

修士論文

障害管理システム利用時の修正遅延要因の分析

伊原 彰紀

2009年2月5日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士(工学) 授与の要件として提出した修士論文である。

伊原 彰紀

審査委員：

松本 健一 教授	(主指導教員)
関 浩之 教授	(副指導教員)
門田 暁人 准教授	(副指導教員)
大平 雅雄 助教	(副指導教員)

障害管理システム利用時の修正遅延要因の分析*

伊原 彰紀

内容梗概

多くのソフトウェア開発プロジェクトでは、障害を一元管理するために障害管理システムを導入している。しかしながら、ソフトウェア開発の中でも特にオープンソースソフトウェア (OSS) 開発では、障害管理システムを利用しているにも関わらず、障害が報告されてから修正が完了するまでの時間（滞留時間）が極めて長い障害が数多く存在するという問題がある。特に、修正作業そのものにかかる時間だけではなく、修正作業に着手するまでに長い時間を要することが指摘されている。障害の滞留時間短縮支援を行うための第一歩として、修正作業着手までに長い時間を要する障害が発生する要因を明らかにすることが本研究の目的である。本論文では、障害の優先度及び重要度別に修正作業着手までの時間を分析する。また、障害報告数と修正作業着手までの時間との関係を分析する。ケーススタディとして、大規模な OSS を開発する 2 つのプロジェクト (Apache HTTP Server Project 及び Mozilla Firefox Project) の分析を行った。分析の結果、修正作業そのものにかかる時間は短いにも関わらず、修正に取り掛かるまでに多くの時間が費やされている障害が多いことが分かった。さらに、定期的に障害を修正する機会を設けることで、障害の滞留時間を短縮できる可能性があることが分かった。

キーワード

障害管理システム, 障害滞留時間, Apache, Firefox

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 修士論文, NAIST-IS-MT0751012, 2009 年 2 月 5 日.

A Delay Factor of Defect Correction on A Bug Tracking System*

Akinori Ihara

Abstract

Many software development projects use a bug tracking system (BTS) to uniform the management of bugs. Regardless of the use of BTS, especially in open source software (OSS) development, projects often face the problem of the delay of bug fix. The delay is caused by not only bug fix operations but also the time until the start of the operations. Toward providing a means for resolving the problem, this thesis tries to identify the factors which have an effect on delay of the start of the bug fix operations. I analyze the time until the start of the bug fix operations by the priority and severity of reported bugs, and analyze the relationship between the number of bug reports and the time until the start of the bug fix operations. The thesis shows a case study using two projects (Apache HTTP Server project and Mozilla Firefox project), which has been developing a large scale OSS. The analysis results indicates that many bugs spend long time until the bug fix operations while the time required for the operations itself is short. The analysis results also suggest that fixing bugs periodically would be able to improve the problem of the delay of bug fix.

Keywords:

bug tracking system, time for bug fix, Apache, Firefox

*Master's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT0751012, February 5, 2009.

関連発表論文

研究会・シンポジウム

1. 伊原 彰紀, 前島 弘敬, 枡本 真佑, 亀井 靖高, 大平 雅雄, 松本 健一. 複数のサブコミュニティを有する OSS コミュニティにおけるコーディネータの分析, 情報処理学会シンポジウム グループウェアとネットワークサービス・ワークショップ 2007 論文集, Vol.2007, No.11, pp.13-18, November 2007.
2. 伊原 彰紀, 大平 雅雄, 枡本 真佑, 亀井 靖高, 松本 健一. 複数のサブコミュニティを有する OSS コミュニティを対象としたネットワーク分析, マルチメディア, 分散, 協調とモバイル (DICOMO2008) シンポジウム, pp.297-303, July 2008.
3. 伊原 彰紀, 亀井 靖高, 大平 雅雄, 枡本 真佑, 松本 健一. OSS プロジェクトにおける障害に関する情報共有の分析, 平成 20 年度 情報処理学会関西支部支部大会 講演論文集, Vol.2008, pp.69-72, October 2008.

目 次

1. 序論	1
1.1 背景	1
1.2 目的とアプローチ	2
1.3 本論文の構成	3
2. 準備	4
2.1 障害管理システム	4
2.2 障害修正のためのコメント	6
2.3 滞留時間, 未対応時間, 対応時間の定義	7
3. 分析方法	9
3.1 未対応時間と滞留時間	9
3.2 重要度, 優先度による未対応時間	10
3.3 障害報告数と未対応障害数	11
4. ケーススタディ1: Apache HTTP Server Project	13
4.1 Apache プロジェクトの歴史と障害管理形態	13
4.2 分析結果	14
4.2.1 基本統計量	14
4.2.2 未対応時間と滞留時間	15
4.2.3 重要度, 優先度による未対応時間	17
4.2.4 障害報告数と未対応障害数	20
5. ケーススタディ2: Mozilla Firefox Project	21
5.1 Firefox プロジェクトの歴史と障害管理形態	21
5.2 分析結果	22
5.2.1 基本統計量	22
5.2.2 未対応時間と滞留時間	23
5.2.3 重要度, 優先度による未対応時間	25

5.2.4	障害報告数と未対応障害数	28
6.	考察	30
6.1	未対応時間と滞留時間	30
6.2	重要度, 優先度による未対応時間	33
6.3	障害報告数と未対応障害数	34
6.4	本論文の制約	34
7.	関連研究	35
7.1	障害修正の問題点	35
7.2	OSS 開発における障害の滞留時間	35
8.	結論	37
	謝辞	38
	参考文献	40
	付録	42
A.	障害報告数と未対応時間が 1 週間以上の障害の推移	42
A.1	Apache	42
A.2	Firefox	43

図 目 次

1	Bugzilla における障害の修正プロセス [6]	5
2	滞留時間, 未対応時間, 対応時間の概略図	7
3	未対応時間と対応時間の関係	9
4	重要度, 優先度別の対応された障害の割合を示す概略図	11
5	Apache における障害修正プロセス	14
6	滞留時間別の障害数の分布 (Apache)	15
7	未対応時間別の障害数の分布 (Apache)	16
8	障害の未対応時間と対応時間の関係 (Apache)	16
9	重要度別の対応された障害数の累積グラフ (Apache)	18
10	優先度別の対応された障害数の累積グラフ (Apache)	18
11	障害報告数と未対応時間が 1 か月以上の障害の推移 (Apache)	20
12	Mozilla における障害修正プロセス	22
13	滞留時間別の障害数の分布 (Firefox)	23
14	未対応時間別の障害数の分布 (Firefox)	24
15	障害の未対応時間と対応時間の関係 (Firefox)	24
16	重要度別の対応された障害数の累積グラフ (Firefox)	26
17	優先度別の対応された障害数の累積グラフ (Firefox)	26
18	障害報告数と未対応時間が 1 か月以上の障害の推移 (Firefox)	28
19	コメント数別の障害数の分布 (Apache)	32
20	コメント数別の障害数の分布 (Firefox)	32
21	障害報告数と未対応時間が 1 週間以上の障害の推移 (Apache)	42
22	障害報告数と未対応時間が 1 週間以上の障害の推移 (Firefox)	43

表 目 次

1	Bugzilla における STATUS の種類	6
2	Bugzilla における RESOLUTION の種類	6
3	Bugzilla の各重要度	10

4	Apache の分析対象データの概要	14
5	Apache における障害の滞留時間	14
6	重要度別の障害数 (Apache)	17
7	優先度別の障害数 (Apache)	19
8	Firefox の分析対象データの概要	22
9	Firefox における障害の滞留時間	22
10	重要度別の障害数 (Firefox)	25
11	優先度別の障害数 (Firefox)	27
12	コメント数の統計量	33

1. 序論

1.1 背景

近年、ソフトウェアの大規模化や複雑化に伴い、障害の数が増加するという問題が発生している。障害が増加すると開発者が対応しきれないため障害が放置されたり、全ての障害修正の状況を把握することが困難になる。そこで、多くのソフトウェア開発プロジェクトでは障害を一元管理するために障害管理システムを利用している。障害管理システムでは障害報告時に重要度や優先度等を指定し、早急に直すべき障害を容易に検索可能にしている。また、多くの障害管理システムでは報告された障害ごとに掲示板を設ける機能があり、修正状況や修正方針に関する議論(情報共有)が行われている。

しかし、ソフトウェア開発の中でも特にオープンソースソフトウェア(OSS)開発では障害管理システムを利用しているにも関わらず、障害の修正が効率的に行われていないことが多い[13][10]。WangらはLinuxディストリビューションの一つであるUbuntuを対象とし、障害の修正状況を分析している。その結果、報告されている障害のうち、約8割の障害は修正が完了しておらず、その中でも多くの障害はアサイン(障害を修正する担当者を割り当てること)さえ行われていないため、修正に取り掛かるまでに長い時間がかかっていると報告している[13]。障害修正に向けた対応が開始されないと、障害を報告してから修正が完了するまでの時間(本論文ではこの時間のことを障害の滞留時間と呼ぶ)が長くなり、OSS開発に様々な影響を及ぼすと考えられる。例えば、頻繁に利用されるモジュールに存在する障害は他のモジュールにも危害を与えやすく、滞留時間の長期化によりシステムに対して深刻な影響を与える可能性がある。また、障害の滞留時間が長くなると修正コストが高くなる可能性もある[14]。障害修正の遅延はOSSに限らず、企業におけるソフトウェア開発でも起こり得ることである。

このようなことから、障害の滞留時間の短縮化が求められている。本論文では、障害の滞留時間を、障害が報告されてから修正が開始されるまでの時間(本論文ではこの時間のことを未対応時間と呼ぶ)と障害の修正が開始されてから修正が完了するまでの時間(本論文ではこの時間のことを障害の対応時間と呼ぶ)の2つ

に分けて考える。このうち、対応時間は開発者の技術力に依存するため容易に短縮することは困難だが、未対応時間は報告されてから早い時期に障害修正に取り掛かることで容易に短縮化を実現できると考えられる。

1.2 目的とアプローチ

本論文では、障害の滞留時間を短縮するための第一歩として、未対応時間の長い障害はどのような障害なのか明らかにすることを目的とする。そのため、本論文では未対応時間と滞留時間の分析、及び重要度別・優先度別の未対応時間の分析、及び障害の報告数による未対応時間の分析の3つの分析を行う。

まず、未対応時間と滞留時間の分析を行う。未対応時間が長くなれば滞留時間が長くなるのは当然であるが、対応時間が長くなると滞留時間はさらに長くなる。そこで、OSS 開発における障害の滞留時間、未対応時間の現状を把握し、未対応時間が長い障害は対応時間にどれくらいの時間がかけているのか明らかにするために、未対応時間と滞留時間の関係を分析する。

次に重要度別・優先度別の未対応時間の分析を行う。未対応時間を最も左右させるのは、重要度や優先度であると考えられる。重要度や優先度は、それぞれ障害の深刻度、優先順位を意味し、開発者はこれらの情報を用いて優先的に修正すべき障害を選別している。これまで、重要度や優先度による滞留時間の違いは明らかにされているが、重要度や優先度による未対応時間の違いは明らかにされていない。本論文では、重要度や優先度を指定することで障害修正を開始する早さが異なるかを知るために、重要度および優先度別の未対応時間を分析する。

