

OSS開発における不具合修正プロセスの 現状と課題

60151039 東 裕之輔

著者

- ▶ 伊原彰紀
- ▶ 大平雅雄
- ▶ 松本健一



背景

- ▶ OSS開発現場では本来不具合報告は歓迎されるものである。
- ▶ プロジェクトによっては一日数百件の不具合報告がある。
- ▶ 報告されたすべての不具合を短期間に処理するのは難しい。



背景

- ▶ 結果OSSプロジェクトでは不具合修正の長期化を招いている。
- ▶ OSSは社会の重要性が増す一方であり、長期化の解決策が望まれている。



本研究の目的

- ▶ 大規模OSSプロジェクトにおける不具合修正プロセスを明らかにすること。
- ▶ 不具合修正プロセスの効率化するための方法を検討することである。



研究の流れ

- ▶ Apache, Firefox, Eclipseの3つのOSSプロジェクト
- ▶ それぞれ不具合管理システムに記録される不具合修正の進捗情報を用いて作業時間を可視化、分析。



- ▶ 不具合修正プロセスの問題を発見する。

不具合管理システム

- ▶ 一般的なOSSの開発では、不具合に関する情報を共有するために



- ▶ などの不具合管理システム(BTS)が利用されている。
- ▶ BTS・・・(バグトラッキングシステム)



BTS

- ▶ これらのBTSは一つの不具合に一つの不具合票を作成し、個々の不具合修正進捗を管理するシステムである。
- ▶ Webを介して誰でも不具合の報告と修正作業進捗を閲覧できる。
- ▶ その不具合修正プロセスは？



