスト担当，手順 (7),(8) がモデレータ担当，手順 (9) がレビューア担当の作業である．

2.3 手順

2.3.1 (1) レビューで指摘された指摘記録データの要約

アナリストは，レビュー指摘記録表に記録されている内容について，類似のもの同士を抽象化し，要約する．

要約したデータはレビュー指摘要約表にまとめる．レビュー指摘要約表は “指摘種別”，“指摘内容”，“修正工数”，

表 1 レビュー指摘記録表の例
Table 1 An example of record table which was pointed out in inspection

検出場所	指摘 ID	指摘種別	指摘内容	原因	修正方法	混入工程	修正工数	···
p.12-15	101	インタフェース	各画面に設置した「戻る」ボタンの位置が統一されていない			基本設計	2	
p.20	102	DB 設計	正規化が出来ていないテーブルがある			詳細設計	5	
p.22	103	例外処理	入力可能文字は英数字のみと制限があるのに，日本語を入力してもらう欄がある．			基本設計	1	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 2 不具合管理表の例
Table 2 An example of bug list detected in testing phase

不具合 ID	不具合種別	不具合の概要	再現方法	結果	修正方法	混入工程	修正工数	···
201	セキュリティ	登録されていないユーザ ID とパスワードでもログインが出来た．	ログイン手続きを行う			コーディング	2	
202	計算処理	計算結果が制限桁数を超えたためエラーが起きた	給与計算機能を使う			外部設計	4	
203	セッション	ログイン後，数分でセッションが切れログアウト状態になる	ログイン後，作業を行う			内部設計	5	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

“見積修正工数”，“指摘内容の合計” の 5 項目から成る．

本研究での要約とは $d_1, d_2, \cdots, d_m$ のレビュー指摘があるとき、類似の欠陥であると判断できるものを抽象化し、1 つのレビュー指摘 $D_1 = \{d_1, \cdots, d_m\}$ として定義し直すことを指す．抽象化の基準や要約の方法は，今後の研究で定義していく予定であり，現時点ではアナリストの経験や判断に任せる．

図 2 は，レビュー指摘記録表のデータをレビュー指摘要約表に要約する様子を表したものである．レビュー指摘記録表には，指摘が m 個，指摘種別が k 種類ある．複数の指摘を 1 つの指摘種別に要約するので $k \leq m$ となる．要約の際には，レビューでの欠陥検出の際に必要な情報を残しつつ，なるべく k を小さくする．

図 2 において，レビュー指摘記録表の指摘：$d101, d103$ が類似の内容だった場合，2 つを要約して新たな指摘 D_1 とする．指摘の概要は，2 つの指摘の概要を要約した説明とする．修正工数は，まとめた指摘の修正工数の平均値とする．つまり，d_1, d_3 が D_1 として要約できる場合，D_1 の修正工数は (d_1 の修正工数+d_3 の修正工数)/2 となる．

指摘の要約が終わったあと，アナリストは見積修正工数の計算を行う．レビューで見逃し，テストで指摘種別 D_1 が見つかっていた場合，修正工数はいくら必要になっていたかをアナリストの過去の経験と知識から見積もる．

2.3.2 (2) テストで検出された不具合データの要約

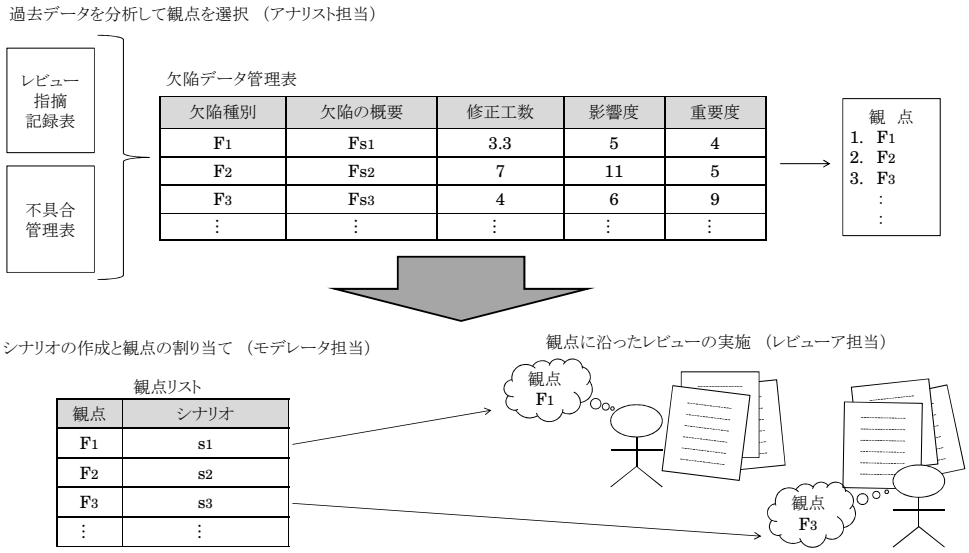
アナリストは，不具合管理表に記録されている各不具合について，種類が似ているもの同士を手順 (1) と同様に要約していく．ここで対象とするデータは，不具合の混入工程がコーディング以前であるもの，つまり，レビューで見逃していてテストで発見された不具合に限定する．要約したデータは不具合データ要約表にまとめていく．不具合データ要約表は “不具合種別”，“不具合の概要”，“修正工数”，“見積修正工数”，“不具合の合計” の 5 項目から成る．