そして、最後に障害の報告数による未対応時間の分析を行う。障害報告数は常に変動し、修正を行う開発者も常に変動する。もし障害報告数が多く、修正を行う開発者が少なければ、全ての障害に対応しきれなくなり、修正に取り掛かるのが遅れてしまうことが考えられる。そこで、障害報告数の増減により未対応時間がどのように変化しているかを明らかにするために、障害報告数と未対応時間が長い障害の数の関係を分析する。

1.3 本論文の構成

本論文の構成は以下の通りである．続く2章では障害管理システムと障害に関するコメントについて説明した後，本論文で扱う障害の滞留時間についての定義を行う．そして，3章ではケーススタディで行う分析方法について説明する．4章では Apache HTTP Server Project(以降，Apache)を対象としたケーススタディを報告し，5章では Mozilla Firefox Project(以降，Firefox)を対象としたケーススタディを報告する．6章では Apache と Firefox に関する議論を行い，7章では関連研究を述べ本研究の立場を明らかにする．最後に，8章では結論を述べる．

2. 準備

本章では障害管理システムを利用している OSS をケーススタディとして分析するための事前知識の説明と、本論文で扱う用語の定義を述べる。2.1 節では障害管理システムの概要を説明する。次に、2.2 節では障害修正のために開発者はどのような内容のコメントを投稿しているのかを述べる。そして、2.3 節で本論文で扱う障害の滞留時間、未対応時間、対応時間の定義を述べる。

2.1 障害管理システム

ソフトウェア開発において障害の修正や管理は重要な作業である。しかし、ソフトウェアの規模が大規模化するに伴い、障害の数が増加し、障害の管理が困難になる。特に OSS 開発では、一般的に開発者が地理的に分散している開発環境であることが多く [8]、誰が修正を行い、現在どのような修正状況であるのか把握することが困難である。そこで、多くの OSS では障害の一元管理及び開発者間の情報共有をサポートする障害管理システムを利用している [4]。

障害管理システムでは障害を報告する際にプロダクトやバージョン番号、障害の解決方法、重要度、優先度などの属性を指定することが可能である。また、障害が報告されてから修正が終わるまでの修正履歴を管理することができる。障害管理システムを用いることで、開発者は修正状態や、重要度などの属性から優先的に修正すべき障害の選別、個々の障害の修正状況などを容易に把握することが可能である [9]。

障害管理システムには Bugzilla[2]、Trac[12]、Mantis[7] など様々な種類があるが、本論文では機能の豊富さなどから多くの OSS が利用している Bugzilla[3] を対象として分析を行う。Bugzilla における修正プロセスを図 1 に示す [6]。また、図 1 における各状態の内容を表 1 に、STATUS が障害の解決を意味する RESOLVED である時に、どのような意味で解決されているか（例えば、障害が修正された、そもそも障害でない指摘であったなど）を示す RESOLUTION の内容を表 2 に示す。OSS により修正プロセスの STATUS が多少異なる場合があるが、Bugzilla を利用する OSS の障害の多くはこの修正プロセスに従って修正される。

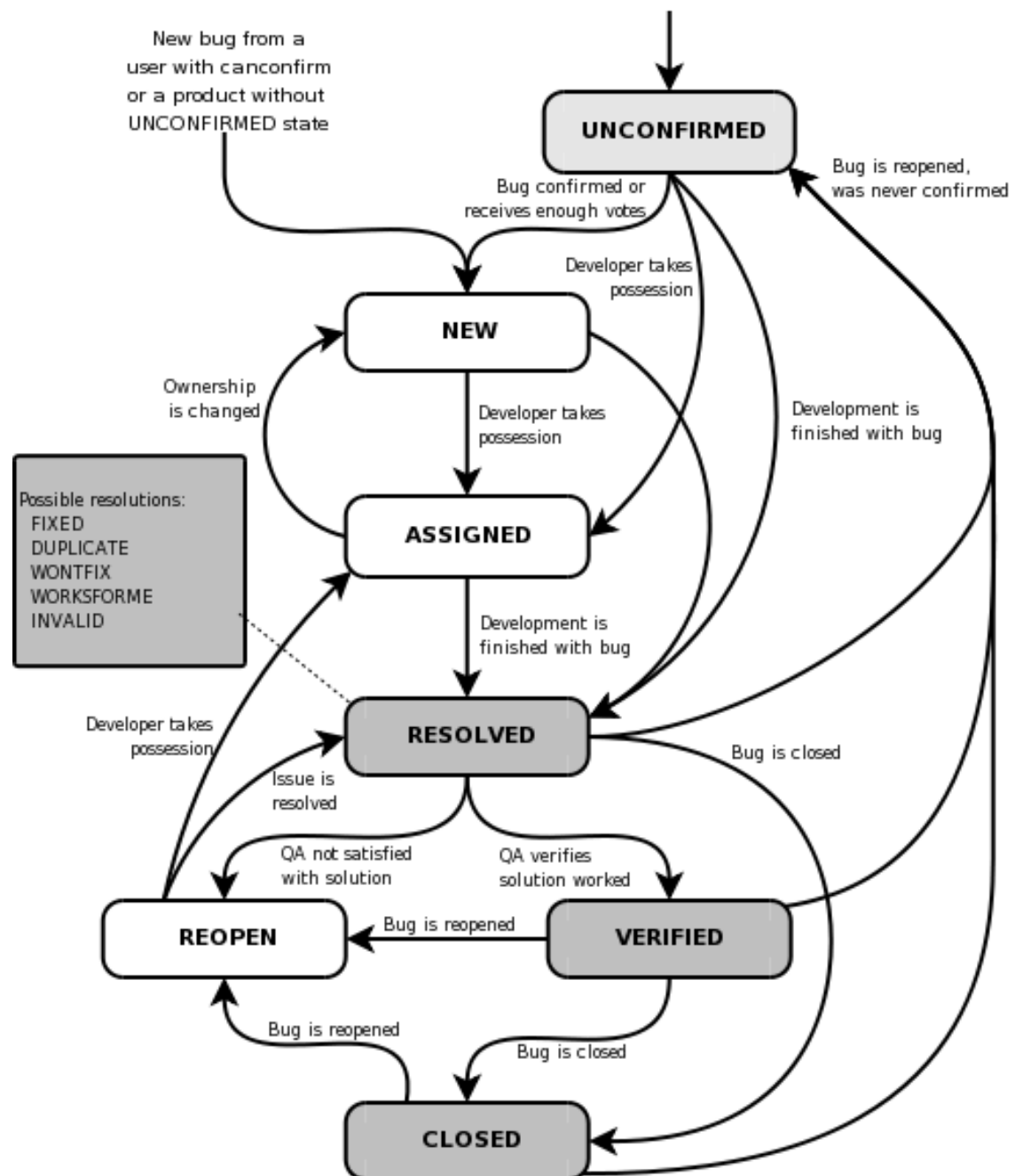


図 1 Bugzilla における障害の修正プロセス [6]

表 1 Bugzilla における STATUS の種類

STATUS の種類	内容
UNCONFIRMED	障害に対する確認作業が行われていない状態
NEW	修正が必要な障害であると確認された状態
ASSIGNED	まだ解決されていないが、修正できると考えられる開発者に割り当てられた状態
REOPENED	一度は解決されたが、その解決は誤っており、再度修正が必要とされた状態
RESOLVED	解決されたが、第三者による検証を待っている状態
VERIFIED	第三者による検証が行われた状態
CLOSED	第三者による検証が正しく行われ、障害は修正されたと判断された状態

表 2 Bugzilla における RESOLUTION の種類

RESOLUTION の種類	内容
FIXED	障害の修正が確認された状態
INVALID	障害ではないと判断された状態
WONTFIX	将来、修正されることはない判断された状態
DUPLICATE	過去に報告されている障害と判断された状態
WORKSFORME	障害を再現できず、コードを解読しても障害が起こった原因が掴めないと判断された状態

2.2 障害修正のためのコメント

Bugzilla では各障害に対して議論 (情報共有) を行うために掲示板を設ける機能がある。掲示板にコメントされる内容は障害によって様々である。以下に、掲示板に投稿されるコメントの代表例を示す。

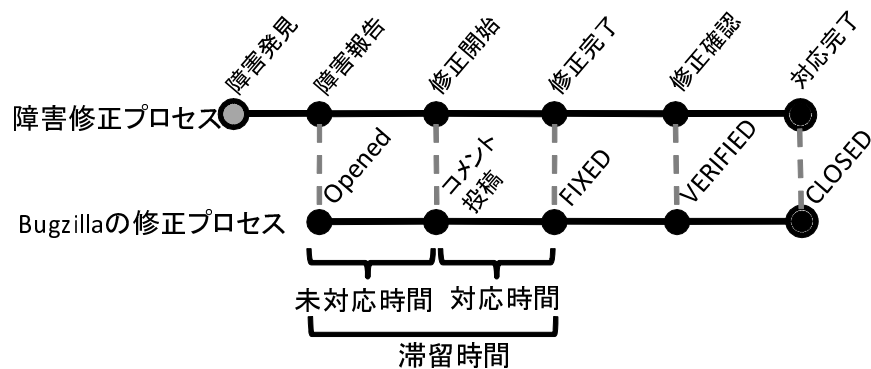


図 2 滞留時間，未対応時間，対応時間の概略図

- 障害報告者への質問
- 障害修正 (パッチの作成) を行った報告，検証の要求
- 修正に関するアドバイス
- 類似障害の紹介

ここに挙げた内容に限らず，掲示板は障害に関する情報共有の場として利用される．掲示板は障害の進捗状況の把握や修正活動のために利用されており，掲示板に記載されているコメントを障害の修正履歴の一つとみなすことができる．よって，本論文では掲示板に1回目のコメントが投稿された時間を障害修正に向けた修正活動の開始時間と考える．ただし，掲示板にコメントが無い障害については，修正に向けた情報共有の開始時間を取得することができず，修正履歴をたどることができないため，本論文では分析対象外とする．

2.3 滞留時間，未対応時間，対応時間の定義

図2に本論文で扱う滞留時間，未対応時間，対応時間の概略図を示す．本論文では，障害が報告されてから修正が完了するまでの時間を障害の滞留時間と呼ぶ．修正が完了する時間は3つのケースが考えられる．一つは，完全に障害が取り除かれた状態 (STATUSがCLOSEDに変更される時間)，次に，障害の修正が確認さ

れた状態 (STATUSが VERIFIEDに変更された時間), 最後に, 障害が修正された状態 (STATUSが CLOSEDに変更された直前の STATUSが RESOLVEDであり, かつ, RESOLUTIONが FIXEDとされた時間) である. CLOSEDや VERIFIEDは障害が修正され, 検証される状態を意味することから, 修正は FIXEDの際に既に完了している. よって, 本論文では STATUSが CLOSEDや VERIFIEDに変更される直前の STATUSが RESOLVEDであり, かつ RESOLUTIONが FIXEDとされた時間を障害修正の完了時間とする. 次に, 障害の未対応時間と対応時間の定義をそれぞれ述べる.

