

修士論文

ソフトウェア開発における障害の滞留時間に 及ぼす要因の分析

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)

ソフトウェア開発における障害の滞留時間に 及ぼす要因の分析*

伊原 彰紀

内容梗概

ここには内容梗概を書く。ここには内容梗概を書く。ここには内容梗概を書く。
ここには内容梗概を書く。ここには内容梗概を書く。ここには内容梗概を書く。
ここには内容梗概を書く。ここには内容梗概を書く。ここには内容梗概を書く。
ここには内容梗概を書く。ここには内容梗概を書く。ここには内容梗概を書く。
ここには内容梗概を書く。ここには内容梗概を書く。ここには内容梗概を書く。

キーワード

バグトラッキングシステム, 情報共有, バグ滞留時間

*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.

hogehoge^{*}

Akinori Ihara

Abstract

This is a sample abstract. This is a sample abstract. This is a sample abstract.
This is a sample abstract. This is a sample abstract. This is a sample abstract.
This is a sample abstract. This is a sample abstract. This is a sample abstract.
This is a sample abstract.

Keywords:

aaa,bbb,ccc

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

目次

1. はじめに	1
1.1 研究の背景	1
1.2 本研究の目的とアプローチ	1
1.3 本論文の構成	2
2. 準備	3
2.1 バグ管理システム	3
2.2 バグの修正プロセスとバグへのコメントの関係	4
2.3 滞留時間の定義	4
3. 分析方法	6
3.1 滞留時間と初期対応時間の長いバグの決定	6
3.2 滞留時間と初期対応時間の分析	6
3.3 重要度, 優先度の分析	7
3.4 報告時期の時系列分析	8
4. ケーススタディ	9
4.1 分析対象 OSS の概要と基本統計量	9
4.1.1 Apache HTTP Server Project 概要	9
4.1.2 Apache HTTP Server Project 基本統計量	10
4.1.3 Mozilla Firefox Project 概要	10
4.1.4 Mozilla Firefox Project 基本統計量	12
4.2 滞留時間と初期対応時間の長いバグの決定	13
5. ケーススタディ1:Apache HTTP Server Project の結果	16
5.1 滞留時間と初期対応時間の分析	16
5.2 重要度, 優先度の分析	17
5.3 報告時期の時系列分析	19

6. ケーススタディ2:Mozilla Firefox Project の結果	21
6.1 滞留時間と初期対応時間の分析	21
6.2 重要度, 優先度の分析	22
6.3 報告時期の時系列分析	24
7. 考察	25
7.1 滞留時間と初期対応時間の分析	25
7.2 重要度分析	26
7.3 報告時期の時系列分析	27
8. 関連研究	27
8.1 OSS における情報共有についての研究	27
8.2 Bugzilla で行われるコメントについての研究	27
8.3 滞留時間についての研究	28
8.4 Bugzilla の優先度や重要度による滞留時間の違いについての研究 .	28
9. 今後の展望	29
謝辞	30
付録	31
A. おまけその1	31
B. おまけその2	31
参考文献	31

図 目 次

1	バグ (Bugzilla) の修正プロセス	3
2	滞留時間, 初期対応時間, 実質対応時間の概略図	5
3	滞留時間と実質対応時間の関係を表す概略図	6
4	Apache のバグ修正プロセスの概略図	9
5	滞留時間のヒストグラム (apache)	11
6	初期対応時間ヒストグラム (apache)	11
7	バグプロセス (mozilla)	12
8	滞留時間のヒストグラム (firefox)	14
9	初期対応時間ヒストグラム (firefox)	14
10	初期対応時間と実質対応時間の関係 (apache)	16
11	未対応バグとバグ報告数の時系列分析 (apache)	20
12	初期対応時間と実質対応時間の関係 (firefox)	21
13	残留バグとバグ報告数の時系列分析 (firefox)	25
14	コメント数のヒストグラム (apache)	26
15	コメント数のヒストグラム (firefox)	27
16	おまけの図	31

