

OSS開発における不具合修正プロセスの現状と課題 不具合修正時間の短縮化へ 向けた分析

伊原彰紀 大平雅雄 松本健一

背景

OSSの普及に伴い、OSSプロジェクトに報告される不具合が急増

報告件数が多く、報告されたすべての不具合を短期間のうちに修正するのは困難

不具合修正の長期化を解消するための方策が望まれている

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の課題を明らかにし、不具合修正プロセスを効率化するための方法を検討する



3つのプロジェクトを対象としたケーススタディを行い、不具合管理システム(BTS)に記録される不具合修正の進捗情報を用いて、過去の不具合修正に必要な一連の作業内容や個々の作業時間の可視化・分析を試みる

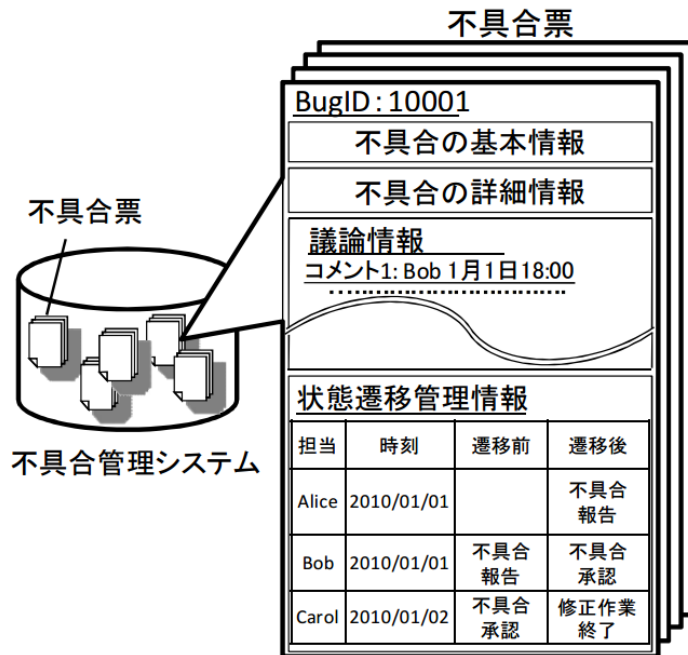
OSSにおける不具合管理

世界中の開発者が不具合情報を共有するため、BTSが利用されている

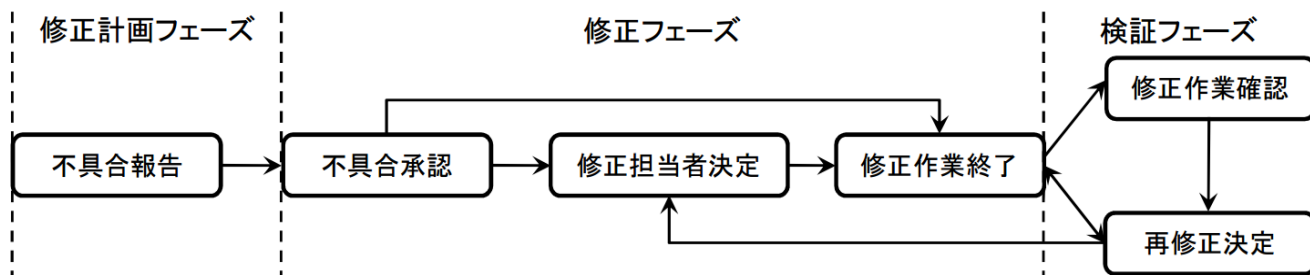
BTSとは

開発者あるいはユーザから報告された1つの不具合に対して1つの不具合表を作成し、個々の不具合修正の進捗を管理するためのシステムである

不具合票の内容



不具合修正プロセス



状態	詳細
不具合報告	開発者又はユーザによって発見された不具合が報告(BTSに登録)された状態。
議論開始	BTSに登録された不具合情報に関して議論が開始された状態。
不具合承認	議論の結果、プロジェクト管理者によって修正すべき不具合として承認された状態。
修正担当者決定	プロジェクト管理者によって修正担当者が決定された状態。
修正作業終了	修正担当者によって修正作業が終了した状態。
再修正決定	不具合の修正が不十分であり、再度修正が要求された状態。
修正作業確認	プロジェクト管理者によって不具合が正しく修正されたことが確認された状態。