- 未対応時間

本論文では図 2 より, 障害が報告されてから修正開始まで (Bugzilla では Opened から 1 回目のコメントが投稿されるまで) の時間を障害の未対応時間とする.

- 対応時間

本論文では図 2 より, 修正開始から修正完了まで (Bugzilla では 1 回目のコメントが投稿されてから STATUSが CLOSEDや VERIFIEDに変更される直前の STATUSが RESOLVEDであり, かつ RESOLUTIONが FIXEDとされた時間まで) を障害の対応時間とする.

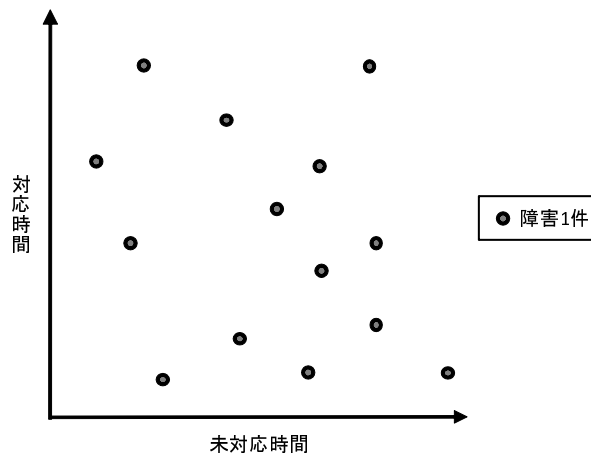


図 3 未対応時間と対応時間の関係

3. 分析方法

本章では障害の滞留時間を短縮するために、未対応時間が長い障害がどのような障害なのかを明らかにするための3つの分析方法について述べる．3.1 節では障害の未対応時間と滞留時間の関係の分析方法について述べる．次に、3.2 節では重要度、優先度による未対応時間についての分析方法を述べ、3.3 節では障害報告数と未対応障害数の関係についての分析方法について述べる．

3.1 未対応時間と滞留時間

Wang らは未対応時間が長い障害が多いと指摘している [13]．そこで、本論文では障害の滞留時間における未対応時間の現状を把握するために、障害の未対応時間と対応時間の関係を分析する．滞留時間は未対応時間と対応時間の和である．滞留時間が長い障害は未対応時間と対応時間の両方が長いのか、もしくは未対応時間が長いというだけで滞留時間が長くなっているのかを明らかにするために分析する．図3のように障害の未対応時間と対応時間の関係を散布図で示す．各プロットは障害1件を示す．この散布図から未対応時間による対応時間の分布を明らかにすることができる．

表 3 Bugzilla の各重要度

重要度	内容
Blocker	開発とテストの少なくとも一方を妨げる障害
Critical	データの損失や深刻なメモリリークを引き起こす障害
Major	主要な機能が欠如している障害
Normal	他の重要度に該当しない一般的な障害
Minor	あまり重要でない機能の欠如, または簡単な代替手段がある障害
Trivial	単語の打ち間違い等の表面上の障害
Enhancement	機能拡張のリクエスト

3.2 重要度，優先度による未対応時間

障害管理システムには開発者が早く直すべき障害を容易に検索可能にするために，システムに対してどの程度深刻な影響を与えるかを示す重要度，開発者がどの程度優先的に修正すべきかを示す優先度を障害ごとに指定する機能がある．Bugzillaにおける重要度は7種類(Blocker, Critical, Major, Normal, Minor, Trivial, Enhancement)に分類されている．各重要度の意味を表3に示す．また，優先度はP1(最も高い優先度)からP5(最も低い優先度)までの5段階に分類されている．

重要度と優先度はそれぞれ障害の深刻度，及び修正するための優先順位をつけるための情報であり，修正に取り掛かるまでの未対応時間を左右すると考え，本論文では重要度が高い障害/低い障害のどちらが早く障害修正に向けた対応が行われるのかを明らかにする．また，優先度についても同様に，優先度が高い障害/低い障害のどちらが早く対応されるのかを明らかにする．

重要度別，優先度別の対応時間は図4のように示す．横軸は未対応時間，縦軸は対応された障害の割合とする．曲線Aは重要度(または優先度)が高い(もしくは低い)障害の累積であり，一方，曲線Bは重要度(または優先度)が低い(もしくは高い)障害の累積である．図4の場合，曲線Aは報告された障害のうち4日

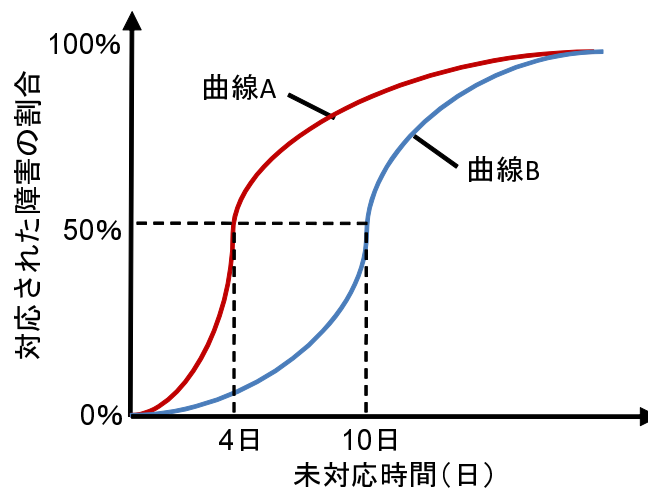


図 4 重要度、優先度別の対応された障害の割合を示す概略図

以内に対応された障害が50%存在することを意味し、曲線Bは報告された障害のうち10日以内に対応された障害が50%存在することを意味する。つまり、曲線Bよりも曲線Aの方が障害への対応が早いことを示す。

また、対応が遅い障害の中で、重要度の高い障害や優先度の高い障害が存在すれば、コメントの内容からなぜ対応が遅くなったのかを検討する。本論文では、未対応時間が1か月以上の障害を対応の遅い障害として扱う。

また、重要度がBlocker, Critical, Majorである障害を重要度の高い障害, Normal, Minor, Trivialである障害を重要度の低い障害とし、優先度については、P1, P2である障害を優先度の高い障害, P3, P4, P5である障害を優先度の低い障害とする。ただし、OSSの機能拡張を示すEnhancementは障害に関する報告とは異なるため分析対象外とする。

3.3 障害報告数と未対応障害数

リリース前に十分にテストを行っておくことが重要であるが、大規模なソフトウェアほどテストを網羅することは難しく、リリース直後は障害報告が増加し、開発者が対応しきれないために、未対応の障害が増加すると考えられる。

対応が遅れる障害はどのような時期に報告された障害なのかを明らかにするために、対応されていない障害の数 (未対応障害数) はリリース時期と関係あるのか、そして、報告数の増減により対応される障害の数は変化するのかを分析する。

本論文では対応の遅い障害を未対応時間が 1 か月以上の障害として分析を行う。参考として、障害の未対応時間が 1 週間以上の場合についての結果を付録に載せる。

4. ケーススタディ1: Apache HTTP Server Project

本章では3章で述べた分析方法を用いて Apache プロジェクト [1] を対象としたケーススタディを行う。4.1 節では分析対象である Apache プロジェクト設立の歴史と Apache における障害管理方法について述べる。そして、4.2 節で分析結果について報告する。

4.1 Apache プロジェクトの歴史と障害管理形態

Apache プロジェクトは、1995 年に NCSA (National Center for Supercomputing Applications) で Robert McCool らにより開発された Web サーバ (NCSA HTTPd) にパッチを提供していた開発者らが発足したプロジェクトである。開発者らはメールなどを通じて NCSA HTTPd の機能拡張や障害修正を独自に続け、1996 年に Apache httpd 1.0 をリリースした。現在は、ver.1.3 系列、ver.2.0 系列そして ver.2.2 系列の開発が並行して進められている。本論文では ver.1.3 系列、ver.2.0 系列、ver.2.2 系列を分析対象とする。

Apache は障害管理システムである Bugzilla を 2003 年 3 月頃から導入している。Apache では障害を発見した際、Bugzilla に投稿する前に、まずユーザ用のメーリングリストに投稿し、数日間返信がなかった場合、Bugzilla に投稿するという報告プロセスが定められている。

Apache プロジェクトが利用する Bugzilla の修正プロセスを図 5 に示す。楕円は STATUS を示し、四角は RESOLUTION を示す。図 1 に示す一般的な Bugzilla の修正プロセスに加え、STATUS に「NEEDINFO」という状態がある。これは、報告者からの情報では修正するための情報として不足している時に「NEEDINFO」の状態に遷移する。

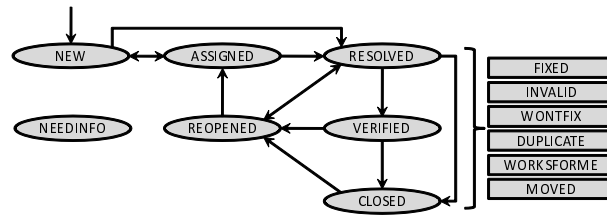


図 5 Apache における障害修正プロセス

表 4 Apache の分析対象データの概要

分析対象障害数	747 件
データ取得開始日	2002 年 3 月 17 日
データ取得終了日	2008 年 11 月 30 日

表 5 Apache における障害の滞留時間

	滞留時間	未対応時間	対応時間
平均 (日)	97.59	27.57	70.02
中央値 (日)	24.13	0.81	4.63
標準偏差	186.05	73.03	167.07
分散	34567.72	5325.78	27873.44
最大値 (日)	2153.45	751.22	1908.70
最小値 (日)	0.00	0.00	0.00

4.2 分析結果

4.2.1 基本統計量

Apache における障害の分析対象データの概要を表 4 に示す。修正が完了した障害 (STATUS が CLOSED, もしくは VERIFIED である障害) は 749 件存在した。749 件のうち、コメントが 1 回も投稿されなかった障害が 2 件存在したため、本論文では 747 件の障害を分析対象とする。

また、障害の滞留時間、未対応時間、対応時間に関するそれぞれの統計量を表

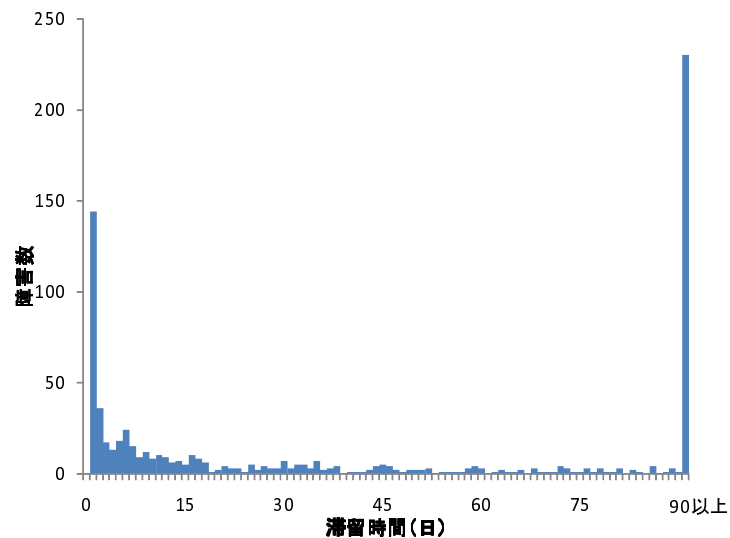


図 6 滞留時間別の障害数の分布 (Apache)