表 目 次

1	各重要度の分類	7
2	Apache の基本統計量	10
3	滞留時間の基本統計量 (apache)	10
4	Firefox の基本統計量	12
5	滞留時間の基本統計量 (firefox)	13
6	滞留時間が長いバグのうち初期対応時間別のバグの数 (apache) . .	16
7	重要度別のバグ数 (apache)	17
8	重要度別の初期対応時間が長いバグ数 (apache)	17
9	優先度別のバグ数 (apache)	18

10	優先度別の初期対応時間が長いバグ数 (apache)	19
11	滞留時間が長いバグのうち初期対応時間別のバグの数 (firefox) . .	21
12	重要度別のバグ数 (firefox)	22
13	重要度別の初期対応時間が長いバグ数 (firefox)	22
14	優先度別のバグ数 (firefox)	23
15	優先度別の初期対応時間が長いバグ数 (firefox)	24
16	コメント数の統計量	28

1. はじめに

1.1 研究の背景

近年，ソフトウェア開発は大規模化に伴い，バグの数が増加し，バグの管理が難しくなる傾向にある．バグの増加により開発者が対応しきれないためバグが放置されたり，バグの管理を正しく行わないことで各バグの修正状況の把握が困難になることが考えられる．

そこで，多くのソフトウェア開発プロジェクトではバグを一元管理するためにバグ管理システムを利用している．バグ管理システムではバグ報告時に重要度，優先度等を設定し，早急に直すべきバグについて容易に検索可能にしている．また，多くのバグ管理システムではバグごとに掲示板などが用意され，修正状況，修正方針に関する議論 (情報共有) を行うことが可能である．

しかし，バグ管理システムを利用しているにも関わらず，バグの修正が効率的に行われていないことが指摘されている．報告されているバグのうち，約8割のバグは修正が完了しておらず，多くのバグがアサイン (バグを修正する人を割り当てること) されていないため，報告してから修正されるまでの時間 (滞留時間) が長くなっていることが指摘されている [2]．このように滞留時間が長くなると，ソフトウェア開発に様々な影響を及ぼすと考えられる．修正されていないバグが頻繁に利用されるモジュールに存在すると，そのモジュール単体では問題がなかったとしても，異なるプログラムから利用されたときに，重要なバグとなる可能性がある．また，滞留時間が長くなると修正コストが高くなる可能性も高くなると指摘されている [1]

そこで，バグの滞留時間短縮に向けて，修正に取り掛かるまでに時間がかかっているかを明らかにする必要がある．

1.2 本研究の目的とアプローチ

本論文では初期対応が遅れているバグを減少させるために，初期対応が遅れているバグがどのようなバグなのかを明らかにすることを目的とする．バグが報告

された後、報告されたバグに対して早い対応を行うことがソフトウェアの品質の向上につながると考える。そして、バグへの対応の遅れを改善することで、滞留時間が短くなり、新しいバグが生まれる可能性が低くなると考える。そこで、バグに対する初期対応を早く、そして的確な修正活動を行うためには、開発者間で、バグが現在どういう修正状況なのか、これからどういう修正が必要なのか、という修正に関する情報共有を早急に行うことが必要である。

本研究ではバグへの対応を、バグに関する情報共有活動と考え、バグが報告されてからバグに関する情報共有を開始する時期を初期対応時間とし、初期対応時間が長いバグがどのようなバグかを明らかにする。

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章ではバグ管理システムとバグに関するコメントについて説明した後、本論文で扱う滞留時間についての定義を行う。そして、3章においてバグの収集データについて説明した後、収集したデータを用いてケーススタディで行う分析方法について説明する。5章ではケーススタディとしてApacheを分析し、6章ではmozillaを分析する。7章ではApacheとMozilla(firefox)に関する議論を行い、8章では関連研究を述べ本研究の立場を明らかにする。そして、最後に9章において今後の展望について述べる。

