欠陥に対する理解の深化による再発予防方法の提案

～ 欠陥情報共有を通じた組織的な再発予防策 ～

A proposal of recurrence prevention measurements by deeply understanding of the defects

- Measures to prevent recurrence through the defect information sharing -

主査　：細川 宣啓　日本アイ・ビー・エム（株）
副主査：永田 敦　ソニー（株）
研究員：田中 裕大　（株）東芝
佐藤 俊之　ソーバル（株）
中村 紀裕　テックスエンジソリューションズ（株）
田村 光義　サイバートラスト（株）

研究概要

ソフトウェア開発において,不具合の再発予防を目的として,その不具合の分析を行っている組織もある.しかし,対策が効かず同種の欠陥が除去されず不具合を再発させてしまうことがある.これは,多くの欠陥は個人と集団の両方に起因するにも関わらず,現在の分析手法では,個人に起因する部分のみへの再発予防策に偏っており,再発予防策が不十分であるためだと想定される.有効な再発予防策を立案するには,欠陥が混入するメカニズムを理解する必要がある.本研究では,欠陥情報共有を通じた組織的な再発予防方法を提案し,有効性を確認する.

Abstract

In the software development, some organizations analyze incidents and take measures to prevent recurring the incidents again. However, some incidents recur because the measures don't work. Though many defects are caused by both of an individual and a group, current analytical methods indicate for only individual causes. So current measurements are insufficient. In order to make valid measures to prevent the recurrence, it is necessary to understand the mechanism by which defects are mixed. In this study, we propose organizational recurrence prevention measurements through the defect information sharing, and we confirm effectiveness of recurring prevention.

1. はじめに

1.1 背景

日々のソフトウェア開発の中で,多くの欠陥が発見・修正されている．発見された欠陥は,不具合管理ツールなどを用いて,組織内で管理・蓄積されている．これらの欠陥の中には,同じメカニズムで繰り返し発生するものも多い.発見された不具合を分析し,原因に直接対策を立案することで,再発予防に取り組んでいる組織もあるが,対策が効かず同種の欠陥が再発してしまうことも少なくない．この場合,欠陥を防げないだけでなく,効果の無い対策にコストを費やすことになり無駄が大きい.そこで本研究では,欠陥への対策が効かない原因についての検討を行い,有効な再発予防方法の提案を行う.

本稿においては,用語を以下の表 1 のように定義する.

表 1. 用語定義

用語	意味
不具合	ソフトウェアが期待結果を満たしていない状態.またはその箇所．障害と同義.
欠陥	ソフトウェアの成果物に含まれる,不具合を引き起こす原因となる記述.
欠陥情報	特定の欠陥に関する情報.「事象」,「修正方法」,「過失」など多くの要素を含む.

1.2 研究の目的

現在,ソフトウェア欠陥の原因分析には,なぜなぜ分析が用いられることが多い．なぜなぜ分析は,問題とその問題を引き起こした要因に対し,繰り返し「なぜ」と質問を繰り返しながら真の原因を突き止めていく手法である．しかし,「なぜ」という質問は,回答者を責めているように感じさせてしまう質問である(文献[1])ため,なぜなぜ分析による原因分析は,欠陥の原因が回答者個人の問題であると結論付ける傾向にあると言われている(文献[2]).実際に，研究員の所属する組織において行われている「なぜなぜ分析」の中には，欠陥の原因が個人にあると結論付けられている分析票が多数存在した.

一方,研究員の欠陥に関するディスカッションにおいて,欠陥の混入原因について,以下の意見が得られた.

- 知識不足などの個人のミスにのみ起因する欠陥は少なく,コミュニケーションのミスなどの複数人(集団)が関わる状況が欠陥を発生させる一因になっている様子が散見される

つまり，欠陥分析を行う際は，集団活動が原因の一端を担っている欠陥に対し個人に起因する部分のみへの対策を行うことで，集団に起因する部分への対策が不十分とならないようにする必要がある.そのためには，欠陥が混入するメカニズムを正確に理解し,共有することが必要となる.

そこで本研究では,下記の 2 つの RQ を設定する．これにより,プロジェクトで発見する欠陥が個人と集団どちらかのみに起因するか,もしくは両方に起因するかの確認を行い,再発予防策の立案に繋がる欠陥混入メカニズムを理解するのに必要な欠陥情報について検討を行う.

RQ1 ： プロジェクトで発生する欠陥は個人に起因するか？集団に起因するか？
RQ2 ： 欠陥混入メカニズムの理解の促進にはどのような情報の共有が必要か？

欠陥の発生が,個人だけでなく集団にも起因する場合,個人への対策のみではなく,プロジェクトや組織といった集団単位での対策も同時に行う必要があるといえる.

2. 関連研究

「Project Fabre」は,ソフトウェアテストシンポジウム 2013 東京において,「欠陥モデリング」を定義し,欠陥を可視化・抽象化する手法を提案している(文献[3])．欠陥モデリングにおける欠陥の要素の定義を表 2 に示す.また,表 2 の各要素を用いた欠陥モデルの構成イメージを付録 1 に記載する．欠陥モデリング手法を利用した欠陥予測についても議論されている(文献[4]).

本研究では,実験において欠陥の特性を表現するために,欠陥モデリングを利用した.

表 2. 「Project Fabre」の欠陥モデリングにおける欠陥の要素の定義

誘発因子	成果物の中に含まれる,人間の思考の誤りを誘発する“トリガー”となる要素のこと.誘発因子が存在すれば,開発者能力・経験・技術力と関係なく過失が引き起こされやすくなる.
過失因子	人間の思考や判断の誤りそのもののこと.欠陥は過失因子の集合（=連続）として生み出される.
増幅因子	過失の連鎖を助長し,欠陥の混入確率を増幅させる要素. 多くは定量的に測定可能である.外乱・環境特性ともいう.
欠陥	成果物に含まれた人間の思考の過ちが具現･表出化したもの.不具合・障害等の「現象」を発生させる.
表出現象	欠陥によって引き起こされる不具合・障害.多くは定量的に測定／加算可能.