5に示す。Apacheにおける滞留時間別の障害数の分布と未対応別の障害数の分布をそれぞれ図6と図7に示す。図6の横軸は障害の滞留時間であり、図7の横軸は障害の未対応時間である。ヒストグラムのデータ区間は共に1日である。また、滞留時間（または未対応時間）が非常に長い（90日以上）の障害については一つの階級にまとめている。

未対応時間の中央値が0.81日であることから、障害の半数は1日以内に対応がなされていることが分かる。しかし、3か月以上対応がなされていない障害は77件存在し、中には2年以上対応されていない障害が存在していることが分かった。

4.2.2 未対応時間と滞留時間

障害の未対応時間と対応時間の関係を図8に散布図で示す。横軸を未対応時間、縦軸を対応時間とする。未対応時間、対応時間ともに900日以内の分布を示す。

図8より、未対応時間が1日以内かつ対応時間に長い時間がかかっている障害と、対応時間が1日以内かつ未対応時間に長い時間がかかっている障害に偏っていることが分かった。また、対応時間が1日以内かつ未対応時間が長い障害は、実際には短い時間で修正が可能であるにも関わらず長い時間放置されていた障害

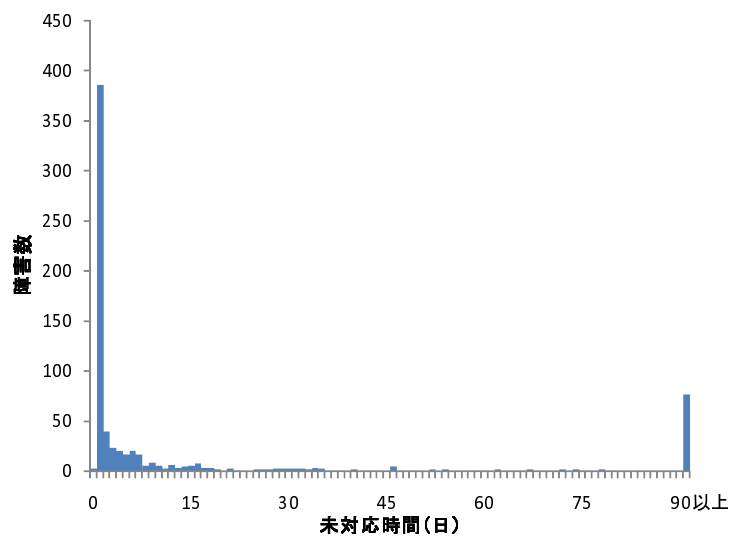


図 7 未対応時間別の障害数の分布 (Apache)

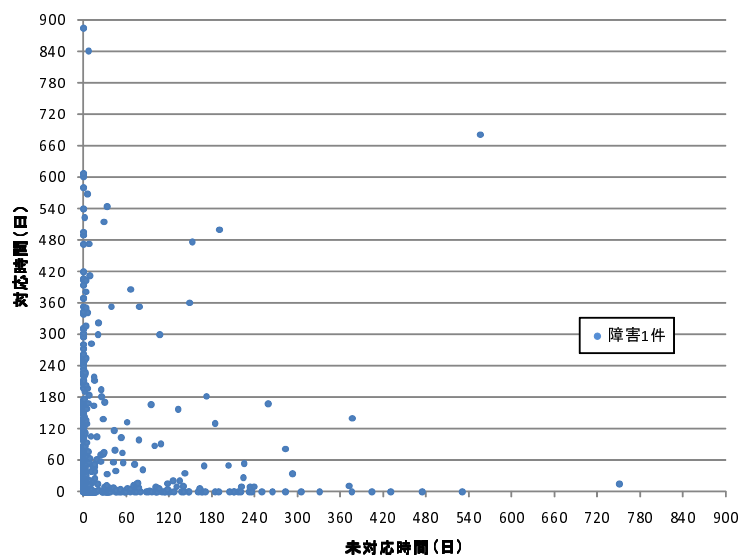


図 8 障害の未対応時間と対応時間の関係 (Apache)

表 6 重要度別の障害数 (Apache)

重要度		障害数 (割合)	
高い	blocker	36(4.8%)	202(27.0%)
	critical	60(8.0%)	
	major	106(14.2%)	
低い	normal	409(54.8%)	545(73.0%)
	minor	132(17.7%)	
	normal	4(0.5%)	
全体		747(100%)	747(100%)

のことであり、滞留時間短縮のための一つの改善点であるといえる。

4.2.3 重要度，優先度による未対応時間

- 重要度

Apache の障害数を重要度別に表 6 に示す。また、重要度別の対応された障害数の累積グラフを図 9 に示す。横軸を未対応時間、縦軸を対応された障害の割合とする。ただし、半数以上の障害の未対応時間は 1 日以内である一方で、最大値は 751 日と偏った分布をしていることから、横軸を対数軸で表す。分析の結果、重要度が高い障害のうち約 55%、重要度の低い障害のうち約 50%が報告されてから 1 日以内に対応されていることが分かった。

- 優先度

Apache の障害数を優先度別に表 7 に示す。また、優先度別の対応された障害数の累積グラフを図 10 に示す。横軸を未対応時間、縦軸を対応された障害の割合とする。ただし、重要度の分析と同様に、半数以上の障害の未対応時間は 1 日以内である一方で、最大値は 751 日と偏った分布をしていることから、横軸を対数軸で表す。分析の結果、優先度が高い障害のうち約 63%、重要度の低い障害のうち約 51%が報告されてから 1 日以内に対応

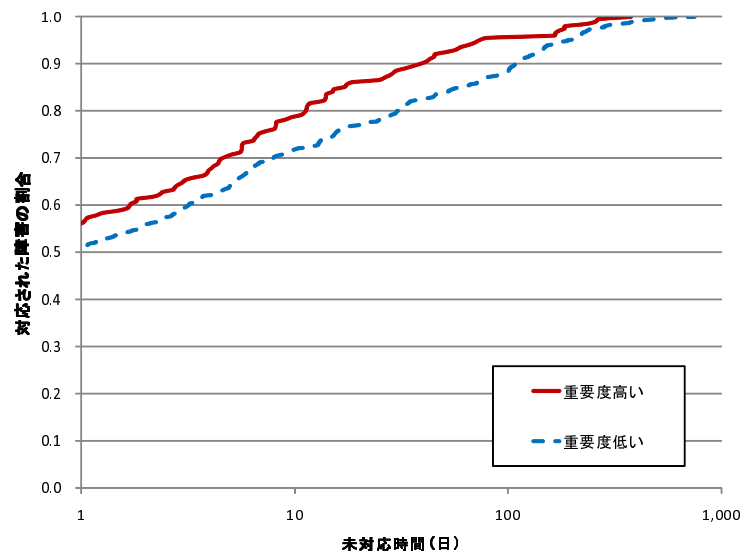


図 9 重要度別の対応された障害数の累積グラフ (Apache)

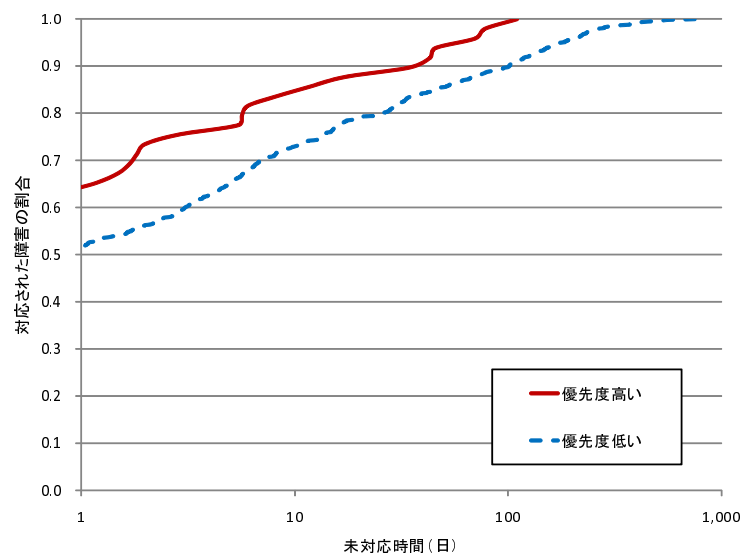


図 10 優先度別の対応された障害数の累積グラフ (Apache)

表 7 優先度別の障害数 (Apache)

優先度		障害数 (割合)	
高い	P1	24(3.2%)	49(6.6%)
	P2	25(3.3%)	
低い	P3	690(92.4%)	
	P4	1(0.1%)	698(93.4%)
	P5	7(0.9%)	
全体		747(100%)	747(100%)

されていることが分かった。

分析対象データの中で重要度、優先度の高い障害は対応が早いことが分かった。しかし、重要度、優先度が高いにも関わらず、長期にわたって対応がなされていない障害も存在した。具体的には、未対応時間が1か月以上で重要度が高い障害が23件(障害全体のうち16.1%)、優先度が高い障害が49件(障害全体のうち6.6%)存在していた。これらの障害の未対応時間がなぜ長くなったのかを明らかにするために障害に対するコメントの内容を分析した。その結果、多かったコメントをいくつか挙げる。

- どのような対応を行うかを指摘するアドバイス
- 以前に似ている障害、もしくは関連する障害を見たことがあるという指摘
- 修正した、またはパッチを作ったという報告

重要度の高い障害の中には、報告者は自分以外の誰かがパッチを作ると期待したが、誰も作らないため自分で修正パッチを作ったケースや、報告の際に修正案を出したが、誰も対応せず、2ヶ月後に他の開発者からの反応があるケースが存在した。中には1年以上経って、報告者にアドバイスを求めるケースもあった。

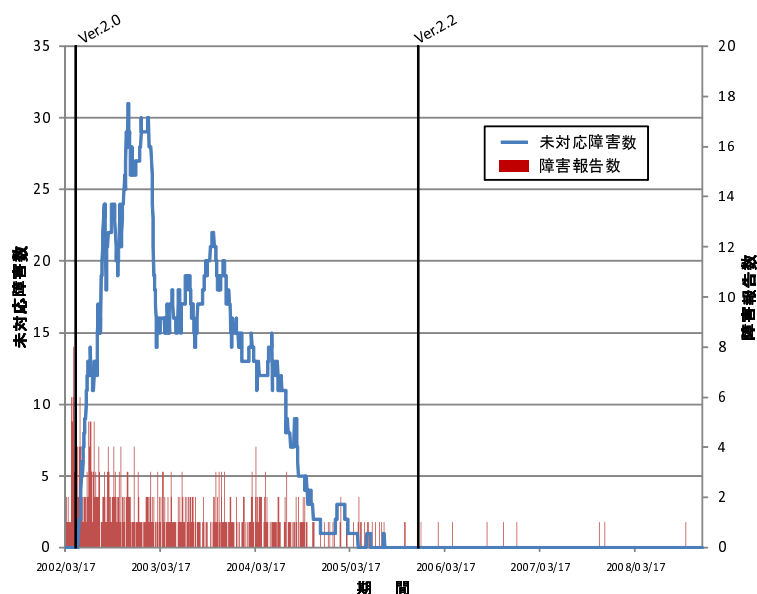


図 11 障害報告数と未対応時間が 1 か月以上の障害の推移 (Apache)