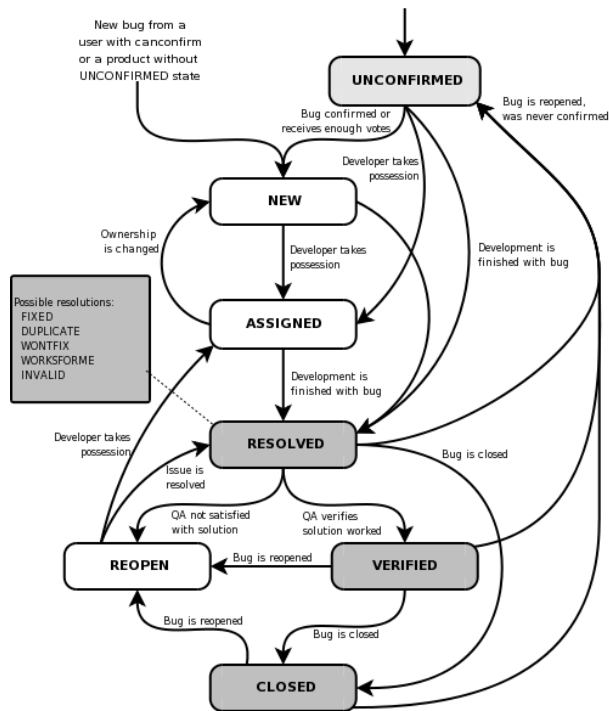


図 1 バグ (Bugzilla) の修正プロセス

2. 準備

本論文ではバグの初期対応時間に時間がかかっている要因を明らかにするためにバグ管理システムを利用した OSS をケーススタディとして分析する。本章ではバグ管理システムの概要を説明した後、修正プロセスと各バグに対するコメントについて説明する。そして、本論文で扱う滞留時間の定義を説明する。

2.1 バグ管理システム

ソフトウェア開発においてバグの修正、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、バグの数が増え、バグの管理が難しくなる傾向にある。OSS も大規模化する傾向にあり、分散開発を行っていることから、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難しい。そこで、多くの OSS ではバグ管理システムが利用されている。

開発者はバグを報告する際、プロダクト、バージョン、解決方法、重要度、優先度などを指定することができる。また、バグが報告されてから修正が終わるまでの管理をすることができ、誰によって、いつ、何を変更した等の変更された履歴が残されている。よって、修正状態や、重要度などから早急に直すべきバグの検索が可能である。

バグ管理システムには Bugzilla, Trac, Mantis など様々な種類があるが、本論文では Bugzilla を利用している OSS 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。

2.2 バグの修正プロセスとバグへのコメントの関係

Bugzilla ではバグの修正履歴の他に、各バグに対して議論 (情報共有) を行うために掲示板の機能がある。この掲示板が利用可能になるのはバグが報告された後、いつでも投稿可能である。掲示板にコメントされる内容はバグによって様々である。例えば、バグ報告者からのバグ情報だけでは修正するための情報として不足している場合、追加してバグの詳細な情報を聞くことがある。また、掲示板を利用してバグを修正しましたという報告を行う場合が多くある。修正したというコメントが投稿された後は、修正活動がほとんど行われないと考えられる。その他にも、報告される以前にメーリングリストや、類似したバグ報告があれば、掲示板を利用して情報共有される。Bugzilla を利用して報告されるバグに対する掲示板は、プロジェクト全体がバグの様子を情報共有し、バグの進捗状況、修正活動を表していることから、バグの修正履歴の一つとして考えられる。

2.3 滞留時間の定義

本論文ではバグが報告されてから修正が完了されるまでの時間を滞留時間とする。バグが報告される時間はバグをバグ管理システムに登録した時間とする。一方、修正の完了時間は複数のケースが考えられる。一つはバグの修正状態を表す Status が CLOSED に変更される時間、次に Status が VERIFIED に変更される時間、最後に Status が CLOSED に変更される直前の FIXED とされる時間であ

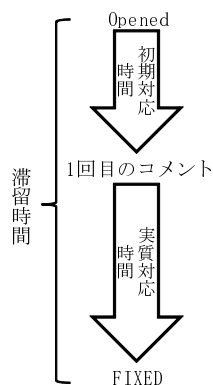


図 2 滞留時間，初期対応時間，実質対応時間の概略図

る。CLOSED や VERIFIED はバグが修正され，レビュー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 Resolution が FIXED とされた時間をバグ修正の完了時間とする。図 2 に本論文で扱うそれぞれの時間の概略図を示す。次に，初期対応時間と実質対応時間の定義をそれぞれ説明する。