ここでの要約は手順 (1) と同様に $b_1, b_2, \cdots, b_n$ の不具合があるとき、類似の不具合であると判断できるものを抽象化し、1 つの不具合 $B_1 = \{b_1, \cdots, b_n\}$ として定義し直すことである．要約はアナリストの経験にもとづいて実施する．

図 3 は，不具合管理表のデータを不具合データ要約表にまとめるまでの流れを表したものである．この不具合管理表には，不具合が n 個，欠陥種別が l 種類ある．複数の不具合を 1 つの不具合種別に要約するので，$l \leq n$ となる．

図 3 において，不具合管理表の不具合：$b101, b103$ が類似の不具合だった場合，2 つを要約して新たな不具合 B_1 と定義する．不具合の概要は，2 つの不具合の概要を要約した説明を記述する．修正工数は，まとめた不具合の修正工数の平均値とする．つまり，b_1, b_3 が B_1 として要約できる場合，B_1 の修正工数は (b_1 の修正工数+b_3 の修正工数)/2 となる．

不具合データの要約が終わったあと，アナリストは見積修正工数の計算を行う．レビューの段階で不具合種別 B_1 を検出できていた場合，そのときの修正工数はいくら必要



欠陥種別	欠陥の概要	修正工数	影響度	重要度
F_1	Fs_1	3.3	5	4
F_2	Fs_2	7	11	5
F_3	Fs_3	4	6	9
⋮	⋮	⋮	⋮	⋮

図 1 提案手法の概要
Fig. 1 Overview of the proposed approach

レビュー指摘記録表

検出場所	指摘ID	指摘種別	指摘内容	原因	修正方法	混入工程	修正工数
p.7, L11-12	d101	d_1	ds_1			基本設計	4
p.11-12	d102	d_2	ds_2			詳細設計	1
p.19, p.21	d103	d_1	ds_1			基本設計	3
p.24	d104	d_3	ds_3			要求定義	3
p.30, L10-12	⋮	⋮	⋮	⋮	⋮	⋮	⋮
p.15, L4-5	dm	d_k	ds_m			詳細設計	2

レビュー指摘要約表

指摘種別	指摘内容	修正工数	見積修正工数	欠陥の合計
D_1	Ds_1	3.5	7	2
D_2	Ds2	5	4	4
⋮	⋮	⋮	⋮	⋮
D_k	Ds_k	2	2	3

図 2 レビュー指摘記録表の要約
Fig. 2 Summary of record table which was pointed out in inspection

になっていたかをアナリストの経験と知識から見積もる．

2.3.3 (3) レビュー指摘要約表と不具合データ要約表の統合

アナリストは，レビュー指摘要約表と不具合データ要約表をさらに 1 つにまとめる．例えば，レビュー指摘要約表と不具合データ要約表で種別が同じものがあった場合，前の手順と同様に，類似の欠陥データを要約し新たな欠陥 F_i として表に追記する．また，種別が異なっていても，類似の欠陥であるとみなすことができる場合は一つにまとめる．例えば，D_1, B_1 は種別は異なるが似ていると判断できる欠陥である場合は，D_1, B_1 を要約して欠陥 F_1 と定義する．

複数のデータを要約した場合の修正工数・見積修正工数は，ともに平均値とする．

2.3.4 (4) 影響度の評価

欠陥データ要約表の各欠陥に対して，アナリストは影響度を評価する．影響度とは，もし欠陥が検出されなかったら，または欠陥が修正できていないままだったらシステムにどれほどの影響が及ぶかを示すものである．評価は 1〜5 の 5 段階で行う．1 が最も影響度が少なく，5 が最も影響度が高いことを示す．