4.2.4 障害報告数と未対応障害数

未対応時間が 1 か月以上の障害数 (未対応障害数) の推移を図 11 に示す。未対応障害数を折れ線グラフ、障害報告数を棒グラフで示す。横軸を期間とし、左側の縦軸を未対応障害数、右側の縦軸を障害報告数とする。障害の未対応時間が 1 週間障害以上であるグラフは付録の図 21 に示す。

2002 年 4 月の ver.2.0 がリリースされた頃から 2004 年中旬頃まで 1 日に平均 1 件の障害が報告されている。また、累積未対応障害数は 2002 年 11 月のピーク時に約 30 件となり、それ以降は減少し、2005 年頃には数件程度にまで減少している。

よって、2002 年 11 月頃までは修正される障害数よりも、報告される障害数の方が多く、2002 年 10 月以降は比較的、報告される障害数よりも修正される障害数の方が多いことが分かる。また、2003 年 1 月から 3 月頃、多くの障害が一度に対応されていることが分かった。障害報告数と未対応障害数の推移に着目すると障害報告数が多い時期ほど、未対応時間の長い障害が多いことが分かった。

5. ケーススタディ2: Mozilla Firefox Project

本章では4章に続き、Firefox プロジェクト [11] を対象としたケーススタディについて述べる。5.1 節では分析対象である Firefox プロジェクトの歴史と障害管理方法について述べる。そして、5.2 節で分析結果について報告する。

5.1 Firefox プロジェクトの歴史と障害管理形態

Firefox は Mozilla プロジェクトにより開発が行われてる。このプロジェクトは 1998 年、Netscape Communications 社が自社で使用していた Web ブラウザをオープンソース化した時に立ち上げられた。これまでは Web ブラウザ、メール、HTML エディタをセットにした「Mozilla」を配布していた。しかし、機能が追加されるに伴い動作速度が低下するなどの理由で、2002 年頃から Web ブラウザである「Firefox」とメールである「Thunderbird」を分離して開発している。現在、Mozilla プロジェクトは多くの企業からの支援を受け、非営利法人 Mozilla Foundation として、2004 年に Firefox 1、2006 年に Firefox 2、そして 2008 年 6 月に Firefox 3 をリリースした。Firefox 3 はリリース後、わずか 24 時間でダウンロード数が 800 万件を超えるほど多くの人に利用されている。

Mozilla プロジェクトでは Firefox と Thunderbird に分離して開発が行われ始めた頃から Bugzilla を導入して障害管理を行っている。Firefox の障害を発見した際には、まず、Firefox プロジェクトがあらかじめ用意しているチェックリストを用いて問題の解決を図り、それでも解決しない場合、Bugzilla に報告するように定められている。

Firefox プロジェクトでは週に一度、Bug Days と呼ばれる会議を設けている。Bug Days には世界中のプロジェクト参加者が Internet Relay Chat(IRC) を用いて重要度、優先度の高い障害の修正方法や、長い間対応されていない障害の対処方法について議論が行われている。

Mozilla プロジェクトにおける Bugzilla の修正プロセスを図 12 に示す。楕円は STATUS を示し、四角は RESOLUTION を示す。図 1 に示す一般的な Bugzilla の修正プロセスと異なり、CLOSED という状態が無い。

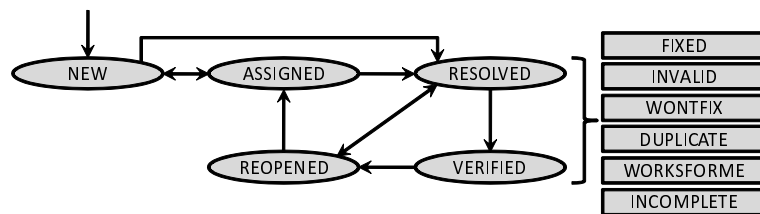


図 12 Mozilla における障害修正プロセス

表 8 Firefox の分析対象データの概要

分析対象障害数	2376 件
データ取得開始日	2001 年 1 月 1 日
データ取得終了日	2008 年 12 月 10 日

表 9 Firefox における障害の滞留時間

	滞留時間	未対応時間	対応時間
平均 (日)	73.95	7.97	65.61
中央値 (日)	14.19	0.05	10.04
標準偏差	161.99	37.15	154.84
分散	26228.80	1379.24	23965.15
最大値 (日)	2372.04	478.96	2333.01
最小値 (日)	0.00	0.00	0.00

5.2 分析結果

5.2.1 基本統計量

Firefox における障害の分析対象データの概要を表 8 に示す。修正が完了した障害 (status が CLOSED もしくは VERIFIED である障害) は 2376 件存在した。コメントが 1 回も投稿されなかった障害は存在しないため、本論文では 2376 件の障害を分析対象とする。また、障害の滞留時間、未対応時間、対応時間に関するそれぞれの統計量を表 9 に示す。

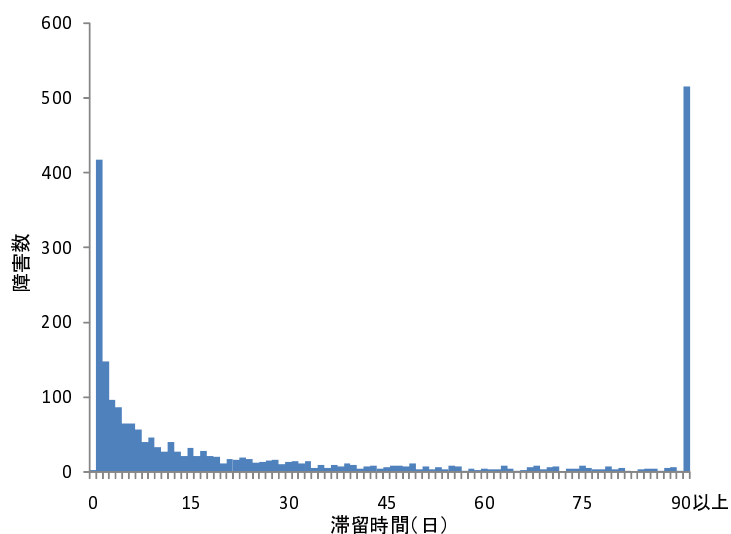


図 13 滞留時間別の障害数の分布 (Firefox)

Firefox における滞留時間別の障害数の分布と未対応時間別の障害数の分布を図 13 と図 14 に示す。図 13 の横軸は障害の滞留時間であり，図 14 の横軸は障害の未対応時間である。ヒストグラムのデータ区間は共に 1 日である。また，滞留時間（または未対応時間）が非常に長い（90 日以上）障害については一つの階級にまとめている。

未対応時間の中央値が 0.05 日であることから，半数以上の障害は 1 日以内に対応がなされていることが分かる。しかし，3 か月以上対応がなされていない障害は 42 件存在し，中には 1 年以上対応されていない障害が存在していることが分かった。

5.2.2 未対応時間と滞留時間

障害の未対応時間と対応時間の関係を図 15 に散布図で示す。横軸を未対応時間，縦軸を対応時間とする。未対応時間，対応時間ともに 900 日以内の分布を示す。

図 15 より，未対応時間が 7 日以内の障害が 87% 存在していることが分かった。また，未対応時間が長く，対応時間が 1 日以内の障害が存在するが，未対応時間が

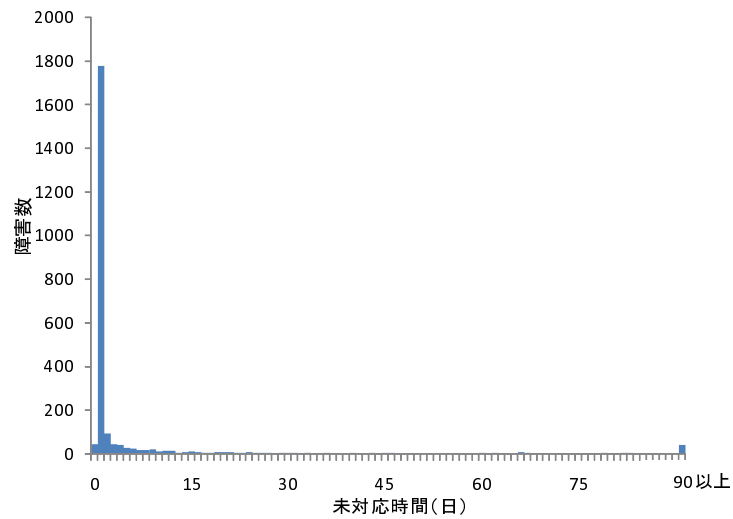


図 14 未対応時間別の障害数の分布 (Firefox)

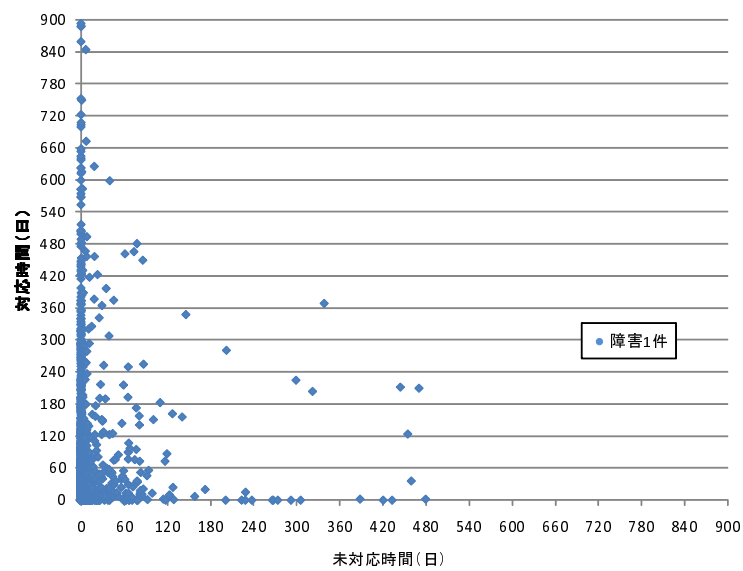


図 15 障害の未対応時間と対応時間の関係 (Firefox)

表 10 重要度別の障害数 (Firefox)

重要度		障害数 (割合)	
高い	blocker	67(2.8%)	445(18.7%)
	critical	109(4.6%)	
	major	269(11.3%)	
低い	normal	1652(69.5%)	1931(81.3%)
	minor	155(6.5%)	
	normal	124(5.2%)	
全体		2376(100%)	2376(100%)

短く対応時間の長い障害の方が多く存在していることが分かった。つまり、Firefox では、障害の滞留時間は対応されていない時間よりも、実際の修正活動に費やしている時間の方が多くを占めているといえる。