- 初期対応時間

本論文ではバグが報告されてからバグに関する 1 回目のコメントが投稿されるまでの時間を初期対応時間とする。

- 実質対応時間

本論文ではバグに関する 1 回目のコメントが投稿されてから Status が CLOSED に変更される直前の Resolution が FIXED とされた時間を実質対応時間とする。

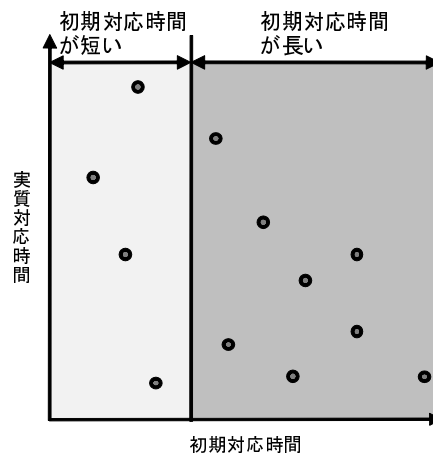


図 3 滞留時間と実質対応時間の関係を表す概略図

3. 分析方法

3.1 滞留時間と初期対応時間の長いバグの決定

本論文では OSS の信頼性に悪影響を及ぼす可能性が高いと考えられる滞留時間、初期対応時間が長いバグに着目する。滞留時間の長いバグを決定するために、本論文では各 OSS を分析する前に滞留時間と初期対応時間の分布を分析する。その後、滞留時間が長いバグを決定する。

3.2 滞留時間と初期対応時間の分析

各 OSS において、滞留時間が長いバグの中で、初期対応時間が長いバグがどれくらい存在するのか明らかにするために滞留時間と初期対応時間が長いバグの分析を行う。図 3 のように初期対応時間と実質対応時間の関係を散布図に表すことで、初期対応に時間がかかっているバグと実質対応に時間がかかっているバグを示す。そして、本論文では滞留時間に時間がかかっているバグの中で、初期対応に時間がかかっているバグがどれくらい存在するか明らかにする。

表 1 各重要度の分類

重要度	内容
Blocker	開発とテストの少なくとも一方を妨げるバグ
Critical	クラッシュ, データの損失, 深刻なメモリリークを引き起こすバグ
Major	主要な機能の欠如しているバグ
Normal	普通のバグ
Minor	マイナーな機能の欠如, または簡単な代替手段があるバグ
Trivial	単語の打ち間違い等の表面上のバグ
Enhancement	機能拡張のリクエスト

3.3 重要度, 優先度の分析

バグ管理システムには早く直すべきバグを容易に検索可能にするために, 各バグに重要度, 優先度が設定される. これは開発者が報告されたバグの重要度, 優先度を参考の一つとする機能である.

Bugzilla における重要度は各バグの深刻度を表しており, Blocker, Critical, Major, Normal, Minor, Trivial, Enhancement の 7 種類に分類されている. それぞれの内容は表 1 に示す. また, 優先度は各バグの重要性和修正すべきバグの優先順位を表しており P1(最も高い優先度) から P5(最も低い優先度) まで 5 種類に分類されている.

Herraiz I らはバグ報告者が設定する重要度, 優先度による滞留時間の違いを明らかにすることを目的とした研究を行っている [3]. 重要度別, 優先度別に滞留時間を分析したところ, 重要度が高いバグは重要度が低いに比べて滞留時間が短いという結果が得られている. 優先度別の滞留時間の分析では P3 に設定されたバグの滞留時間が最も短く, 優先度と滞留時間の長さには関係がないことが分かった. しかし, Herrera I らの研究は滞留時間の違いを分析しており, 初期対応時間とは異なる.