不具合修正プロセスでの3つのフェーズ

▶ プロセスには3つのフェーズがある。

▶ 修正計画フェーズ

修正の不具合を報告する。



▶ 修正フェーズ

不具合を承認し、修正する。



▶ 検証フェーズ

修正作業を確認し、再修正を検討する。



BTSを用いた一般的な不具合修正プロセス

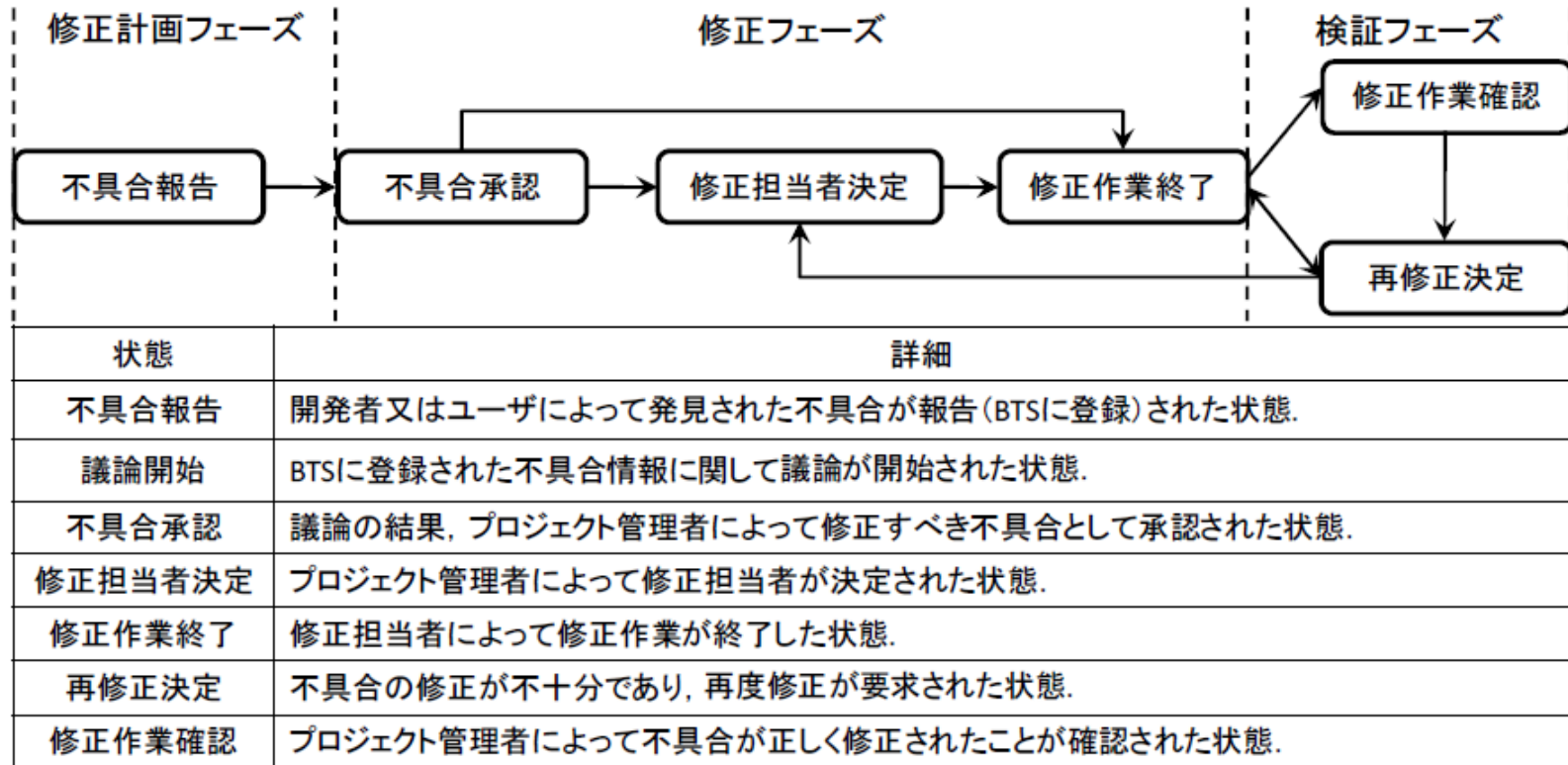


図2 BTS を用いた一般的な不具合修正プロセス

不具合管理の問題

- ▶ BTSに登録されている不具合の中には、修正作業が滞っているものもある。
- ▶ 修正に向けた議論や作業が開始されない。
- ▶ 修正担当者が決まらない。問題が複雑。
- ▶ 検証作業が開始されない。



不具合管理の問題

- ▶ このような不具合の中には、優先度や重要度が高い不具合が含まれていることも多く、プロジェクト管理者は検索機能を用いて定期的に調査しなければならない。
- ▶ しかし、プロジェクト管理者は修正が滞る原因となる作業を把握するには大量の不具合を相対的に比較検討しなければならない。



不具合修正プロセスの課題

- ▶ 修正に要する時間の長期化はOSSのリリース遅延やサポート遅延に繋がる。
- ▶ プロジェクト管理者が不具合修正プロセス中の各作業時間や各フェーズに要する時間、状態遷移のパターンを効率的に理解することが必要である。

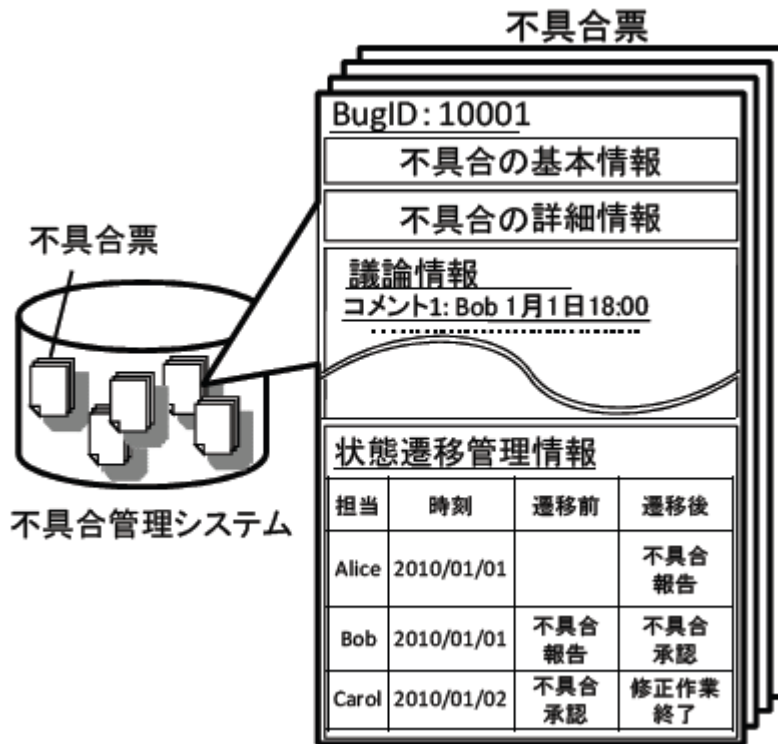


分析方法

- ▶ Bugzillaの不具合票から情報を取得し、以下のような図を順に作成していく。
 1. 状態遷移図時刻表の作成
 2. 状態遷移時間表の作成
 3. 作業時間表の作成
 4. 状態遷移図の構築