分析方法についての概要

状態遷移図で表し、一連の作業内容や個々の作業時間を可視化・分析し不具合修正プロセスを効率化するための指針を検討する

プロジェクト特有の状態遷移や時間を要する作業・フェーズを把握することに重点を置く

これにより、プロジェクト管理者は自身のプロジェクト不具合修正プロセスの問題点を発見できると期待される

分析の手順

① 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	

分析の手順

② 状態遷移時間表の作成

(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

分析の手順

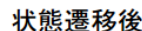
③ 作業時間表の作成

(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	

④ 状態遷移図の構築

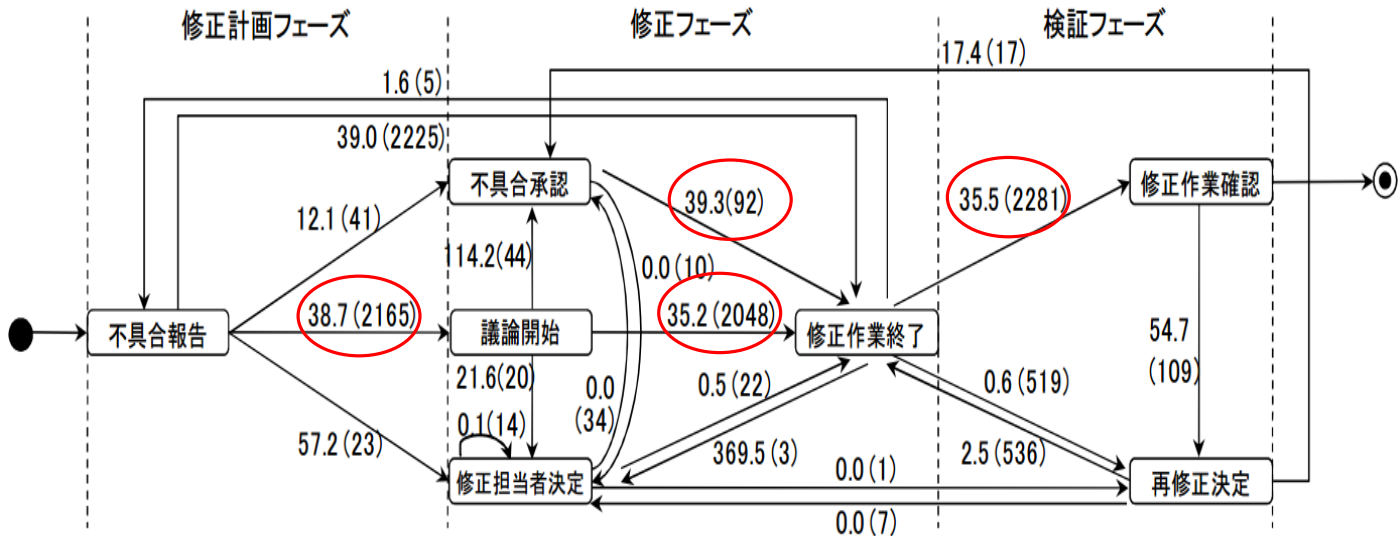
修正フェーズ



ケーススタディ

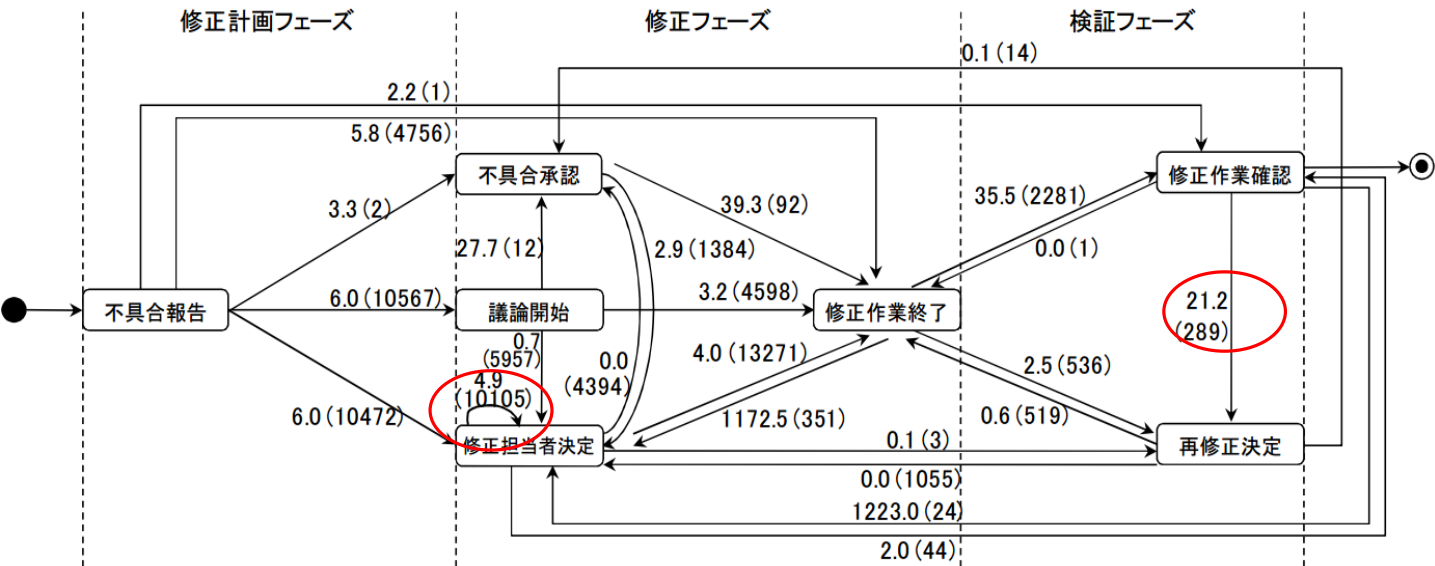
(Apache HTTP Serverプロジェクト)