5.2.3 重要度，優先度による未対応時間

- 重要度

障害数を重要度別に表 10 に示す。また、重要度別の対応された障害数の累積グラフを図 16 に示す。横軸を未対応時間、縦軸を対応された障害の割合とする。ただし、半数以上の障害の未対応時間は 1 日以内である一方で、最大値は 478 日と偏った分布をしていることから、横軸を対数軸で表す。分析の結果、重要度の高い障害のうち約 87%、重要度の低い障害のうち約 74%が報告されてから 1 日以内に対応されていることが分かった。重要度が低い障害は高い障害に比べて、対応が遅いことが分かる。

- 優先度

障害数を優先度別に表 11 に示す。図や表中の”-”は、障害報告の際に優先度が指定されなかった障害のことを表す。また、優先度別の対応された障害数の累積グラフを図 17 に示す。横軸を未対応時間、縦軸を対応された障

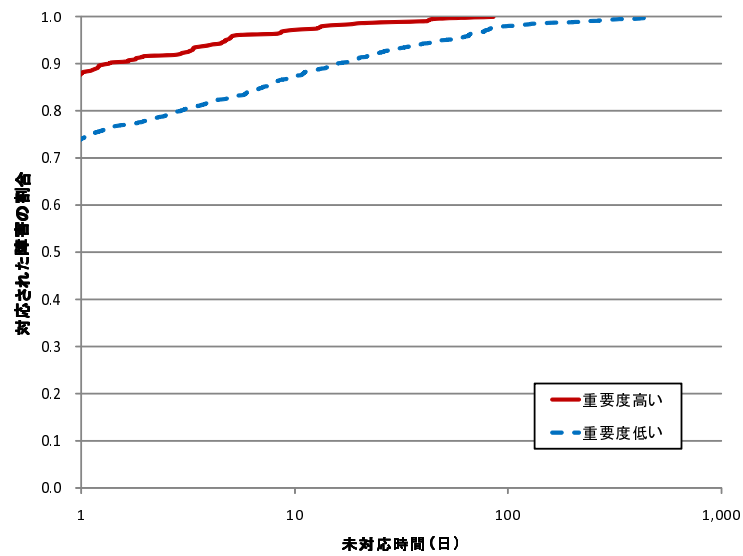


図 16 重要度別の対応された障害数の累積グラフ (Firefox)

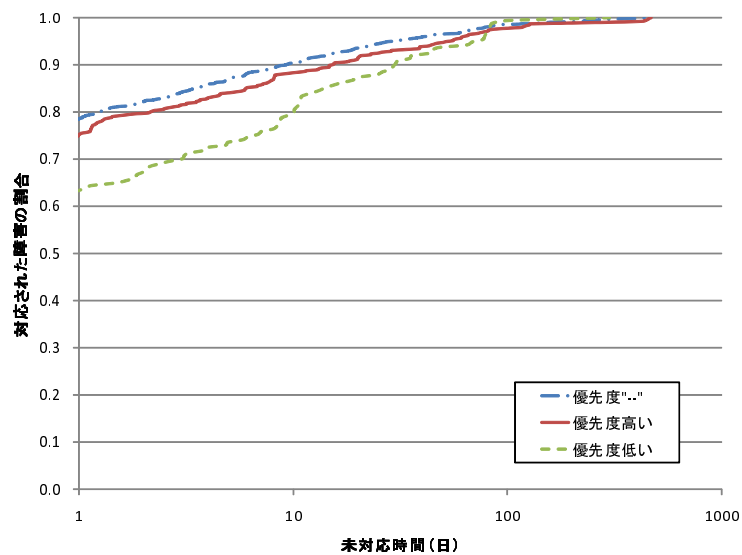


図 17 優先度別の対応された障害数の累積グラフ (Firefox)

表 11 優先度別の障害数 (Firefox)

優先度		障害数 (割合)	
高い	P1	221(9.3%)	535(22.5%)
	P2	314(13.2%)	
低い	P3	122(5.1%)	174(7.3%)
	P4	40(1.7%)	
	P5	12(0.5%)	
—	—	1667(70.2%)	1667(70.2%)
全体		2376(100%)	2376(100%)

害の割合とする。ただし、重要度の分析と同様に、半数以上の障害の未対応時間は1日以内である一方で、最大値は478日と偏った分布をしていることから、横軸を対数軸で表す。分析の結果、優先度が指定されていない障害がほとんどであることが分かった。また、これら優先度の指定されていない障害は他の障害と比べて、対応が早いことが分かった。次に、優先度の高い障害と低い障害を比べると、未対応時間が1日以内の障害は優先度の高い障害が約75%、優先度の低い障害が約63%存在していた。優先度が低い障害は高い障害に比べて、対応が遅いことが分かった。

重要度、優先度の高い障害は対応が早いことが分かったが、重要度、優先度が高いにも関わらず長期にわたって対応がなされていない障害も存在していることが分かった。例えば、障害滞留時間が1か月以上で重要度が高い障害は5件(障害全体のうち0.2%)、優先度が高い障害は37件(障害全体のうち1.5%)存在していた。これらの障害の未対応時間がなぜ長くなったのかを明らかにするために障害に対するコメントの内容を分析した。多かったコメントの内容をいくつか挙げる。

- どのような対応を行うかを指摘するアドバイス
- 修正した、またはパッチを作ったという報告

未対応時間が長く、優先度の高い障害の中に、自動生成したコメントが存在し

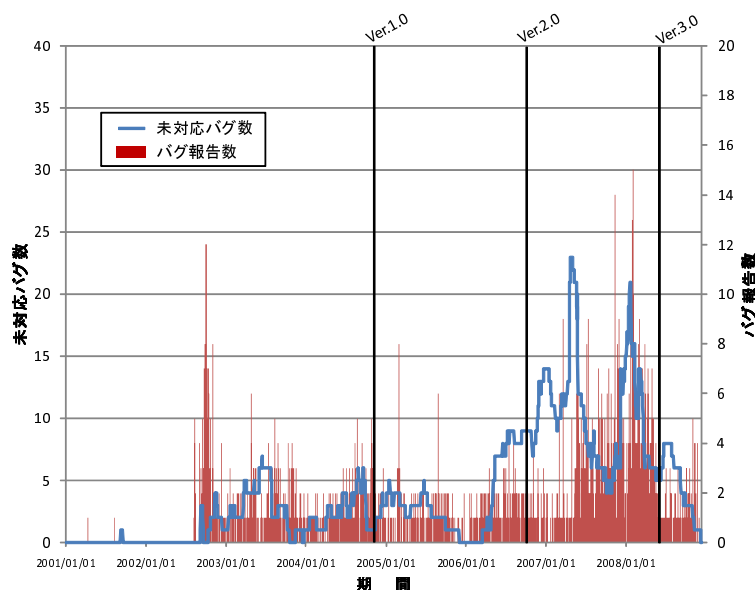


図 18 障害報告数と未対応時間が1か月以上の障害の推移 (Firefox)

た。内容は、「3か月以上誰からも修正プロセスの変更、コメントがない場合、解決手段がないものと見なし、今後2週間以内に何らかの変更等がなければ、自動的に解決とする」といったものである。このようなコメントが投稿された障害は2件存在し、ともに約10か月対応されず、同時期に報告された障害であることが分かった。

5.2.4 障害報告数と未対応障害数

未対応時間が1か月以上の障害数の推移を図18に示す。未対応障害数を折れ線グラフ、障害報告数を棒グラフで示す。横軸を期間とし、左側の縦軸を未対応障害数、右側の縦軸を障害報告数とする。障害の未対応時間が1週間以上であるグラフは付録の図22に示す。Bugzillaが導入された2002年中旬頃から障害報告は行われ始め、2005年の障害報告はやや少なかったが(173件)、2003年、2004年、2006年は年間250件前後の障害が報告されている。しかし、2006年10月にver.2.0がリリースされて以降、障害報告の数が増加し、2007年は626件の障害が報告されていることが分かった。そして、2008年6月にver.3.0がリリースされ

る直前から障害報告数が減少していることが分かった。また、未対応障害数についても同様の傾向が見られ、2002 年中旬ころから ver.2.0 がリリースされる前後までは未対応障害は 50～100 件であった、その後、未対応障害数は増加し、ピーク時の 2008 年 1 月頃に 200 件程度の障害を抱えていたが、以降は減少傾向にある。2002 年中旬頃から 2006 年 10 月の ver2.0 がリリースされるころまでは、未対応障害数が 10 件以内に維持していることから、障害の報告数と同程度の障害数が修正されているといえる。しかし、ver.2.0 がリリースされて以降、修正される障害数よりも報告数の方が多い。また、2007 年 3 月頃、2008 年初旬の短期間に多くの障害が修正されていることが分かる。さらに、障害報告数と未対応障害数の推移から障害報告数が多い時期ほど、障害未対応時間は長くなる傾向にあることが分かった。

6. 考察

本章では4章, 5章で行ったケーススタディから得た結果を基に考察を行う. 6.1節では障害の未対応時間と滞留時間についての分析結果に基づいた考察, 6.2節では重要度, 優先度による未対応時間についての分析結果に基づいた考察, 6.3節では障害報告数と未対応障害数についての分析結果に基づいた考察を述べる.

6.1 未対応時間と滞留時間

Apache, Firefox 共通して, 対応時間1日以内で, 未対応時間が長い障害が多く存在していることが分かった. このような障害は対応されてから修正が完了するまでの時間は短い, 報告されてから対応するまでの時間が長い障害であり, 早急に対応されていれば, 滞留時間が長くなることが避けられた可能性が高い. また, Apache と Firefox を比較すると Apache に比べて Firefox の方が未対応時間が短いことが分かった. 言い換えると, Firefox は障害報告後, 掲示板に書き込む時間が早いということである. Firefox の方が未対応時間が短い理由としては, (1) 開発者数の違い, (2) 掲示板を利用する意識の高さの違い, (3) 障害対応の形態の違いなどの理由が考えられる.

1. 開発者数の違い

Apache と Firefox ではソフトウェアとしての性質が異なるため, 開発に求められる要求や技術者にとって開発に参加するモチベーションが異なると考えられる. Apache はサーバのソフトウェアということもあり, 開発期間が長くなると, システムを運用するための新たな機能の追加よりも, システムの質を高くすることが求められる. 一方, Firefox はシステムを運用するためのソフトウェアに加え, アドオンなど拡張機能の開発が求められる. つまり Firefox よりも Apache の方が高い技術力が求められるため, Firefox の方が開発に気軽に参加することが可能であり, 開発者が多いと考えられる. 開発者が多いと, 掲示板に書き込む開発者も多くなることから, 障害報告後のコメントも早くなり, 未対応時間が短縮され则认为られる.