状態遷移時刻表の作成



(a) 状態遷移時刻表

BugID	No.	状態遷移前	状態遷移後	遷移時刻	コメント時刻
10001	(1)	不具合報告	不具合承認	1月1日 15:30	1月1日 13:30
	(2)	不具合承認	修正作業終了	1月1日 17:00	
	(3)	修正作業終了	不具合確認	1月2日 18:30	
10002	(1)	不具合報告	修正作業終了	1月4日 12:00	1月4日 11:00
⋮	⋮	⋮	⋮	⋮	⋮
24356	(3)	不具合解決	不具合確認	9月3日 18:00	9月3日 11:00
24357	(1)	不具合報告	不具合承認	9月7日 10:00	9月7日 13:00
	(2)	不具合承認	担当者決定	9月7日 12:00	
	(3)	担当者決定	修正作業終了	9月9日 14:00	
	(4)	修正作業終了	修正作業確認	9月9日 20:30	

状態遷移、遷移時刻、最初のコメントが投稿された時刻を取得し、(a)を作成する。

状態遷移時間表、作業時間表の作成

(b)各不具合票の各状態
遷移にかかる時間を算
出し、作成する。

(b) 状態遷移時間表						
BugID	No.	状態遷移前		状態遷移後		遷移 時間 (日)
		状態	時刻	状態	時刻	
10001	(1)	不具合報告	1月1日06:10	不具合承認	1月1日15:30	0.3
	(2)	不具合承認	1月1日15:30	修正作業終了	1月1日17:00	0.1
	(3)	修正作業終了	1月1日17:00	修正作業確認	1月2日18:30	1.1
		⋮	⋮	⋮	⋮	⋮
24357	(3)	担当者決定	9月7日12:00	修正作業終了	9月9日14:00	2.1
	(4)	修正作業終了	9月9日14:00	修正作業確認	9月9日20:30	0.3

(a) 状態遷移時刻表

(c) 作業時間表		
不具合報告 - 不具合承認		
BugID	遷移時間	作業時間
10001	0.3	12.1
⋮	⋮	
24403	6.8	
修正作業終了-修正作業確認		
BugID	遷移時間	作業時間
10001	1.1	35.5
⋮	⋮	
24357	0.3	
担当者決定 - 修正作業終了		
BugID	遷移時間	作業時間
10004	3.5	0.5
⋮	⋮	
24357	2.1	
⋮		

(c)すべての不具合を対象とし
て遷移元と遷移先が同じ状
態遷移対を抽出し、作業時間
を算出し、作成する。



作業時間表に抽出された各状態と各遷移を状態遷移図として記述する。さらに導出した各作業時間を遷移経路のエッジに記載し、その遷移経路を経由した不具合の数をエッジに記載。

ケーススタディの対象

プロジェクト	Apache	Eclipse	Firefox
種類	Webサーバ	統合ソフトウェア環境	Webブラウザ
不具合件数	4,923	26,113	63,652
期間	2003/1～2008/12	2001/10～2009/11	2003/1～2008/12