本論文では重要度, 優先度はバグの深刻度, 修正するための優先順位をつける

ための機能であることから、重要度、優先度により初期対応にかかる時間が異なると考え、初期対応時間が長いバグの重要度、優先度を明らかにするために分析を行う。また、重要度の高いバグに関しては、なぜ対応が遅くなったのかをコメントの内容などから検討する。

本論文では OSS に及ぼす障害を分析対象としているため、OSS の機能拡張を表す Enhancement は分析対象外とし、重要度が Blocker, Critical, Major であるバグを重要度の高いバグ, Normal, Minor, Trivial であるバグを重要度の低いバグとする。また優先度についても、P1, P2 であるバグを優先度の高いバグ, P3, P4, P5 であるバグを優先度の低いバグとする。

3.4 報告時期の時系列分析

バグ報告は常に一定量ではなく、バグ報告が多い時期もあれば、少ない時期もある。特に OSS ではソースコードを公開していることもあり、リリース後にバグ報告が多いことが考えられる。そこで、対応が遅れるバグはどのような時期に報告されたバグなのかを明らかにするために初期対応に時間がかかるバグはリリース時期と関係あるのか、そして、報告数により対応されるバグの数は変化するかを分析する。

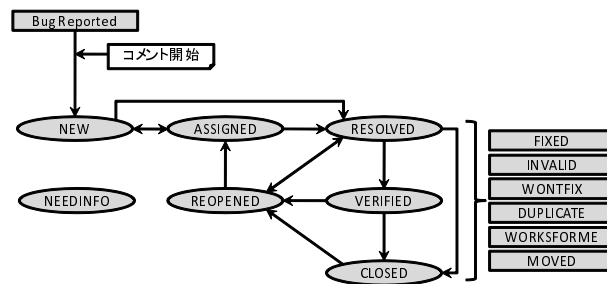


図 4 Apache のバグ修正プロセスの概略図

4. ケーススタディ

本論文ではケーススタディとして Apache HTTP Server Project と Mozilla Firefox Project を分析する．この2つのプロジェクトにおいて，特に滞留時間，初期対応時間の長いプロジェクトに着目して分析を行う．そこで，本章では各 OSS の概要と基本統計量を示す．その後，滞留時間が長いバグ，初期対応時間が長いバグを決定するために，各プロジェクトにおける滞留時間と初期対応時間の分布を分析する．

4.1 分析対象 OSS の概要と基本統計量

4.1.1 Apache HTTP Server Project 概要

Apache プロジェクトは，1995 年に NCSA (National Center for Supercomputing Applications) で Rob 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機能拡張やバグ修正を独自に行い，1996 年に Apache httpd 1.0 をリリースした．開発は今日も続いており，ver1.3，ver2.0 そして ver2.2 のリリースが行われている．Apache HTTP Server Project が使用する Bugzilla の修正プロセスを図 4 に示す．

表 2 Apache の基本統計量

基本統計量	値
取得対象バグ数	747
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 3 滞留時間の基本統計量 (apache)

基本統計量	滞留時間	初期対応時間	実質対応時間
平均	97.59	27.57	70.02
中央値	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	2153.45	751.22	1908.70
最小値	0.00	0.00	0.00

4.1.2 Apache HTTP Server Project 基本統計量

本論文で扱う Apache HTTP Server Project において対象とするバグの基本統計量を表 2 に示す。また、滞留時間、初期対応時間、実質対応時間に関するそれぞれの統計量を表 3 に示し、Apache における滞留時間のヒストグラムを図 5 に、横軸は滞留時間とし、区間を 1 日とする。縦軸はバグ数とする。Apache における初期対応時間のヒストグラムを図 6 に、横軸は初期対応時間とし、区間を 1 日とする。縦軸はバグ数とする。

4.1.3 Mozilla Firefox Project 概要

Firefox は Mozilla プロジェクトにより開発が行われてる。Mozilla プロジェクトは 1998 年、Netscape 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するまで Web ブラウザ、メーラー、HTML エディタ「をセットにした「Mozilla」を配