2. 掲示板を利用する意識の高さの違い

本論文で考える掲示板を利用する意識の違いは掲示板に書き込まれるコメント数の違いである。掲示板を利用するにあたって、コメント数やコメントを行う時間についての規定はない。よって、掲示板を利用する意識が高ければ、掲示板に頻繁にコメントが書き込まれ、障害への対応時間も早くなると考えられる。そこで、Firefox の開発者の方が Apache の開発者よりも掲示板を利用する意識が高いかどうかを明らかにするために、各障害に対するコメント数の分布を分析する。Apache, Firefox におけるコメント数別の障害数の分布をそれぞれ図 19, 図 20 に示す。横軸はコメント数であり、ヒストグラムのデータ区間は共に 1 回である。また、Apache, Firefox のコメント数に関する統計量を表 12 に示す。コメント数は Apache に比べて Firefox の方が多いことが分かった。特に、各障害に対するコメント数の最頻値が Apache では 1 回であるのに対し、Firefox では 6 回という結果が出ている。Apache に比べ Firefox の方がソフトウェアの性質上、開発を行う上で議論が必要である可能性はあるが、Firefox では各障害に書き込まれるコメント数が多いことから、Apache よりも Firefox の開発者の方が掲示板を利用する意識が高いことが考えられる。

3. 障害対応の形態の違い

Firefox は 1 週間に一度、前の 1 週間に投稿された障害に関して、IRC を利用して議論する機会 (Bug Days と呼ばれる会議) を設けている。一方、Apache では定期的に議論をする機会を設けられていない。OSS の多くの開発者はボランティアで働いているため、報告されてすぐに対応を行うとは限らない。よって、障害の未対応時間を短縮するためには Firefox のように定期的に報告された障害に関する議論を設け、対応されていない障害に関しての議論を行う必要がある。

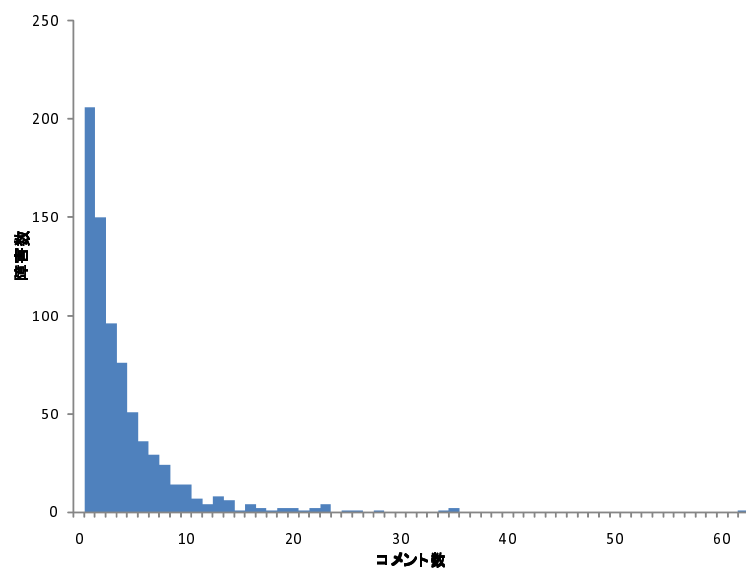


図 19 コメント数別の障害数の分布 (Apache)

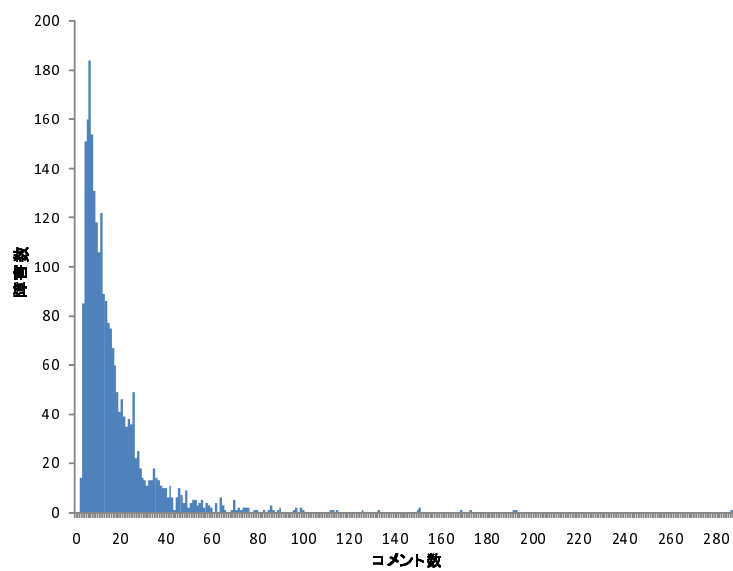


図 20 コメント数別の障害数の分布 (Firefox)

表 12 コメント数の統計量

	Apache	Firefox
平均 (回)	4.22	16.56
中央値 (回)	3.00	12.00
最頻値 (回)	1.00	6.00
標準偏差	4.95	15.14
分散	24.51	228.97
最大値 (回)	62.00	150.00
最小値 (回)	1.00	2.00

6.2 重要度，優先度による未対応時間

重要度や優先度が低い障害は高い障害に比べて，未対応時間が長いことが分かった．どのような障害を優先的に対応するかに対しては，重要度や優先度が少なからず影響していると言える．そのため，発見した障害を報告する際には重要度や優先度の指定を慎重に行うべきである．

また，未対応時間が長く，重要度，優先度の高い障害のコメントの内容を分析した結果，関連した障害についての指摘や修正方法に関するアドバイスが投稿されたケースが多く見られた．障害の報告者が高い重要度や高い優先度を指定したとしても，障害に関する指摘やアドバイスなどの対応が遅れると，対応が遅いプロジェクトとみなされてしまう可能性がある．報告者はボランティアで開発，障害報告を行っているため，対応が遅れるとプロジェクトからの離脱が考えられる．よって，障害報告後の対応は早期に行うべきである．指摘やアドバイスの他にも，修正やパッチを作ったコメントも多く見られた．一方で，報告者の技術では修正できないという報告もあった．本論文では修正された障害のみを対象としたが，修正されていない障害の中には，報告者が対応できず，誰にも対応されない障害も多く存在すると考えられる．よって，障害報告を行う際，報告者自身で修正可能な障害かどうかを明示しておく，より早期に対応されると考える．

6.3 障害報告数と未対応障害数

Apache, Firefox に共通して、障害報告数が多い時期ほど、未対応時間は長くなる傾向にあることが分かった。障害報告が多くなると開発者が全ての障害報告に対応できなくなり、対応されない障害が増加する。つまり、障害報告が増加傾向にある時に報告しても対応してもらえない可能性がある。そこで、障害の報告数を常に把握し、報告が多い時は細かい作業分担を行うことが一つの対応策であると考ええる。また、Firefox は Apache に比べ障害報告が多いにも関わらず未対応障害数が少ないことが分かった。Firefox は週に一度障害の対処に関する議論を行っていることから、未対応障害数が少ないという結果が得られたと考えられる。よって、定期的に報告された障害に関する会議を設けることが、障害の早期解決及びソフトウェア全体としての信頼性の向上に繋がると考えられる。

また、Apache では短期間に多くの障害修正が行われている時期があることが分かった。Apache の開発者が利用するメーリングリストの中に障害の修正が一度に行われていたことを示す履歴が残っている。そのメールは 2003 年 2 月頃に投稿されたメールであり、「報告されたが CLOSED されていない障害を検索し、順番に対応している」と記述されている。このメールが投稿された時期と同時期の 2003 年 1 月から 3 月頃に、これまで対応されていなかった障害の約 50% が対応されている。これは障害管理システムに登録された障害のうち、未対応であった障害を短期間に一度に対応したことが考えられる。障害の早期解決とソフトウェアの信頼性を向上させるためには、このような一時期にまとめて対処するのではなく、Firefox のように定期的に会議を設けて、未対応の障害に対処するべきである。

6.4 本論文の制約

本論文では各障害において 1 回目のコメントが投稿されるまでを障害の未対応時間とした。ただし、1 回目のコメントが修正活動の開始時間とは限らない。例えば、1 回目のコメントを報告者自身が障害情報の追加として書いている場合もある。この場合、1 回目のコメントが修正活動の開始とは言えないため、正確な修正活動の開始時間を決定するためにはコメントの内容から修正活動の開始時間

を考える必要がある。

また、本論文では2つのプロジェクトを対象としたケーススタディのみを扱った。分析結果の一般化を図るために、他のプロジェクトとの比較を行うことが今後の課題である。

7. 関連研究

7.1 障害修正の問題点

近年、OSS の進化の研究が盛んに行われており [13]、OSS の進化には OSS 開発者らによって形成されるコミュニティが重要な役割を担っていることが指摘されている。これまで OSS の進化を計測するための指標として、単一企業による開発や管理技術に基づいた指標が用いられていた。しかし、OSS 開発は企業の開発と異なる部分が多いため、同一の指標を用いるのは妥当ではない。Wang らは OSS の進化を計測するための適切な指標の必要性を挙げ、OSS の進化の特徴を考慮した指標を提案している。提案指標の一つとして、ソフトウェア中に存在している障害の数や修正された障害の数などが挙げられている。提案指標のケーススタディとして、Linux ディストリビューションである Ubuntu を対象とした分析を行った。その結果、報告されている障害のうち、修正されている障害は2割程度に過ぎず、修正されていない障害の中にはアサインさえ行われていない障害が多く存在していることが明らかとなった。

Wang らの研究により、修正までに時間がかかっている障害が多く存在していることが指摘されているが、修正されていない障害はどのような障害なのかについては明らかにされていない。また、修正までにどれくらいの時間がかかっているかについても明らかにされていない。

7.2 OSS 開発における障害の滞留時間

Mockus らは OSS 開発が成功する理由を調査するために、2つの有名な大規模プロジェクトである Apache プロジェクトと Mozilla プロジェクトの分析を行った

[10]. ユーザにとって問題が素早く解決されることが重要であることから、調査項目の一つとして障害の滞留時間を分析している。

障害に関する分析の一つとして障害の解決にはどれくらいの時間がかかっているかを分析している。具体的には、優先度により滞留時間は異なるか、また、モジュールによって滞留時間は異なるかを明らかにした。分析の結果、カーネルやプロトコルに関するモジュールや広い範囲で利用される機能を持つモジュールに存在する障害は滞留時間が短いことが分かった。また、優先度別の滞留時間は P1, P3 に指定された障害のうち 50% が約 30 日以内、P2 に指定された障害のうち 50% が約 80 日以内、P4, P5 に指定された障害のうち 50% は滞留時間が約 100 日以内であることが分かった。

また、Herraiz らは障害報告者が指定する重要度、優先度が複雑であることを問題点として挙げ、分類項目の簡略化のために重要度別、優先度別の滞留時間の違いを統合開発環境の一つである Eclipse を対象に分析した [5]。重要度別の滞留時間を分析したところ、重要度が高い障害は低い障害に比べて滞留時間が短いことが分かった。また、Eclipse プロジェクトでは重要度の分類項目を 7 種類としているが、7 種類のそれぞれに明確な分布の差がないことから、分類は Important, Non-important, Request for enhancement の 3 種類で十分であると指摘している。さらに、優先度別の滞留時間を分析したところ、優先度が P3 に指定されている障害の滞留時間が最も短いことが分かった。また、優先度に関しても、5 段階の分類項目それぞれに明確な分布の差がないことから、3 段階の分類で十分であると指摘している。