Apache 状態遷移図

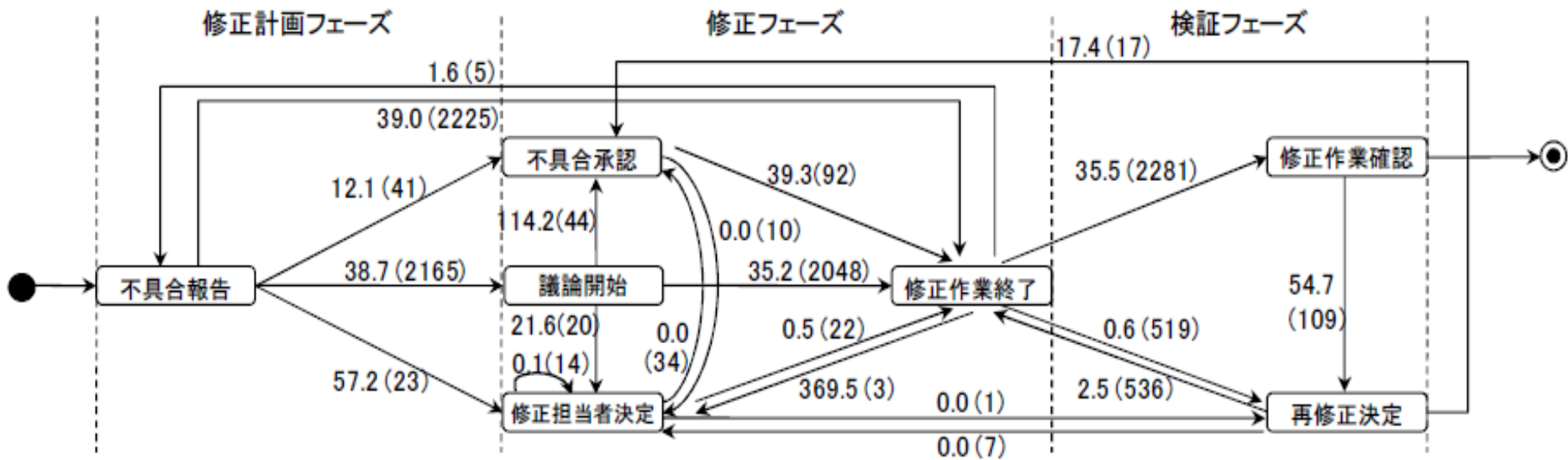


図 4 Apache プロジェクトの不具合修正プロセス

Eclipse Platform 状態遷移図

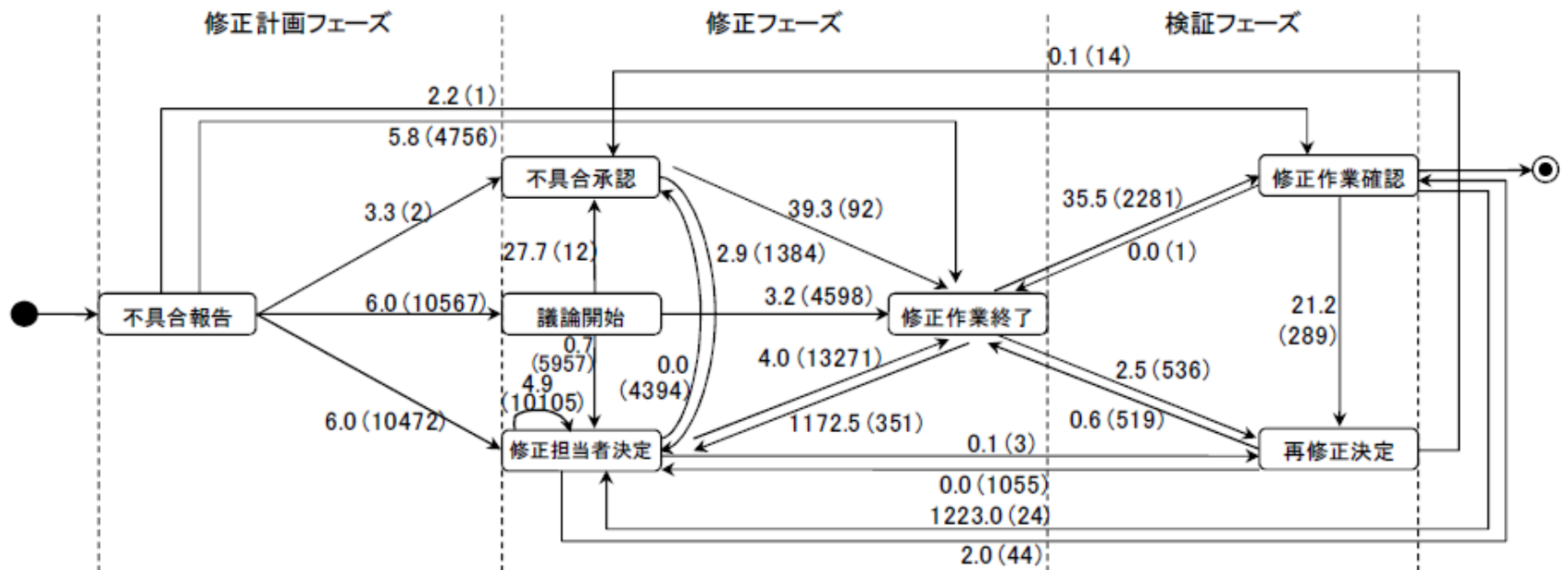


図5 Eclipse Platform プロジェクトの不具合修正プロセス

Firefox 状態遷移図

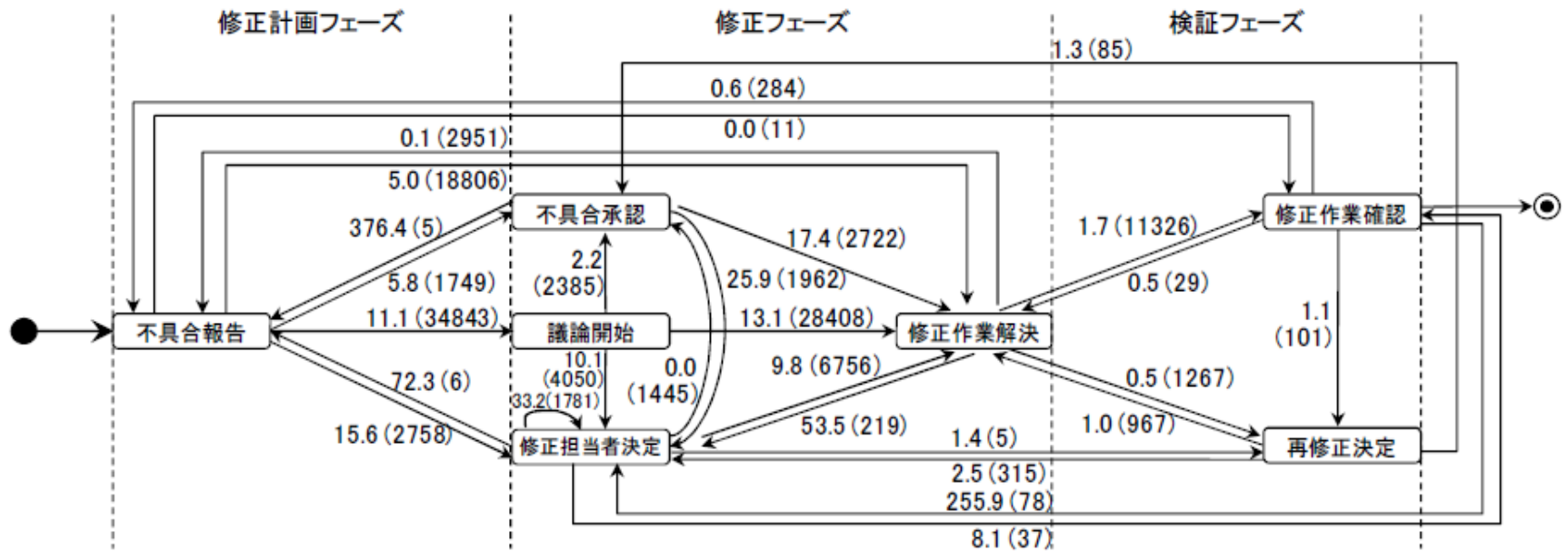


図6 Mozilla Firefox プロジェクトの不具合修正プロセス

ケーススタディの結果

	Apache	Eclipse	Firefox
修正計画フェーズ	(1) 不具合報告⇒修正作業終了 (2) 約40日	(1) 不具合報告⇒議論開始 不具合報告⇒修正担当者決定 (2) 約6日	(1) 不具合報告⇒修正作業終了 (2) 不具合報告⇒議論開始 (3) 約11日
修正フェーズ	(1) 議論開始⇒修正作業終了 (2) 約40日	(1) 修正作業終了⇒修正内容確認 (2) 約4日	(1) 議論開始⇒修正作業終了 (2) 約13日
検証フェーズ	(1) 修正作業終了⇒修正内容確認 (2) 25日	(1) 修正作業終了⇒修正内容確認 (2) 約35日	(1) 修正作業終了⇒修正内容確認 (2) 約2日