3. 提案

3.1 今回の提案

本稿では欠陥情報の共有を通じた組織的な再発予防方法を提案する．1.1 で記述しているように,一般的に欠陥混入原因は開発者個人に依存するものと考えられているため,再発予防策も個人に依存した対策に偏りがちである．また，欠陥が発生したプロジェクトの全容を把握せずに欠陥が発生した瞬間の局所的な情報のみで対策を立案していることも個人に依存した対策に偏りがちになる原因の 1 つであると考えられる．

本稿では局所的な情報に基づいた再発予防策の立案を打破するために，プロジェクトの体制や環境である周囲の状況（以下，これを「環境要因」と呼ぶ）に着目し，欠陥情報の共有の際に環境要因を付加することで欠陥混入発生メカニズムの理解促進を図る．欠陥混入発生メカニズムを正確に理解させることで対策すべき対象者を個人から複数人数,すなわち開発者集団に目を向けさせ，個人に依存した再発予防策からの脱却を図るための組織的な欠陥再発予防方法を提案する．

また，欠陥情報の共有において過失の他に環境要因に着目した検証を行い，本稿中の測定データより,立案された対策内容から欠陥の起因元について考察する．

最終的に,検出された欠陥に対して集団で発生させたという認識を持ってプロジェクトに臨むことの重要性とプロジェクト関係者全体での予防努力の必要性を示す．

3.2 検証内容

3.1 の提案内容の有効性を示すために以下の検証を行う．

組織によって障害票の粒度は異なるが,本稿では「障害票から欠陥情報を抜粋した状態」と本稿で提案する「過失と環境要因の組み合わせを付加した状態」を以下のように定義する．

欠陥情報から「現象」「欠陥」「過失」になる部分を抜き出し,図 1 の「(a) モデル化前」の状態に整理したものを「障害票から欠陥情報を抜粋した状態」として扱う（以下,これを「モデル化前」と呼ぶ）．また,図 1 の「(b) モデル化後」のように，モデル化前の状態に環境要因を付加したものを「過失と環境要因の組み合わせを付加した状態」として扱う（以下,これを「モデル化後」と呼ぶ）．なお,モデル化前,モデル化後ともに「Project Fabre」の欠陥モデリングの定義を基に図式化している．

図式化したモデル化前の欠陥情報とモデル化後の欠陥情報を共有し,再発予防策のアンケートを実施する．アンケート実施後にモデル化前とモデル化後で立案された各々の再発予防策が個人と集団のどちらを対象とした対策であるか分類し,変化を比較する．

当検証はモデル化前とモデル化後の欠陥情報の差異である環境要因こそが立案された対策の変化をもたらしたことを示す.従って,モデル化後の欠陥情報がモデル化前の欠陥情報より「集団での再発予防対策」の発案を促すものであるとするならば,「集団での再発予防対策」の発案に環境要因の付加が有効であることを示すことが可能である.

また,対策方法を立ててもらう際には,目的や誰に対する立案対策であるかを具体的な内容にするために 5W1H を入れてもらうことを制約として設けた.

5W1H を用いることでそれぞれ以下の内容を明示的にすることを目的としている.

①Why（なぜ）: 何を防ぐ目的で立てた対応策であるかを明示する項目.

②Who（誰が）: 対応策を実施する作業者を明示する項目.

③Where（どこで）: どのようなシステムや環境においての対応策であるかを明示する項目.

④When（いつ）: 作業工程のどのタイミングにおいての対応策であるかを明示する項目.

⑤What（何を）: 対応策が何に対して実施するものであるかを明示する項目.

⑥How（どのように）: ⑤What の内容に対し，具体的にどのように作業・行動を行うかを明示する項目.

4. 実験

4.1 実験方法

3.2 で記述した検証内容を確認するために,研究員が所属する組織にて,以下の手順で実験を行った.

【手順 1】

研究員が所属する組織にて発生した欠陥情報を収集する.

収集した 2 ケースの欠陥情報から機密情報を除いて実験を行う.ケース 1 は図 1 参照,ケース 2 は図 2 を参照.

【手順 2】

手順 1 の欠陥ケースの障害票に記載されている情報から「現象」「欠陥」「過失」となる部分を抜き出し,「Project Fabre」の「欠陥モデリング」定義を用いて図式化する.（モデル化前）

【手順 3】

手順 2 のモデル化前の図に「Project Fabre」の「欠陥モデリング」定義を用いて環境要因を付加する.（モデル化後）

なお,モデリングをする際には「過失因子」は人の誤り,「誘発因子」は本実験では環境要因という前提でモデリングする. また,環境要因はプロジェクト関係者へのヒアリングを行うことで誘発因子として追記する.

また,1 モデル 1 過失とするため,過失が 2 つ以上存在する場合は過失毎にモデルを分けて図を作成する.

【手順 4】

モデル化前の欠陥情報を実験協力者に共有し,自身のプロジェクトでの再発予防策を立案してもらう.（モデル化前）

再発予防策の立案は立案策の主旨を明確にするために,表 3 の 5W1H 欠陥予防対策シートを基に各項目を埋めてもらう形式にする.

また,実験協力者は実験に使用する欠陥が発生したプロジェクトとは関係のない人を選定し, 協力者 12 名で実験を行った.

【手順 5】

手順 4 同様にモデル化後の欠陥情報を実験協力者に共有し,自身のプロジェクトでの再発予防策を立案してもらう.（モデル化後）

再発予防策の立案は立案策の主旨を明確にするために,表 3 の 5W1H 欠陥予防対策シー

トを基に各項目を埋めてもらう形式にする．

なお，実験協力者は手順 4 のアンケートに回答した人と同様の人を対象とした．

【手順 6】

手順 4 と手順 5 のアンケート結果を集計し，アンケート内容の再発予防策から，「Who（誰が）」「What（何を）」「How（どのように）」の記載箇所を抜き出して内容を要約する．

【手順 7】

手順 6 で要約した各再発予防策の対策概要を「対策分類」として記載する．

【手順 8】

手順 7 で記載した「対策分類」を基に再発予防策を「個人向けの対策」であるか，「集団向けの対策」であるかを観点に分類する．

表 3. 5W1H 欠陥予防対策シート

No	モデル化	なぜ(Why) (導入する目的)	誰が(Who) (対策を行う人)	どこで(Where) (対策を行う対象・環境・状況)	いつ・どのタイミングで(When) (作業工程)	何を(What) (具体的な内容)	どのように(How) (解決方法)	補足
例	前or後	○○を防ぐために ○○をするために	○○が	○○の開発やシステムに対して ○○の状況に対して	○○する前(後)に	○○を	○○する	
1	前							
2								
3								
4	後							
5								
6								