多くの不具合は各フェーズに同時間(40日)かかっている



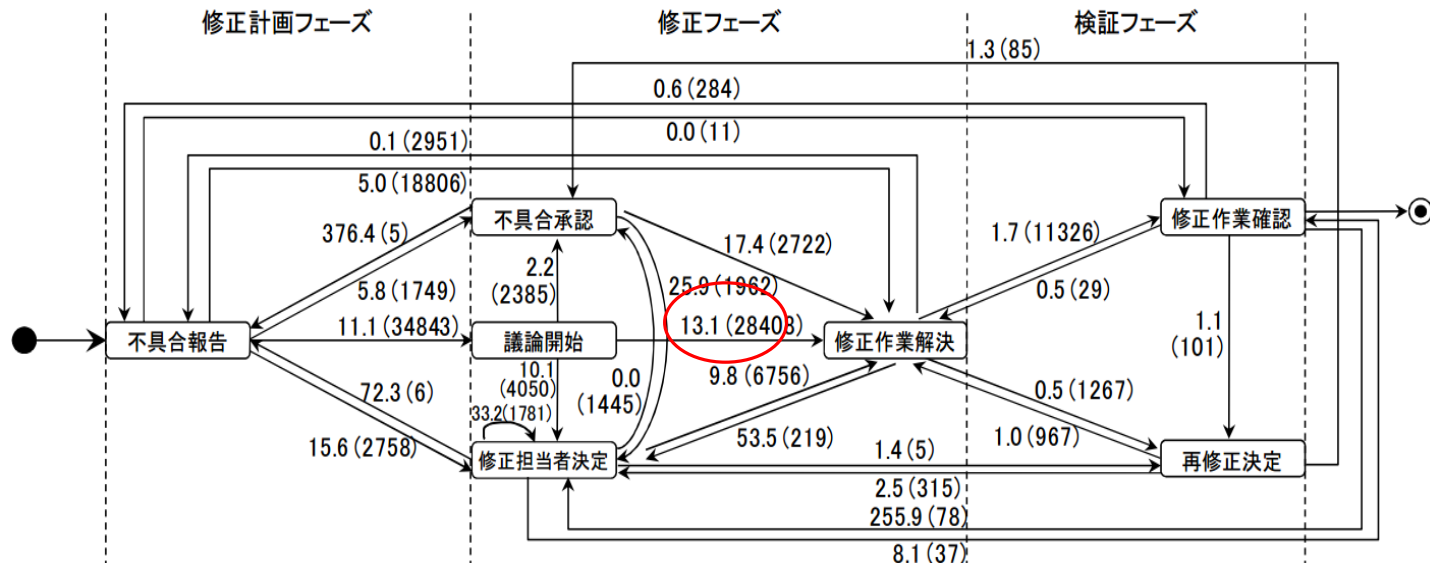
ケーススタディ (Eclipse Platformプロジェクト)

検証フェーズの「修正内容確認」に遷移するまでの時間が最も長い



ケーススタディ (Mozilla Firefoxプロジェクト)

修正計画フェーズや修正フェーズに長い時間を要している



ケーススタディの結果

	Apache HTTP Server	Eclipse Platform	Mozilla Firefox
修正計画フェーズ	(1) 不具合報告→修正作業終了、 不具合報告→議論開始 (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、 不具合報告→議論開始 (2) 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) 修正担当者決定→修正担当者決定、 修正担当者決定→修正作業終了 (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) 約35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
最も改善が必要と 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

考察(修正計画フェーズ)

3 プロジェクトすべてにおいて不具合報告後に議論開始状態に遷移する不具合票が数多く存在することが確認された

ライトユーザの多いプロジェクトでは特に、報告内容として必要な情報に漏れがないように不具合票の報告フォーマットをあらかじめ用意する必要がある

考察(修正フェーズ)

プロジェクト管理者がプロジェクト参加者の専門性を把握できていないため、適切な修正担当者を決定することが困難

不具合修正担当者の自動決定手法の導入と効果の検証が今後の課題である

考察(検証フェーズ)

不具合の再修正によって解決日が先延ばしになってしまうことがある

再修正が必要となる可能性の高いモジュールをあらかじめ見積もっておくことで、プロジェクト管理者は再修正が必要と予測される不具合を優先的に検証し、修正担当候補者へ早急に再修正の依頼を行うことができる

まとめ

3プロジェクト共に、不具合についての議論がなされた後に実際の修正が開始される場合が多く、修正計画フェーズでは時間を要することが分かった

今後はプロジェクト管理者が各作業時間を正確に把握するために、OSSの種類や特徴、不具合の特徴による各作業時間について分析を行うことが課題である