表 3 は、欠陥データ要約表の各欠陥の影響度を評価した表である．欠陥 F_2 は検出できなければシステムに大きな影響を与えてしまうので影響度は 5 に，欠陥 F_i はもし完全に修正出来ていなくてもそれほど大きな影響は与えないので影響度は 3 に，という具合に評価をしている様子を表している．

2.3.5 (5) 優先度の評価

アナリストは欠陥の優先度を評価する．優先度とは，早めに検出しておくべきである，見逃してはいけないなど，その欠陥を検出することの優先順位を示す度合である．評価は，影響度と同様に 1〜5 の 5 段階で行う．

表 4 は、欠陥データ要約表の各欠陥の優先度を評価した表である．欠陥 F_2 は修正工数が大きく影響度も高い．そ

不具合管理表

不具合ID	不具合種別	不具合の要約	再現方法	結果	修正方法	混入工程	修正工数
b101	b_1	bs_1	TC1			内部設計	5
b102	b_2	bs_2	TC2			内部設計	3
b103	b_1	bs_3	TC3			外部設計	1
b104	b_3	bs_4	TC4			内部設計	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
bn	b_l	bs_n	TCn			外部設計	2

不具合データ要約表

不具合種別	不具合の概要	修正工数	見積修正工数	不具合の合計
B_1	Bs_1	3	3	2
B_2	Bs_2	4	2	7
⋮	⋮	⋮	⋮	⋮
B_l	Bs_l	9	5	5

図 3 不具合管理表の要約
Fig. 3 Summary of bug list detected in testing phase

レビュー指摘要約表

指摘種別	指摘内容	修正工数	見積修正工数	欠陥の合計
D_1	Ds_1	3.5	7	2
D_2	Ds_2	5	4	4
⋮	⋮	⋮	⋮	⋮
D_k	Ds_k	2	2	3

不具合データ要約表

不具合種別	不具合の概要	修正工数	見積修正工数	不具合の合計
B_1	Bs_1	3	3	2
B_2	Bs_2	4	2	7
⋮	⋮	⋮	⋮	⋮
B_l	Bs_l	9	5	5

欠陥データ要約表

欠陥種別	欠陥の概要	修正工数	見積修正工数	欠陥の合計
F_1	Fs_1	3.3	5	4
F_2	Fs_2	7	11	5
F_3	Fs_3	4	6	9
⋮	⋮	⋮	⋮	⋮
F_i	Fs_i	5	3.5	6

図 4 指摘記録データと不具合データの要約
Fig. 4 Summary of the past data

表 3 欠陥がシステムに及ぼす影響度
Table 3 Impact which defect exerts on the system

欠陥種別	欠陥の概要	修正工数	見積修正工数	欠陥の合計	影響度
F_1	Fs_1	3.3	5	4	4
F_2	Fs_2	7	11	3	5
F_3	Fs_3	4	6	9	5
⋮	⋮	⋮	⋮	⋮	
F_i	Fs_i	5	3.5	6	3

れだけでなく，レビューで発見した方が修正工数は小さくなると予想されているため，優先度は 5 と評価される．欠陥 F_i は影響度は 3 で修正工数もそれほど大きくないため，優先度は 4 と評価されている．

優先度は，修正工数・見積修正工数・影響度を見て，アナリストの知識と経験から評価してもらう．

表 4 欠陥の優先度
Table 4 Priority of defects

欠陥種別	欠陥の概要	修正工数	見積修正工数	欠陥の合計	影響度	優先度
F_1	Fs_1	3.3	5	4	4	3
F_2	Fs_2	7	11	3	5	5
F_3	Fs_3	4	6	9	5	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮
F_i	Fs_i	5	3.5	6	3	4

2.3.6 (6) 観点の選択

欠陥データ要約表から，優先度を基準に観点として適当と考えられる欠陥種別を選択する．図 5 は，観点を選択している様子を表したものである．図では優先度が 4 以上であることを基準にして欠陥種別を選択している．その結果，欠陥種別 $F_2, F_3, \cdots, F_i$ が選ばれる．選んだ欠陥種別が，レビューで優先的に検出する欠陥種別であり，観点となる．