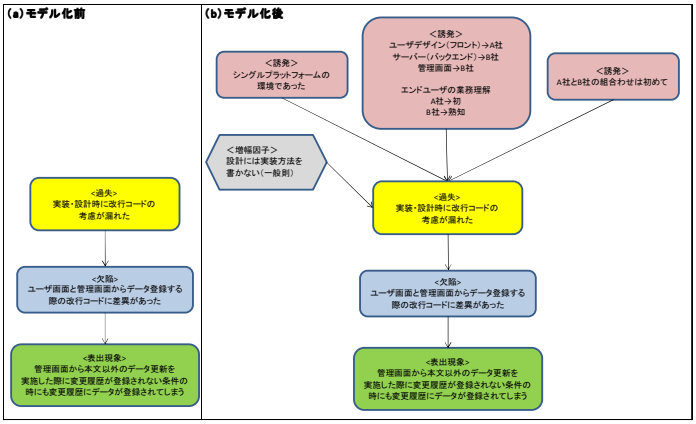

図 1. 検証に用いた欠陥ケース 1

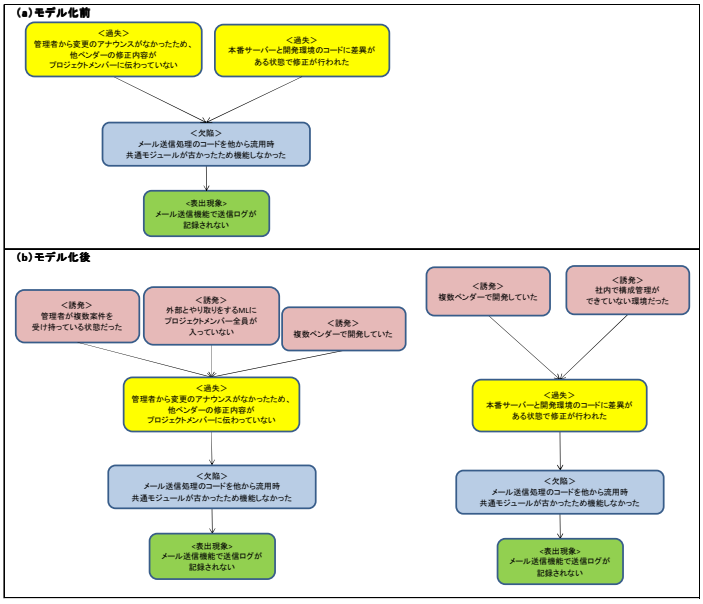

図 2. 検証に用いた欠陥ケース 2

4.2　実験結果

欠陥情報の内容によって立案された再発予防策を「個人向けの対策」と「集団向けの対策」で分類し，モデル化前とモデル化後での対策の変化について確認する．

欠陥ケース 1 の再発予防策の分類結果を表 4 に示した（詳細は付録 4 参照）．「モデル化前」の「個人向けの対策」と「集団向けの対策」は，ほぼ同等の割合を占めているが，図 3 の円グラフで示す通り，「モデル化後」は「集団向けの対策」が 80%を超える結果となり割合が上昇した．また，具体的な対策分類を比較した際の大きな変化として，「モデル化前」に存在しなかった「他社との共同作業」（12 件）が全体の 40%を占めたことが挙げられる．

表 4. 欠陥ケース 1 の対策分類表

欠陥ケース1

障害要からの対策		
個人向けの対策（47.2%）		17
	個人のレビュー作業	8
	テストによる除去	5
	個人の確認作業	3
	個人のコーディング手法	1
集団向けの対策（52.8%）		19
	ルール作成	9
	複数名でのレビュー	5
	情報共有	3
	環境構築	1
	教育	1

欠陥モデルからの対策		
個人向けの対策（16.7%）		6
	テストによる除去	4
	個人の確認作業	1
	個人の作業改善	1
集団向けの対策（83.3%）		30
	他社との共同作業	12
	情報共有	10
	ルール作成	6
	他チームと調整	1
	複数名でのレビュー	1

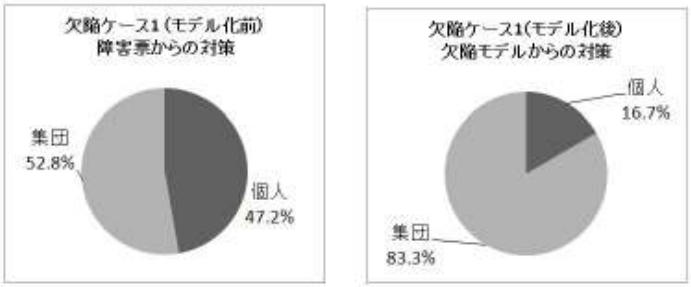

図 3. 欠陥ケース 1 の対策分類グラフ

欠陥ケース 2 の再発予防策の分類結果を表 5 に示した（詳細は付録 5 参照）．「モデル化前」の「個人向けの対策」と「集団向けの対策」は，欠陥ケース 1 と同様にほぼ同等の割合を占めているが，図 4 の円グラフで示す通り，「モデル化後」は「集団向けの対策」が 90%に近い結果となり，割合が上昇している．また，具体的な対策分類を比較した際の変化として，対策内容に「環境整備」（8 件），「役割分担」（5 件）とそれぞれ「モデル化前」では存在しなかった新しい対策分類が増える結果となった．

表 5. 欠陥ケース 2 の対策分類表

欠陥ケース2

障害素からの対策		
個人向けの対策（45.5%）		20
	コードの管理	9
	テスト	6
	ルール順守	5
集団向けの対策（54.5%）		24
	情報共有	10
	計画作成	5
	他プロジェクトと調整	4
	コードの環境整備	3
	教育	2

欠陥モデルからの対策		
個人向けの対策（10.9%）		5
	ルール順守	4
	作業改善	1
集団向けの対策（89.1%）		41
	情報共有	14
	環境整備	8
	他プロジェクトと調整	6
	計画作成	5
	役割分担	5
	教育	2

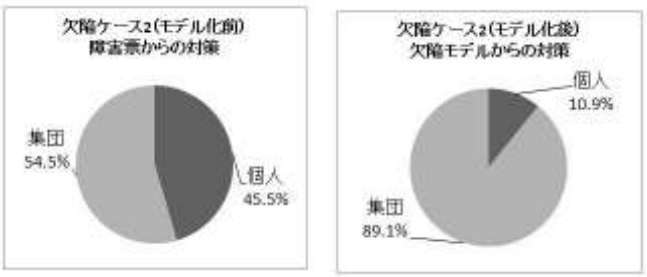

図 4. 欠陥ケース 2 の検証結果

欠陥ケース 2 の「モデル化前」と「モデル化後」を比較した結果,欠陥ケース 1 同様に立案される再発予防策が「個人向けの対策」から「集団向けの対策」へと割合が変化した.

欠陥ケース 2 でも「集団向けの対策」の割合が 34.6%ほど上昇した他に対策内容に「環境整備」(8 件),「役割分担」(5 件) とそれぞれ「モデル化前」では存在しなかった新しい対策分類が増える結果となった.

4.3 考察

本研究の実験結果に対して,考察を次に示す.

RQ1 : プロジェクトで発生する欠陥は個人に起因するか？集団に起因するか？

欠陥情報を基に欠陥再発予防策を検討した結果,表 4,表 5 に示したとおり,「個人向けの対策」と「集団向けの対策」の両方が存在することが分かった.このことから, プロジェクトで発生する欠陥は「個人に起因する欠陥」と「集団に起因する欠陥」の両方が存在する傾向にあると考えられる.

RQ2 : 欠陥混入メカニズムの理解の促進にはどのような情報の共有が必要か？

本稿中で障害票の情報と置いた「モデル化前」の情報で再発予防策を立案した場合,「個人向けの対策」が半分近くを占めた.一方,「環境要因」を含めた「モデル化後」の情報から再発予防策を立案した場合,「個人向けの対策」は 20%以下となり,「集団向けの対策」の割合が 80%以上を占める結果となった.