これらの研究は障害が報告されてから修正が完了されるまでの時間の違いを分析している。一方、本論文では未対応時間が長いために滞留時間が長くなっているという指摘から、障害の未対応時間が長い障害がどういう障害なのかを明らかにすることを目的として分析を行った。

8. 結論

本論文では障害の滞留時間を短縮するために、未対応時間が長い障害がどのような障害なのかを明らかにすることを目的として分析を行った。Apache プロジェクトと Firefox プロジェクトのケーススタディを行った結果、得られた主な知見は以下の通りである。

- 短時間で修正可能な障害であるにも関わらず、対応されるまでに多くの時間を要している障害が多く存在していることが分かった。これらの障害は短い時間で修正が可能であるにも関わらず、長い期間放置されていた障害であるため、早期対応により障害の遅延を避けられた可能性がある。
- 重要度や優先度が低い障害は高い障害に比べて、未対応時間が長いことが分かった。どのような障害を優先的に対応するかに対しては、重要度や優先度が少なからず影響しているといえる。そのため、発見した障害を報告する際には重要度や優先度の指定を慎重に行うべきである。
- 定期的に障害に関する会議を設ける Firefox では会議を設けていない Apache に比べ未対応の障害の数が少ないことが分かった。よって、滞留時間短縮のためには障害に関する会議を定期的に設けることが一つの改善点であると考えられる。

本論文で得られた知見を用いることで、BTS を用いる他の OSS 開発において障害の修正遅延を低減できると考えられる。ただし、これらの知見の一般化のためには、障害管理システムを利用する他のソフトウェアをケーススタディとして分析する必要がある。

今後、企業の障害情報を用いて分析し、企業における障害の滞留時間を短縮するきっかけとなることを期待する。

謝辞

本研究を進めるに当たり多くの方々に、御指導、御協力、御支援を賜りました。この場を借りてお世話になった方々に感謝の意を表したいと思います。

主指導教員であり本論文の審査委員を務めて頂いた、奈良先端科学技術大学院大学 情報科学研究科 松本 健一 教授に対し、厚く御礼を申し上げます。大学院入学以来、様々な視点から御指導して頂き、研究者としての心構えや、研究者としての在り方など数多くの御指導を賜りました。また、研究以外の様々な活動の場でも支援して頂きました。深く感謝致します。

副指導教員であり本論文の審査委員を務めて頂いた、奈良先端科学技術大学院大学 情報科学研究科 関 浩之 教授には本研究を進めるに当たり、非常に細やかな御指導を賜りました。また、学内の発表では多数の御質問と御指導を頂きました。心より感謝致します。

副指導教員であり本論文の審査委員を務めて頂いた、奈良先端科学技術大学院大学 情報科学研究科 門田 暁人 准教授には本論文に対して多くの建設的なアドバイスを賜りました。研究以外の様々な活動の場においてもお世話になり、心より感謝致します。

副指導教員であり本論文の審査委員を務めて頂いた、奈良先端科学技術大学院大学 情報科学研究科 大平 雅雄 助教に対し、厚く御礼申し上げます。大学院入学以来二年間、研究、進路など様々な面で御指導頂きました。研究に関しては本研究をまとめるに至るまで、何度もくじけそうになった私に数々のご助言をしてくださいました。学会発表に参加する際、論文執筆の作法から発表のアドバイスなど懇切丁寧に御指導して頂きました。研究者としての心構えだけでなく、将来については何度もご相談に乗って頂きました。深く感謝致します。

奈良先端科学技術大学院大学 情報科学研究科 森崎 修司 助教には本論文について一歩離れたところから有益な御指摘、御意見を頂きました。また、社会における研究者の位置づけ、研究者としての心構えを御指導して頂きました。心より感謝致します。

奈良先端科学技術大学院大学 情報科学研究科 角田 雅照 特任助教には本論文について一歩離れたところから有益な御指摘、御意見を頂きました。研究だけで

なく、日常生活においても心優しく接して頂きました。心より感謝致します。

奈良先端科学技術大学院大学 ソフトウェア工学講座 亀井 靖高 氏には論文執筆に対するアドバイス、プレゼンテーションの作法等、適切かつ建設的なアドバイスを頂きました。入学当初、未熟であった私を研究にのみならず、大学生活においても御指導して頂きました。氏からの励ましがあったからこそ、二年間諦めずに研究を続けていくことができました。深く感謝致します。

奈良先端科学技術大学院大学 ソフトウェア工学講座 杉本 真佑 氏には論文執筆に対するアドバイス、プレゼンテーションの作法等、適切かつ建設的なアドバイスを頂きました。氏の研究に対する真摯な姿勢に心打たれ、研究意欲を駆り立てられました。深く感謝致します。

研究室の卒業生である大阪大学大学院の柿本 健 特任助教、日本電気(株)の前島 弘敬 氏、(株)日立製作所の大西 洋司 氏には、1年という短い間でしたが同じ研究グループで、まだまだ未熟な私を懇切丁寧に御指導して頂き、心より感謝致します。

奈良先端科学技術大学院大学 ソフトウェア工学講座の皆様には日頃より多大な御協力と御助言を頂きました。研究においては皆様からの御支援、励ましが私の大きな支えとなりました。研究のみならず大学院の生活においても、皆様の明るさ、優しさにより楽しい二年間を過ごすことができました。深く感謝致します。

その他にも、研究発表の際に議論させて頂いた方々に心より感謝致します。

最後に、大学院において研究するに当たり、私を励まし、経済的にも精神的にも支えてくれた家族に深く感謝致します。

参考文献

- [1] Apache HTTP Server Project, <http://httpd.apache.org/>.
- [2] Bugzilla.org, <http://www.bugzilla.org/>.
- [3] Chiara, F., and Francesco, M.: Impact analysis by mining software and change request repositories. Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS2005), pp.19–22, 2005.
- [4] Fischer, M., Pinzger, M., and Gall, H.: Analyzing and relating bug report data for feature tracking. 10th Working Conference on Reverse Engineering (WCRE2003), pp.90–99, 2003.
- [5] Herraiz, I, German, D.M., Gonzalez-Barahona, J.M., and Robles, G.: Towards a simplification of the bug report form in eclipse. Proceedings of the 2008 International Working Conference on Mining Software Repositories (MSR2008), pp.145–148, 2008.
- [6] Life Cycle of a Bug, <http://www.bugzilla.org/docs/3.2/en/html/lifecycle.html>.
- [7] MANTIS, <http://www.mantisbt.org/>.
- [8] Matsumoto, S., Kamei, Y., Ohira, M. and Matsumoto, K.: A comparison study on the coordination between developers and users in foss communities, Socio-Technical Congruence (STC2008), No.8, pp.1–9, 2008.
- [9] Michlmayr, M., Hunt, F., and Probert, D.: Quality and the reliance on individuals in free software projects, Proceedings of the 3rd Workshop on Open Source Software Engineering (WOSSE2005), pp.105–109, 2005.
- [10] Mockus, A, Fielding, R.T., and Herbsleb, J.D.: Two case studies of open source software development: apache and mozilla, ACM Transactions on Software Engineering and Methodology, Vol.11, No.3, pp.309–346, 2002.

- [11] Mozilla Project, <http://www.mozilla.org/>.
- [12] Trac, <http://trac.edgewall.org/>.
- [13] Wang, Y., Guo, D., and Shi, H.: Measuring the evolution of open source software systems with their communities, SIGSOFT Software Engineering Notes 32, No.6, 2007.
- [14] 松村知子, 門田暁人, 森崎修司, 松本健一.: マルチベンダ情報システム開発における障害修正工数の要因分析, 情報処理学会論文誌, Vol.48, No.5, pp.1926–1935, 2007.

付録

A. 障害報告数と未対応時間が1週間以上の障害の推移

A.1 Apache

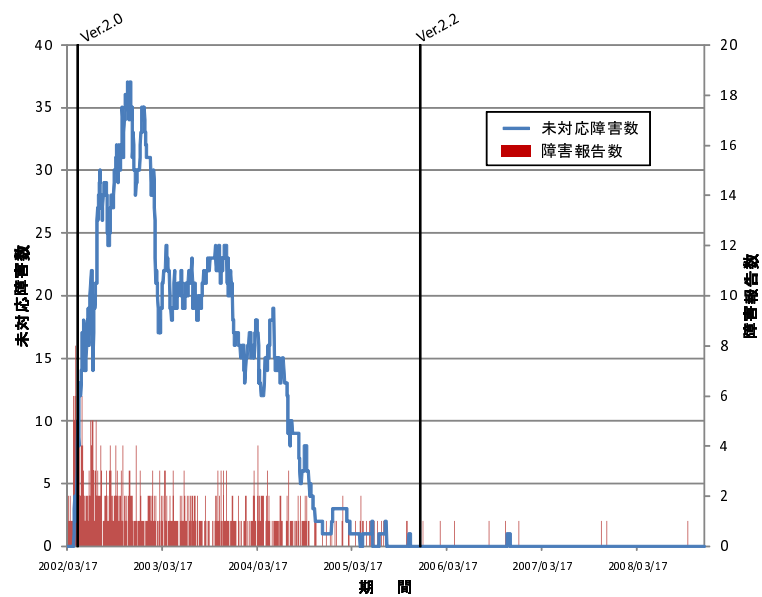


図 21 障害報告数と未対応時間が1週間以上の障害の推移 (Apache)

A.2 Firefox

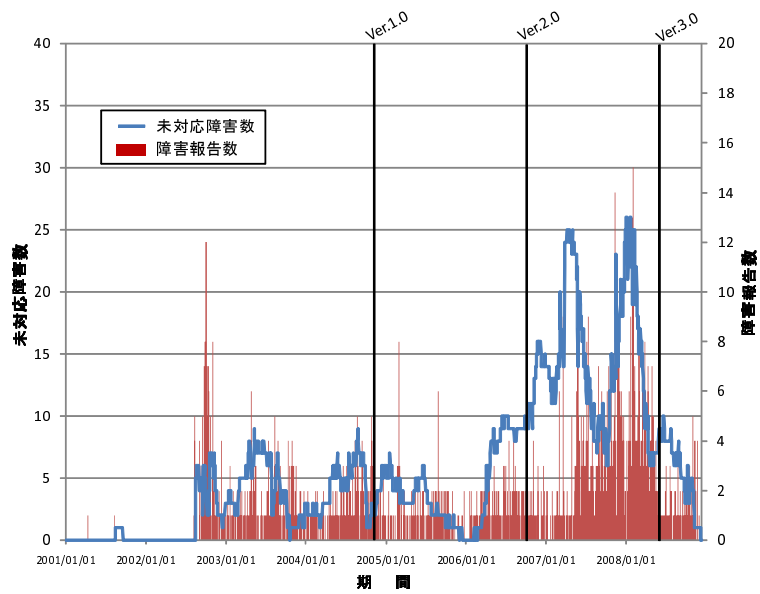


図 22 障害報告数と未対応時間が1週間以上の障害の推移 (Firefox)