2.3.7 (7) 欠陥検出のためのシナリオ作成

観点を選択し終えたら，モデレータは各観点に対してシナリオを作成する．シナリオとは，読み進め方のガイドとなるもので，観点に当てはまる欠陥を検出するためにどういう順序や着眼点でレビュー対象を読んでいけばよいかを示したものである．

2.3.8 (8) レビューアへの観点の割り当て

モデレータは，観点リストの各観点をレビューアに割り当てる．観点の割り当て方に制約はなく，レビューアの経験や得意な分野によって割り当てる観点の種類と個数をモデレータが決定する．例えば，経験の多いレビューアには観点を複数個割り当てたり，観点リストのなかでも特に重要であると考えられる観点は複数のレビューアに割り当てたりする．

2.3.9 (9) 観点に沿ったレビューの実施

レビューアは，割り当てられた観点に沿ってレビューを

欠陥種別	欠陥の概要	修正工数	見積修正工数	欠陥の合計	影響度	重要度
F1	Fs1	3.3	5	4	4	3
F2	Fs2	7	11	5	5	5
F3	Fs3	4	6	9	5	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Fi	Fsi	5	3.5	6	3	4

観点
1. F2
2. F3
3. :
:
N. Fi

図 5 観点とする欠陥種別の選択
Fig. 5 Selecting the defect type

表 5 仮想プロジェクトの概要
Table 5 Overview of the virtual project

プロジェクト内容	レンタルショップの貸出管理システムの開発（バージョン n）
チーム編成	アナリスト, モデレータ, レビューア r_1, r_2, r_3
利用するデータ	
レビュー記録表	バージョン n-1 のレビュー記録表を利用．最低限必要である “欠陥種別”, “欠陥の概要”, “修正工数” のデータは揃っている．
不具合管理表	バージョン n-1 のレビュー記録表を利用．最低限必要である “不具合種別”, “不具合の概要”, “混入工程”, “修正工数” のデータは揃っている．

行う．シナリオを参考に，観点に該当する欠陥を検出する．

3. 利用シナリオ

3.1 概要

提案手法を利用して観点を設定するまでの流れを具体例を使って紹介する．レンタルショップの貸出管理システムの開発を行うという想定で，利用シナリオを紹介する．表 5 は，プロジェクトの概要を示したものである．レビューを行うメンバーは，アナリスト，モデレータ，レビューア r_1, r_2, r_3 の計 5 人である．

3.2 利用手順

3.2.1 レビューで指摘された指摘記録データの要約

アナリストは，レビュー指摘記録表に記録されている内容について，類似のもの同士を抽象化し，要約する．図 6 は，レビュー指摘記録表のデータを要約する流れを示したものである．レビュー指摘記録表を見ると，指摘 ID : $d101$ と $d104$ の指摘はともにセキュリティに関する指摘であることがわかる．アナリストは，これらを類似の指摘であると判断し，要約し，レビュー指摘要約表に記録する．他のレビュー指摘についても同様に，類似のレビュー指摘同士を要約し，レビュー指摘要約表を作成する．

3.2.2 テストで検出された不具合データの要約

アナリストは，不具合管理表に記録されている不具合のうち，混入工程がコーディングよりも前である不具合を対象に不具合の要約を行う．図 7 は，不具合管理表のデータを要約する流れを示したものである．不具合管理表を見ると，不具合 ID : $b211$ と $b215$ の不具合はともにデータベース設計に関する不具合であることがわかる．アナリストは，これらを類似の不具合であると判断し，不具合データ要約表に記録する．他の不具合についても同様に，類似の不具合同士を要約し，不具合データ要約表を作成する．

3.2.3 レビュー指摘要約表と不具合データ要約表の統合

作成したレビュー指摘要約表と不具合データ要約表をもとに欠陥データ要約表を作成する．アナリストは，前の手順と同じように，レビュー指摘要約表と不具合データ要約表で種別が同じで要約できると判断したものは要約し 1 つの欠陥とする．要約できないものはそれぞれ，欠陥データ要約表に記録する．

3.2.4 影響度の評価

欠陥データ要約表の各欠陥に対して，アナリストは影響度を評価する．データベース設計の欠陥は商品の管理に関わるもので業務に大きく支障を与えるので影響度を 4 に，商品だけでなく顧客の情報も管理しているシステムにとってセキュリティの欠陥は問題であるため影響度は 5 に，という具合でアナリストは影響度を決定する．