この結果は「環境要因」を組み合わせて欠陥情報を共有することで, 欠陥混入メカニズムの理解が促進されたためと考えられる.また,理解の促進により立案者がプロジェクトの全体像を把握できたため,対策の割合に変化が発生した.これは,「環境要因」が欠陥混入メカニズムの理解促進に必要な共有すべき情報の 1 つであることを示している.

図 3,図 4 で示した通り,「環境要因」の情報を付与する前の「モデル化前」の情報を基に再発予防策を立案した場合においても「集団向けの対策」は「個人向けの対策」と比較して半分を占めている.これは,「モデル化前」の情報を基に再発予防策の立案を行う場合も「集団に起因する欠陥」に着目して検討が行われており,兆候を掴んでいるといえる.しかし,「環境要因」等の欠陥を混入させた背景である欠陥混入メカニズムを把握する情報が「モデル化前」の情報には明確には記述されていない. そのため,「個人に起因する欠陥」として扱われ,有効な対策を打つことができなかったと思われる.

この課題を解消するために, 再発予防策を立案した際に「個人向けの対策」に偏っていた場合,「集団向けの対策」を行う必要が無いかをプロジェクト関係者全体で再検証を行い,立案された再発予防策の妥当性を確認する取組みが必要である.また,欠陥予防は,成果物に欠陥を直接埋め込むわけではないプロジェクトマネージャーやプロジェクトリーダーをはじめ,プロジェクト関係者全体の予防努力も必要である. したがって,検出された欠陥に

対して集団で発生させたという認識を持ってプロジェクトに臨むことが考え方として理解される必要がある.

5. おわりに

5.1 まとめ

本研究では，欠陥情報の共有を通じた組織的な再発予防方法を提案した.検証結果から,「環境要因」を組み合わせた欠陥情報の共有を行うことで,欠陥混入メカニズムの理解を促進させることを示した.また,欠陥混入のメカニズムの理解促進が「組織的な再発予防策」の立案を助長することを示した.

5.2 今後の展望

今回実施した 2 件の実験において,比較的近い数値の実験結果が得られた．今後は対象とする欠陥情報の数,種類を増やすことで,この結果が一般的に成り立つものなのかの検証を行いたい.

また,現在,多くの組織では,欠陥情報を活用できていないだけでなく,欠陥を埋め込んだ開発者への攻撃材料として欠陥情報が利用されることも少なくない.これにより,開発者やプロジェクトは,欠陥データの共有に消極的である．研究員は,欠陥情報の共有が組織にとって「価値があるもの」であると考えている．今回の提案により,組織内での欠陥データの価値が見直され,管理・活用されるようになることを期待したい.

6. 参考文献

[1] Eric E. Vogt, Juanita Brown, and David Isaacs, THE ART OF POWERFUL QUESTIONS: Catalyzing Insight, Innovation, and Action, Whole Systems Associates, CA, USA, 2003

[2] 永田敦,「反復プロセスと欠陥モデリングによるソフトウェア要因分析の改善 アジャイルな RCA の導入とその効果」ソフトウェア・シンポジウム 2015 和歌山

[3] 細川宣啓,西康晴,嬉野綾,野中誠,原佑貴子,「過失に着目した欠陥のモデリング-バグ分析はなぜうまくいかないのか？」ソフトウェアテストシンポジウム 2013 東京 セッション C4「不具合情報の活用」

[4] 細川宣啓,永田敦,柏原一雄,岡本晃,鈴木裕一郎,田村光義,東久保理江子,保栖真輝,「ソフトウェア欠陥予測アルゴリズム」,日本科学技術連盟 SQiP 研究会,2014

付録 1.　「Project Fabre」の欠陥モデリング

「Project Fabre」が定義する欠陥が持つ要素（表出現象・欠陥・過失因子・誘発因子・増幅因子)をモデリング手法で可視化,抽象化した欠陥モデル図を図 5 として以下に示す.

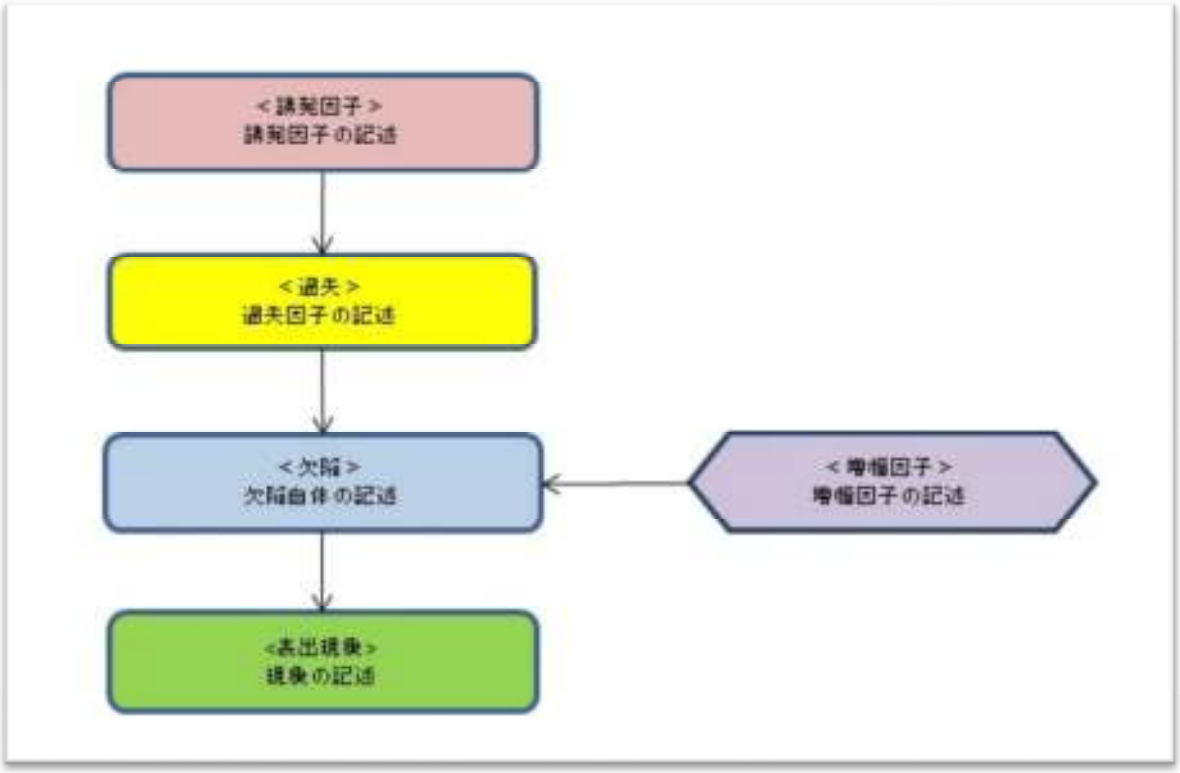

図 5.　「Project Fabre」の欠陥モデリング記載例

付録 2.　実験使用欠陥モデル（欠陥ケース 1）

「4.1 実験方法」の【手順 2】,【手順 3】において作成した欠陥ケース 1 のモデル化前とモデル化後の拡大図を以下に示す.