改善が必要とされるフェーズ	すべてのフェーズ	検証フェーズ	修正計画フェーズ 修正フェーズ



不具合修正プロセス改善のための方策

▶ 修正計画フェーズ

- ▶ Firefoxなど、多くで不具合報告後➡議論開始に遷移している。
- ▶ これは不具合の詳細を聞くための議論である。
- ▶ Firefoxなどではライトユーザーが多いので、不具合報告の情報が記載されていないことが多い。



不具合修正プロセス改善のための方策

▶ 修正フェーズ

- ▶ 繰り返し修正担当者が変更されるため時間がかかる。
- ▶ 管理者が参加者の専門性を把握できていない。
- ▶ OSSでは、修正担当者の研究は活発に行われているが、まだ実際に適応された事例がほとんどない。



不具合修正プロセス改善のための方策

▶ 検証フェーズ

- ▶ 再修正が必要とされたとき、思い出すのが困難となり、検証フェーズ長期化につながっている。
- ▶ 再修正が必要な不具合を予測する、予測モデルが提案されている。
- ▶ 再修正が必要とされる不具合を優先的に検証できる。



注意

- ▶ ただし不具合管理のルールやポリシーはプロジェクトごとによって異なる。
- ▶ 作業時間は一様に定義することはできない。
- ▶ プロジェクト管理者が状態遷移図を確認し、ボトルネックとなっている作業を特定する必要がある。