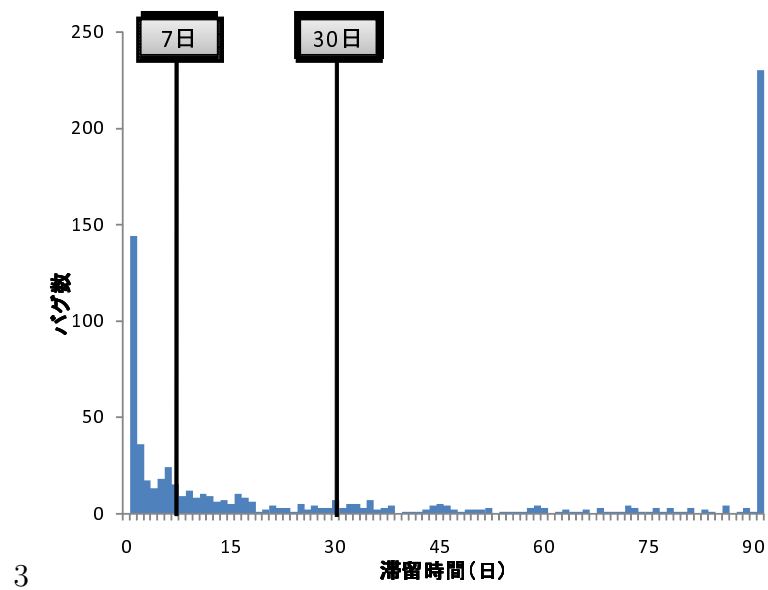


図 5 滞在時間のヒストグラム (apache)

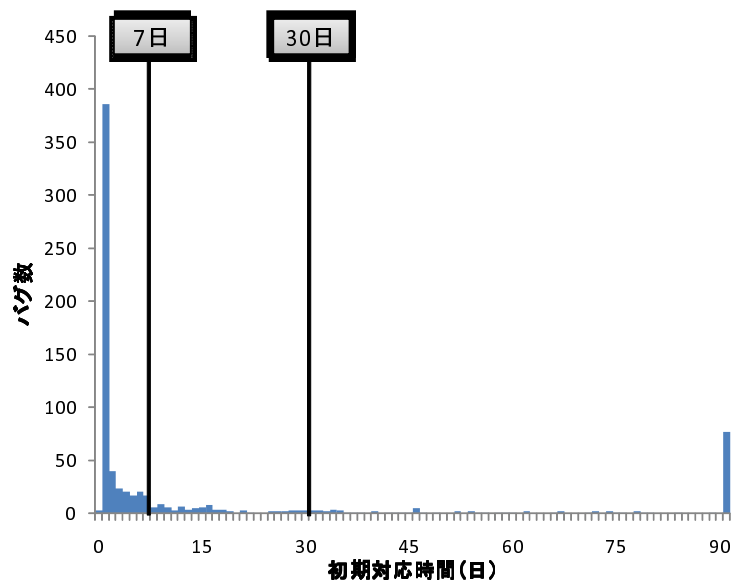


図 6 初期対応時間ヒストグラム (apache)

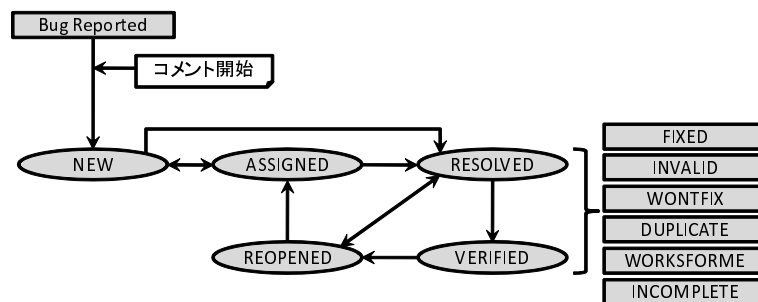


図 7 バグプロセス (mozilla)

表 4 Firefox の基本統計量

基本統計量	値
取得対象バグ数	2376
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

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 Firefox と 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 となっており、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 Foundation が利用している Bugzilla の修正プロセスを図 7 に示す。