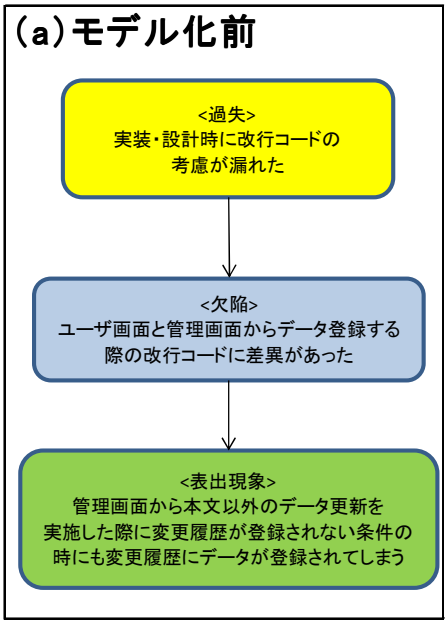

図 6.　検証に用いた欠陥ケース 1（モデル化前）

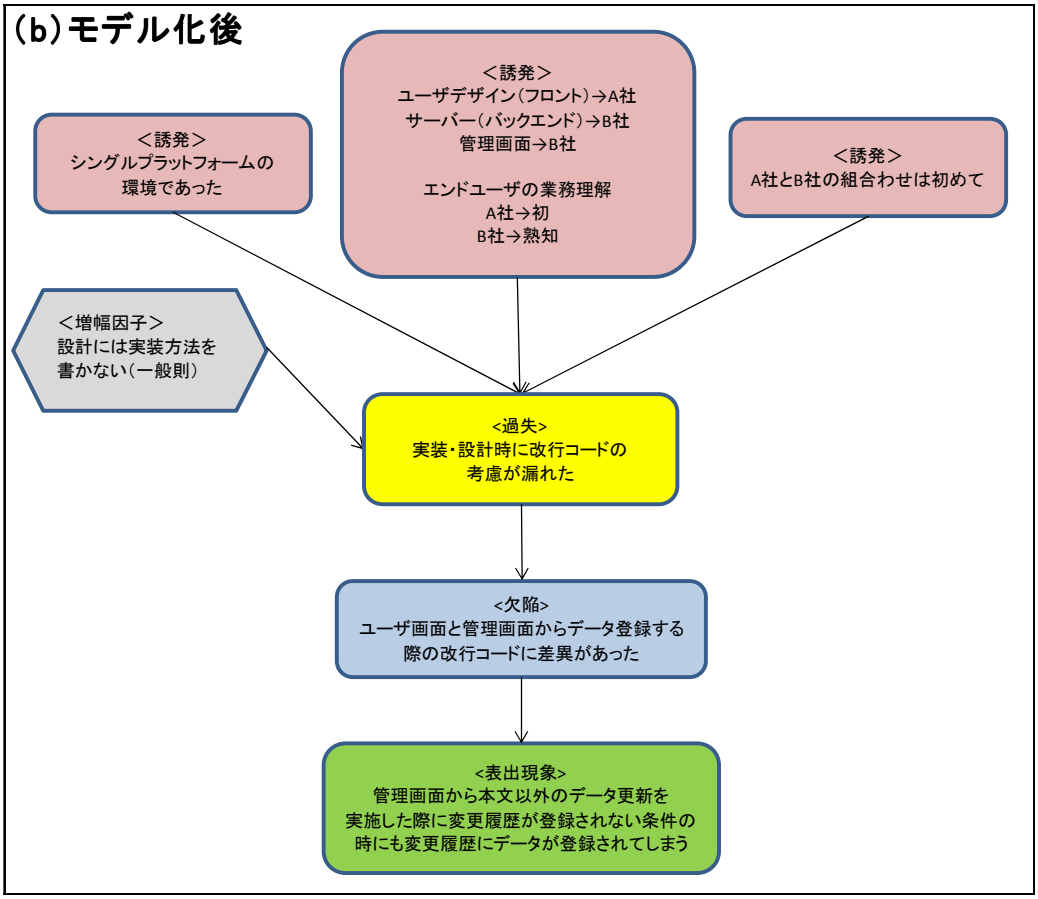

図 7.　検証に用いた欠陥ケース 1（モデル化後）

付録 3.　実験使用欠陥モデル（欠陥ケース 2）

「4.1 実験方法」の【手順 2】,【手順 3】において作成した欠陥ケース 2 のモデル化前とモデル化後の拡大図を以下に示す.