3.2.5 優先度の評価

修正工数・見積修正工数・影響度を判断材料にアナリストは各欠陥の優先度を評価する．データベース設計に関する欠陥は修正工数が大きく影響度も高いので優先度は 5 に，計算処理の欠陥は影響度は高いが修正工数は小さいので優先度は 3 に，という具合でアナリストは優先度を評価していく．

表 6 は，各欠陥の影響度と優先度を評価し終えた欠陥データ要約表である．

3.2.6 観点の選択

評価をし終えたアナリストは，欠陥データ要約表から優先度を基準に観点として適切と考えられる欠陥種別を選択する．今回は，アナリストは「優先度が 5 であること」という基準に該当する欠陥種別を選んだので，データベース

レビュー指摘記録表

検出場所	指摘ID	指摘種別	指摘内容	原因	修正方法	混入工程	修正工数
p.15, l.4-5	d101	セキュリティ	編集機能のアクセス権限が設定されていない	顧客の要求確認不足	アクセス権限の再定義	要求定義	2
p.21-22	d102	インタフェース	商品の「区分情報」の入力項目が抜けている	設計ミス	画面設計のやり直し	外部設計	1
p.29, p.31	d103	DB設計	テーブル名は違うが，属性がほぼ同じテーブルが存在している	ER図の設計ミス	各テーブルの属性確認	内部設計	3
p.33	d105	例外処理	エラーと警告のメッセージ区分がされていない	設計者の定義し忘れ	コード設計のやり直し	内部設計	2
p.36 l.10-12	d104	セキュリティ	パスワードの変更手続は管理者以外できない仕様になっている	アクセス権限の定義ミス	アクセス権限の再定義	内部設計	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

レビュー指摘要約表

指摘種別	指摘内容	修正工数	見積修正工数	欠陥の合計
セキュリティ	各機能に対してアクセス権限が定義されていない	2	3	5
DB設計	必要なテーブルと属性の定義に漏れがある	5	9	4
例外処理	表示メッセージの区分分けがされていない	3	3	3
⋮	⋮	⋮	⋮	⋮

図 6 レビュー指摘記録表の要約

Fig. 6 Summary of record table which was pointed out in inspection

不具合管理表

不具合ID	不具合種別	不具合の要約	再現方法	結果	修正方法	混入工程	修正工数
b211	DB設計	返却手続きをしても，商品DBの「状態」項目が変化しない	返却する商品のバーコードを読み取り，返却手続きを行う			コーディング	5
b212	計算処理	返却日が翌月になる場合の貸出日数が正しく計算されない	商品のバーコードを読み取り，貸出手続きを行う			内部設計	4
b213	セッション	「戻る」ボタンを押すと，接続が切れてログイン画面に戻る	xx画面で，「戻る」ボタンを押す			外部設計	2
b214	例外処理	エラーメッセージが表示されず，そのまま次の処理に進む	入力必須項目を未入力のまま，登録ボタンを押す			外部設計	3
b215	DB設計	該当するデータは存在するのに，検索結果が返ってこない	商品IDを入力して検索ボタンを押す			内部設計	8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

不具合データ要約表

不具合種別	不具合の概要	修正工数	見積修正工数	不具合の合計
DB設計	SQL文の命令の記述に誤りがある	6.5	5	2
計算処理	貸出期間・料金の計算結果が予想と異なる	4	3	2
例外処理	エラー時の処理が実行されない	4	4	3
⋮	⋮	⋮	⋮	⋮

図 7 不具合管理表の要約

Fig. 7 Summary of bug list detected in testing phase

設計，セキュリティ，例外処理，…を選んだ．そして，選んだ欠陥種別がレビューで重点的に見るべき観点となる．

3.2.7 欠陥検出のためのシナリオ作成

アナリストが観点を選択し終えたら，モデレータは各観点に対してシナリオを作成する．誰の視点でどの点に注意してレビューをすればいいかなど，着眼点や読み進め方を定義する．

表 7 は，観点リストの例である．DB 設計に関する欠陥を検出するためのシナリオの場合，開発者の視点では編成ついて，顧客の視点では管理するデータの項目に不備はないかに着目してレビューを行うようにというシナリオを作成する．

3.2.8 レビューアへの観点の割り当て

モデレータは，レビューアの能力や経験の程度を考慮し

表 7 観点リスト

Table 7 List of viewpoint

観点	シナリオ
DB 設計	開発者の視点で，DB 編成に着目する．正規化・主キーの設定・各属性の型の定義等に誤りや漏れがないかを確認．顧客の視点で，管理するデータに不備はないかを確認．
セキュリティ	顧客の視点で，ユーザ確認のための認証処理の数は妥当か（認証が多すぎないかなど）を確認．開発者の視点で，パスワードの定義（文字数や入力時の表示方法など）に不備はないかを確認．
⋮	⋮