4.1.4 Mozilla Firefox Project 基本統計量

本論文で扱う Mozilla Firefox Project において対象とするバグの基本統計量を表 4 に示す。また、滞留時間、初期対応時間、実質対応時間に関するそれぞれの統計量を表 5 に示す。Firefox における滞留時間のヒストグラムを図 8 に、横軸は滞留時間とし、区間を 1 日とする。縦軸はバグ数とする。Firefox における初期対応時間のヒストグラムを図 9 に、横軸は初期対応時間とし、区間を 1 日とする。

表 5 滞留時間の基本統計量 (firefox)

基本統計量	滞留時間	初期対応時間	実質対応時間
平均	73.95	7.97	65.61
中央値	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	2372.04	478.96	2333.01
最小値	0.00	0.00	0.00

縦軸はバグ数とする。Firefox における実質対応時間のヒストグラムを図 9 に、

4.2 滞留時間と初期対応時間の長いバグの決定

滞留時間、初期対応時間の分布から平均値と標準偏差より、それぞれのプロジェクトのデータの分布を基に滞留時間と初期対応時間の長いバグの決定することを考える。図 5, 図 8, 図 6, 図 9 をそれぞれ分析すると、どのヒストグラムも 0 日に偏っており、正規分布の形を成していないことが分かる。よって、平均値と標準偏差の和から滞留時間、初期対応時間の長いバグを導出するのは妥当ではない。

そこで、本論文ではまず初期対応時間が長いバグを Apache と Firefox のそれぞれの初期対応時間の平均値である 27.57 日 (約 1 か月) 以上のバグと、7.97 日 (約 1 週間) 以上のバグと考える。Apache, Firefox のそれぞれの初期対応時間のヒストグラムを分析すると、多くは 1 週間、1 か月以内であり、Apache, Firefox 共に OSS でありながら、日にち、曜日、時間関係なく、積極的に活動を行っている開発者が多く存在するため、初期対応時間が 1 週間、1 か月かかるのは対応が遅いと考えられる。

次に滞留時間のバグの決定を行う。コメントの中には 1 回目のコメントで修正完了の報告も存在するため、この場合、初期対応時間と滞留時間がほとんど変わらない。よって、本論文では初期対応時間に着目していることから、滞留時間の長いバグは初期対応時間の長いバグの条件と同様に、滞留時間の長いバグとして

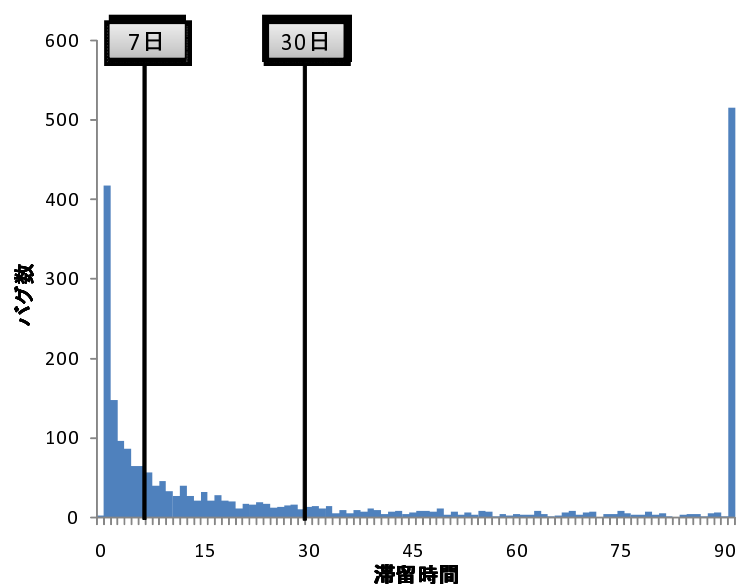


図 8 滞在時間のヒストグラム (firefox)