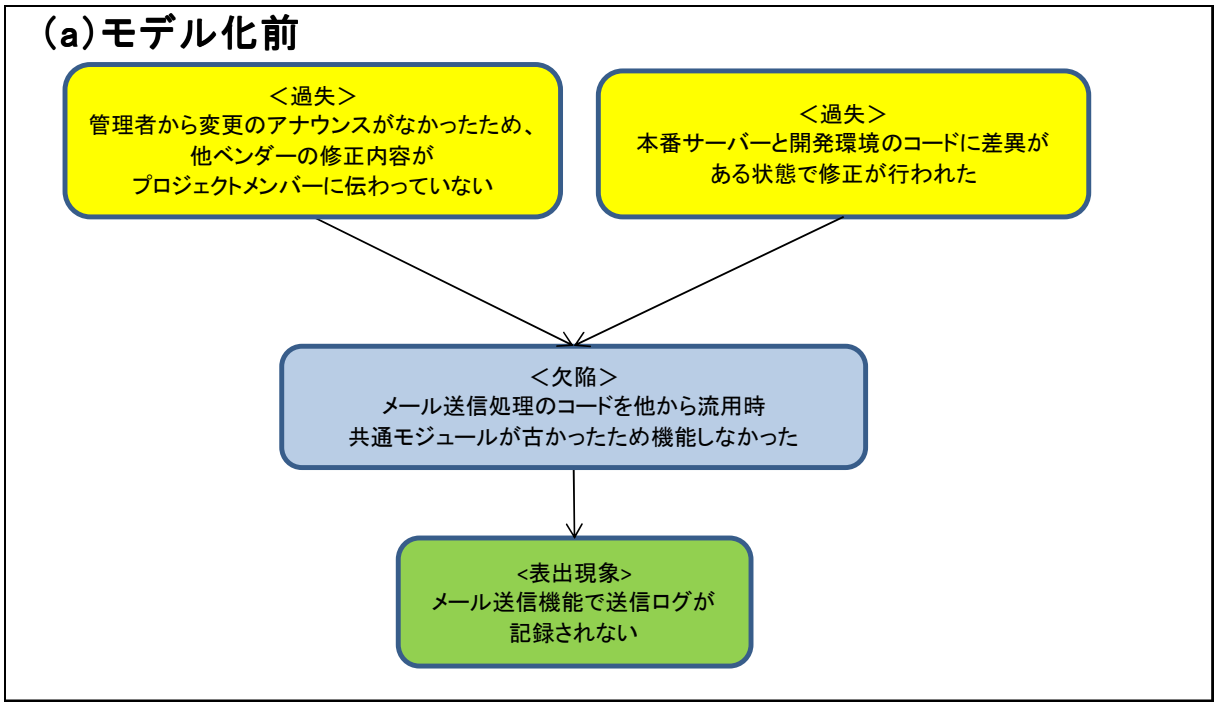

図 8.　検証に用いた欠陥ケース 2（モデル化前）

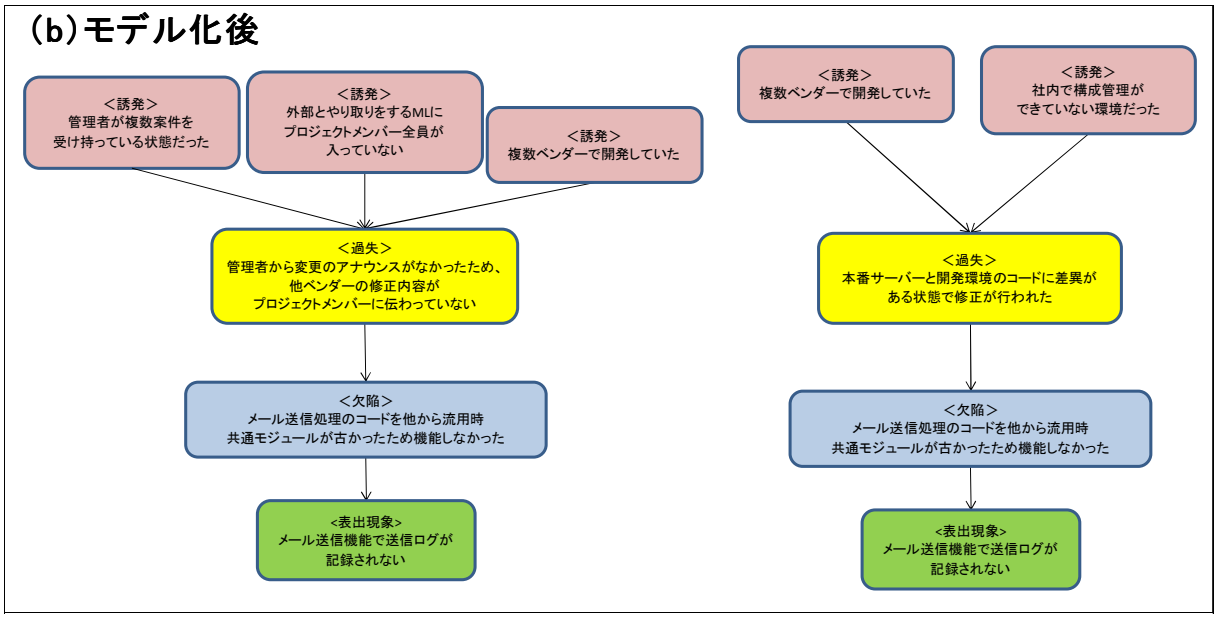

図 9.　検証に用いた欠陥ケース 2（モデル化後）

付録 4.　アンケート結果（欠陥ケース 1）

「4.1 実験方法」の【手順 5】において実験協力者から回収した欠陥ケース 1 の再発予防策を表 6, 表 7 として以下に示す.

表 6. 欠陥ケース 1 モデル化前から再発予防策

【モデル化前】

対応者	対策分類	誰が	何を	どのように
個人向けの対策	個人のレビュー作業	開発者が	関係箇所の洗い出しを	設計書およびソースコードを基に確認する
個人向けの対策	個人のレビュー作業	開発者が	設計書及びソースコードを	修正する
個人向けの対策	個人のレビュー作業	開発者が	過去の類似システムの不具合情報を	収集して気をつけるべきリストとして整理しておく
個人向けの対策	個人のレビュー作業	開発者が	システム内の同様の履歴登録記録を	ソースコードレビューする
個人向けの対策	個人のレビュー作業	コード作成者が	コードを	他の箇所に再利用可能か確認する
個人向けの対策	個人のレビュー作業	コード作成者が	ソースコードを	レビューして、他箇所に影響がないかを確認する
個人向けの対策	個人のレビュー作業	コード作成者が	類似コード箇所を	コード解析する
個人向けの対策	個人のレビュー作業	開発メンバが	影響箇所を	特定しておく
個人向けの対策	テストによる除去	開発者が	テスト対象を	画面とする
個人向けの対策	テストによる除去	テスト担当者が	対象部分を	テストする
個人向けの対策	テストによる除去	テスターが	¥r, ¥n, ¥r¥n それぞれの動作を	確認する
個人向けの対策	テストによる除去	テスト担当者が	制御コードを	期待された値が入力されているか確認する
個人向けの対策	テストによる除去	テスト担当者が	両方の画面を	テストする
個人向けの対策	個人の確認作業	開発者が	改行コードの形式を	ドキュメントで確認する
個人向けの対策	個人の確認作業	品質保証メンバーが	改行コードの差異による影響を	確認する
個人向けの対策	個人の確認作業	開発者が	登録する改行コードを	ピアリングで確認する
個人向けの対策	個人のコーディング手法	プログラマが	ソースコードを	コード中に改行コードを入れない仕組みを造る
集団向けの対策	ルール作成	開発メンバが	改行コードのルールを	統一させる
集団向けの対策	ルール作成	開発メンバが	改行コードの仕様を	決める
集団向けの対策	ルール作成	プログラマが	改行コードの入れ方を¥nか¥r¥nに統一する	¥nか¥r¥nに統一する
集団向けの対策	ルール作成	PLが	コーディングルールを	決めておく
集団向けの対策	ルール作成	開発チームが	コーディングルールを	決めておく
集団向けの対策	ルール作成	開発リーダーが	コーディングルールを	作成する
集団向けの対策	ルール作成	開発担当者全員が	コーディングルールを	話し合う
集団向けの対策	ルール作成	プロジェクトリーダが	改行コードに対する	開発基準書を作成する
集団向けの対策	ルール作成	PMが	機構設計書を	作成する
集団向けの対策	複数名でのレビュー	実装した開発者以外が	コードを	レビューする
集団向けの対策	複数名でのレビュー	開発リーダーが	コードを	レビューする
集団向けの対策	複数名でのレビュー	複数の開発者が	テキスト処理に関する仕様を	ピアレビューする
集団向けの対策	複数名でのレビュー	複数の開発者が	ログ機能に関する仕様を	ピアレビューする
集団向けの対策	複数名でのレビュー	複数のテスト担当者が	テスト仕様に関する仕様を	ピアレビューする
集団向けの対策	情報共有	PMが	改行コード等のアプリ仕様を	メンバー全員に対して書面で周知して認識合わせを図る
集団向けの対策	情報共有	PMが	機構設計書を	メンバー全員に対して書面で周知して認識合わせを図る
集団向けの対策	情報共有	PMが	コーディング規約、サンプルコードを	メンバー全員に対して書面で周知して認識合わせを図る
集団向けの対策	環境構築	開発チームが	共通する処理を別々に作成せずに	共通化する
集団向けの対策	教育	プロジェクトリーダーが	コーディングルールを	ミーティングの議題にする