表 6 欠陥データ要約表
Table 6 Summary table of the past data

欠陥種別	欠陥の概要	修正工数	見積修正工数	欠陥の合計	影響度	優先度
DB 設計	各テーブルの定義に不備があるため，ＳＱＬ文の命令が正しく実行されない	3.3	6	4	4	5
セキュリティ	アクセス権限が定義されておらず，機能の利用可能者が不明確である	7	11	3	5	5
計算処理	期日と料金の計算処理の設定に誤りがある	4	6	9	4	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮

て観点を割り当てる．データベース設計の経験が多いレビューア r_1 には DB 設計，例外処理の 2 つの観点を割り当てたり，セキュリティは重要な観点なのでレビューア r_2, r_3 の 2 人に割り当てるなど，観点の個数やレビューアの人数も考慮して全ての観点をレビューアに割り当てる．

3.2.9 観点に沿ったレビューの実施

レビューア r_1, r_2, r_3 は，割り当てられた観点に沿ってレビューを行う．シナリオを参考に，観点に該当する欠陥を検出する．シナリオに書かれていること以外でも，レビューア自身が気づいた欠陥があった場合は指摘する．

4. 考察

提案手法により，観点を設定することで検出すべき欠陥が明確になることが期待される．観点が無かった場合と比べてレビューの目的が明らかになるため，闇雲に欠陥を探すよりも効率的な作業ができると考えられる．

修正工数とシステムに与える影響度を考慮して観点を設定しているので，観点に該当する欠陥を検出することで修正工数の削減や深刻な欠陥の早期修正につながることが期待できる．

システマチックに欠陥種別を決定することは，レビューアの間や開発に関わる関係者の間でソフトウェアにとって何が大切かを合意することにもつながる．“セキュリティが大事” とレビューア個人が考えて，レビューを実施するよりも，具体的な過去のデータにもとづいて関係者がその危険性を十分に理解した上でセキュリティの重要性を意識するほうが，複数メンバによる相互確認につながることが期待される．同様にそれほど優先順位の高くない指摘に長い時間をかけることが減ることが期待される．

現時点では，要約はアナリスト個人の経験や知識に依存する作業が多いため，アナリストによっては，適切に欠陥種別を選択することができない可能性がある．要約するレビュー指摘，不具合データ，欠陥データの選択や要約の方法，工数の見積りをシステマチックに実施する方法を検討することは今後の重要な課題の 1 つである．

5. まとめ

本稿では，レビューにおいて優先的に検出すべき欠陥 (重点検出欠陥) を選択する手法を提案した．提案手法では，過去のレビュー指摘データ，不具合データをもとに欠陥を抽象化することにより，レビューア個人の感覚に依存せず，レビューでの重点検出欠陥を設定することができる．重点検出欠陥は，欠陥の修正工数と欠陥がシステムに与える影響を考慮して選択するので，レビュー時間が限られている場合や複数のレビューアによる問題検出の場合に，どのような問題から検出するかを合意しやすく，明確になる．

本稿では，レンタルショップの支援システムを想定し，提案手法の利用シナリオを示した．今後は，実際に過去のレビュー指摘データ，不具合データから重点検出欠陥を選択できるかどうかを確認する．また，選択した重点検出欠陥をレビューで検出することにより，効果が得られるかどうかを調べる予定である．あわせて，欠陥の要約や修正工数の見積りをより体系的な方法に洗練する．

謝辞 本研究は文部科学省科学研究補助費（基盤研究 B: 課題番号 23300009）による助成を受けた．

参考文献

[1] V. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard, M. Zelkowitz: The Empirical Investigation of Perspective-based Reading, Journal of Empirical Software Engineering, vol.2, no.1, pp.133-164(2006)
[2] A. Porter and L. Votta: An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspections, In Proceedings of the 16th International Conference on Software engineering, pp. 103-112(1994)
[3] A. Porter, L. Votta, V. Basili: Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment, IEEE Transactions on Software Engineering, vol.21, no.6, pp. 563-575(1995)
[4] T. Thelin, P. Runeson, B. Regnell: Usage-based Reading: An Experiment to Guide Reviewers with Use Cases, Information and Software Technology, vol. 43, no. 15, pp. 925-938(2001)