表 7. 欠陥ケース 1 モデル化後から再発予防策

【モデル化後】

対応者	対策分類	誰が	何を	どのように
個人向けの対策	テストによる除去	テスト担当者が	A社とB社で作成した画面を	動作比較する
個人向けの対策	テストによる除去	テスト担当者が	両方の画面を確認するための	テストを実施する
個人向けの対策	テストによる除去	テスト担当者が	仕様通りに動作するかを	確認する
個人向けの対策	テストによる除去	仕様作成者、設計者	A社、B社のI/Fについての試験項目を	先行してテストする
個人向けの対策	個人の確認作業	プロジェクトの開発メンバーが	エンドユーザごとの設計の差異を	確認する
個人向けの対策	個人の作業改善	開発チームが	実装方法を	明記する
集団向けの対策	他社との共同作業	A社、B社の開発者が	設計ドキュメントを	クロスレビューする
集団向けの対策	他社との共同作業	A社、B社の開発者が	中間生成物を	送付して内容に認識違いが無いか確認する
集団向けの対策	他社との共同作業	A社、B社の開発者が	B社の設計上の問題点を	レビューの場でカバーする
集団向けの対策	他社との共同作業	A社の開発者が	コーディングルールを	本プロジェクトと合っているかB社に確認する
集団向けの対策	他社との共同作業	B社の開発者が	A社のコーディングルールを	確認する
集団向けの対策	他社との共同作業	プロジェクトリーダが	インターフェースや実装時の注意点を	他のプロジェクトに確認する
集団向けの対策	他社との共同作業	委託元、A社、B社の3社が	設計ガイドラインルールを	ミーティングで確認する
集団向けの対策	他社との共同作業	開発関係者／SQAG/SEPGが	設計時のレビューチェックシートを	見直す
集団向けの対策	他社との共同作業	各会社が	コードを	レビューする
集団向けの対策	他社との共同作業	自社と他社の開発メンバーが	自社、他社の開発が影響する部分を	問題がないかレビューする
集団向けの対策	他社との共同作業	双方の開発者が	互いの仕様を	紹介し合う
集団向けの対策	他社との共同作業	別会社間が	コードを内容を	レビューする
集団向けの対策	情報共有	PMが	品質保証計画書を	メンバー全員に対し、書面で周知する
集団向けの対策	情報共有	プロジェクトチーム(PMまたはPL)が	情報を共有する場を	設ける
集団向けの対策	情報共有	開発者が	画面設計の共有を	ミーティングにて実施する
集団向けの対策	情報共有	開発者が	気になるリスクを洗い出し	洗い出して対策を議論する
集団向けの対策	情報共有	開発者が	開発規約に従っているか	チェックできるツールを提供する
集団向けの対策	情報共有	各会社が	プラットフォーム情報を	共有する
集団向けの対策	情報共有	仕様作成者、設計者	A社、B社のI/Fについての注意事項を	仕様書にリスト化する
集団向けの対策	情報共有	仕様作成者、設計者	A社、B社のI/Fについての要確認事項を	試験項目としてテスト仕様書にリスト化する。
集団向けの対策	情報共有	仕様作成者、設計者	利用者視点に立った注意事項を	仕様書にリスト化する
集団向けの対策	情報共有	仕様作成者、設計者	利用者視点に立った要確認事項を	試験項目としてテスト仕様書にリスト化する。
集団向けの対策	ルール作成	B社開が	コーディングルールを	作成する
集団向けの対策	ルール作成	PMが	品質保証計画書を	作成する
集団向けの対策	ルール作成	プロジェクトリーダが	改行コードを記述した開発基準書を	作成する
集団向けの対策	ルール作成	開発者が	改行コードの入れ方を¥n か¥r¥n に統一する	¥n か¥r¥n に統一する
集団向けの対策	ルール作成	開発者が	開発規約を	整備する
集団向けの対策	ルール作成	自社の開発者が	仕様のガイドラインを	作成する
集団向けの対策	他チームと調整	評価チームが	実装方法の仕様書記載を	依頼する
集団向けの対策	複数名でのレビュー	PLが	レビューアを	両社の開発メンバを対象とする

付録 5.　アンケート結果（欠陥ケース 2）

「4.1 実験方法」の【手順 5】において実験協力者から回収した欠陥ケース 2 の再発予防策を表 8,表 9 として以下に示す.

表 8. 欠陥ケース 2 モデル化前から再発予防策

【モデル化前】

対応者	対策分類	誰が	何を	どのように
個人向けの対策	コードの管理	P.開発者が	払い出すコードを	修正前に最新版を払い出す
個人向けの対策	コードの管理	開発者が	モジュールの差異を	マージする
個人向けの対策	コードの管理	PMが	開発環境とのコードの差異を	どちらが正しいか常に確認する
個人向けの対策	コードの管理	開発者が	開発環境とのコードの差異を	確認する
個人向けの対策	コードの管理	開発者が	変更点を	文書に記載する
個人向けの対策	コードの管理	PL, および, 変更管理担当者が	変更内容を	記録する
個人向けの対策	コードの管理	環境構築担当者が	各コードの差分を	文書化する
個人向けの対策	コードの管理	開発者が	本番環境と開発環境の差異を	リスト化して、確認する
個人向けの対策	コードの管理	開発担当者が	コードにおいて	最新版であることを確認する
個人向けの対策	テスト	コード作成者	適用したコードを	テストする
個人向けの対策	テスト	テスト担当者	コードを適用した後を	運用テストする
個人向けの対策	テスト	コード作成者	適用したコードを	テストする
個人向けの対策	テスト	モジュール管理者	本番環境＋修正モジュールを	動作確認テストを行う
個人向けの対策	テスト	開発者が	変更箇所に対して	探索的テストする
個人向けの対策	テスト	テスト担当者が	コードカバレッジを	計測して開発漏洩を検出する
個人向けの対策	ルール順守	開発者やテスターが	構成管理計画に基づいた成果物管理を	実践する
個人向けの対策	ルール順守	構成管理担当者が	コード更新時にチェックインするルールを	運用する
個人向けの対策	ルール順守	開発担当者が	変更内容を	コミットする
個人向けの対策	ルール順守	開発メンバが	ルールを	遵守する
個人向けの対策	ルール順守	開発者が	利用できない共通モジュールを	削除する
集団向けの対策	情報共有	P.開発者が	変更箇所を	アナウンスする
集団向けの対策	情報共有	PL, および, 変更管理担当者が	変更内容を	開発者に連絡する
集団向けの対策	情報共有	管理者が	修正内容を	確実に伝える手段を整える
集団向けの対策	情報共有	環境構築担当者が	各コードの差分を	連絡する(共有する)
集団向けの対策	情報共有	コードの変更をおこなった人が	変更内容を	口頭で伝える
集団向けの対策	情報共有	管理者が	周知方法を	見直す
集団向けの対策	情報共有	管理者	修正内容, 修正モジュール	PLパンリに連絡する
集団向けの対策	情報共有	PMが	情報を	全員に周知する
集団向けの対策	情報共有	管理者が	修正内容を	プロジェクトメンバーに確実に伝える
集団向けの対策	情報共有	メンバ全員が	管理者からの変更内容を	参照して、チケットをクローズとする
集団向けの対策	計画作成	開発者やテスターが	構成管理計画書を	作成する
集団向けの対策	計画作成	開発チームが	ルールを	決める
集団向けの対策	計画作成	P.開発者が	ルールを	作成する
集団向けの対策	計画作成	管理者が	構成管理担当者を	設定する
集団向けの対策	計画作成	開発リーダが	プロジェクトに関するプロセスを	策定する
集団向けの対策	他プロジェクトと調整	構成管理担当者が	他ベンダーが修正したコードを	入手する
集団向けの対策	他プロジェクトと調整	構成管理担当者が	構成管理システムの共有運用を	他ベンダーと施行する
集団向けの対策	他プロジェクトと調整	管理者が	変更連絡を	打ち合わせの場で伝えてもらうことを依頼する
集団向けの対策	他プロジェクトと調整	プロジェクトリーダーが	他ベンダーの修正箇について	ヒアリングする
集団向けの対策	コードの標準整備	管理者が	自社開発環境のコードを	常に最新の状態になるようにする
集団向けの対策	コードの標準整備	PMが	他ベンダの作成する共通モジュールを	自社内で開発に制約をかける
集団向けの対策	コードの標準整備	開発リーダが	開発環境を	コピーする
集団向けの対策	教育	管理者が	コード管理方法を	教育する
集団向けの対策	教育	PMが	本番環境へのメリースに	リリース手順書を提示させる

表 9. 欠陥ケース 2 モデル化後から再発予防策

【モデル化前】

対応者	対策分類	誰が	何を	どのように
個人向けの対策	ルール順守	管理者が	設定された構成管理ルールを	遵守する
個人向けの対策	ルール順守	全員が	構成管理ルールを	遵守する
個人向けの対策	ルール順守	モジュール管理者が	ソースの構成管理を	徹底する
個人向けの対策	ルール順守	開発者が	構成管理を	ルール通りに実施する
個人向けの対策	作業改善	管理者が	メール振り分けを	単体ごとに分類されるように設定する
集団向けの対策	情報共有	管理者が	外部とのやりとりやコードの修正内容を	開発者に送るプロセスを導入する
集団向けの対策	情報共有	PMが	プロジェクト情報を	メンバー全員に情報共有する
集団向けの対策	情報共有	P.関係者が	ML に通知が登録されているかを	確認する
集団向けの対策	情報共有	開発関係者が	担当ベンダーの修正情報を	相互にメンバ全員に送り込み
集団向けの対策	情報共有	開発チームが	修正内容を	通知する
集団向けの対策	情報共有	開発チームが	修正内容を通知するための外部MLに	全員参加させる
集団向けの対策	情報共有	ML管理者が	ML 更新メールを	送信する
集団向けの対策	情報共有	変更したベンダーの管理者が	変更内容と記述した文書の所在を	通知する
集団向けの対策	情報共有	会議出席者が	会議内容の報告を	作成して ML で通知を行う
集団向けの対策	情報共有	変更を行った人が	報告手段を	メール連絡と他の手段を併用する
集団向けの対策	情報共有	管理者が	開発担当者全員を	ML に登録する
集団向けの対策	情報共有	ML メンバーが	情報の閲覧を	徹底する
集団向けの対策	情報共有	管理者が	修正作業の共有を	ミーティングで実施する
集団向けの対策	情報共有	メンバ全員が	管理者からの変更内容を	参照して、チケットで履歴を残してクローズとする
集団向けの対策	環境整備	P.関係者(PJL, SEPG)が	構成管理環境を	整備する
集団向けの対策	環境整備	開発作業をしている複数ベンダーが	構成管理システムの共有運用を	検討する
集団向けの対策	環境整備	PMが	構成管理ツールを	導入させる
集団向けの対策	環境整備	PMが	構成管理のためのインフラを	整備させる
集団向けの対策	環境整備	管理者が	外部とのやりとりやコードの修正内容を	開発者に送るツールを導入する
集団向けの対策	環境整備	開発環境を構築する担当者	構成管理を	本番環境と開発環境で構成管理を行える環境を構築する
集団向けの対策	環境整備	開発リーダが	環境を	コピーする
集団向けの対策	環境整備	PMが	構成管理ツールを	導入する
集団向けの対策	新プロジェクトと調整	PMが	プロジェクト情報を	会社管理者と適宜で交換する
集団向けの対策	新プロジェクトと調整	開発する複数ベンダーが	更新されたコードベースラインを	相互に受付する
集団向けの対策	新プロジェクトと調整	開発する複数ベンダーが	会議を	行う
集団向けの対策	新プロジェクトと調整	管理者が	ML 登録者数をやすことを	承認元に依頼する
集団向けの対策	新プロジェクトと調整	開発する複数ベンダーが	共同レビューを	行い認識の違いを解消する
集団向けの対策	新プロジェクトと調整	開発リーダが	複数ベンダー間での情報共有方法を	確認する
集団向けの対策	計画作成	開発者が	構成管理のルールを	作る
集団向けの対策	計画作成	P.関係者(PJL, SEPG)が	メール環境と運用ルールを	整備する
集団向けの対策	計画作成	開発関係者が	チェックインする共通ルールを	決める
集団向けの対策	計画作成	管理者が	構成管理ルールを	設定する
集団向けの対策	計画作成	管理者が	ML の運用ルールを	決める
集団向けの対策	計画作成	管理者が	情報展開のルールを	決める
集団向けの対策	役割分担	PMが	管理業務を	サブプロマネに代行させる
集団向けの対策	役割分担	上位管理者が	PMの担当する範囲を	調整する
集団向けの対策	役割分担	開発関係者が	メーリングリストによる更新連絡を	複数名で定期的に確認する
集団向けの対策	役割分担	管理者が	構成管理担当者を	設定する
集団向けの対策	役割分担	管理者の上司が	作業負荷が軽減できるよう	業務状況して作業役割の変更を行う
集団向けの対策	教育	リーダが	構成管理方法を	教育する
集団向けの対策	教育	管理者が	コードの管理方法を	ミーティングの機会に取り上げて教育する