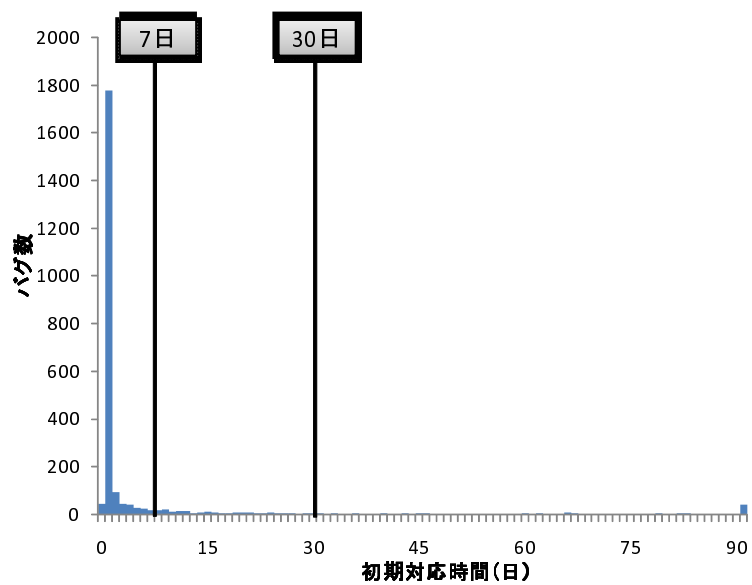


図 9 初期対応時間ヒストグラム (firefox)

1 週間以上のバグ, 1 か月以上のバグを分析することにする.

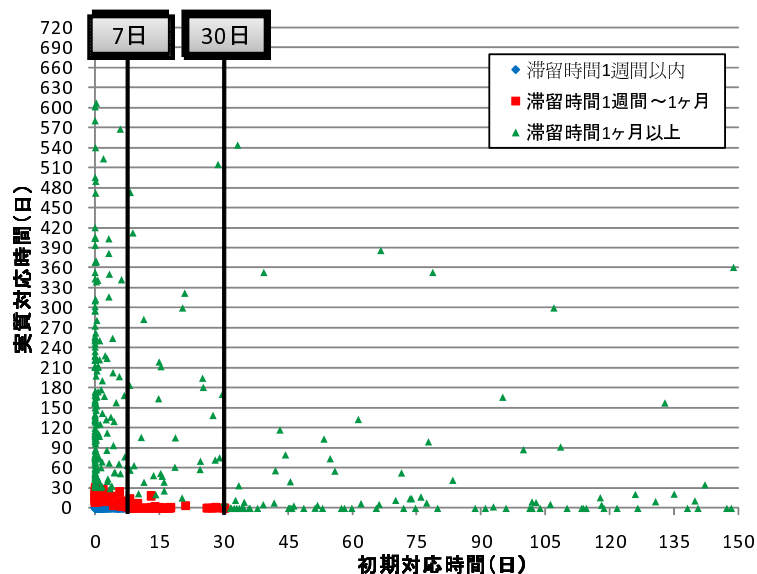


図 10 初期対応時間と実質対応時間の関係 (apache)

表 6 滞留時間が長いバグのうち初期対応時間別のバグの数 (apache)

		初期対応時間		
		0 日以上	1 週間以上	1 ヶ月以上
滞留時間	1 週間以上	471(63%)	212(28%)	132(18%)
	1 ヶ月以上	349(47%)	165(22%)	132(18%)

5. ケーススタディ 1: Apache HTTP Server Project の結果

5.1 滞留時間と初期対応時間の分析

各バグの初期対応時間と実質対応時間の関係を表す散布図を図 10 に示す。横軸を初期対応時間、縦軸を実質対応時間とする。グラフ中の縦の線は初期対応時間が 7 日と 30 日を示す。また、滞留時間が長いバグのうち初期対応時間別のバグの数を表 6 に示す。

図 10 から初期対応時間、実質対応時間が共に時間がかかっているバグよりも、

表 7 重要度別のバグ数 (apache)

重要度		バグ数 (割合)	バグ数 (割合)
高い	blocker	36(5%)	202(27%)
	critical	60(8%)	
	major	106(14%)	
低い	normal	409(55%)	545(73%)
	minor	132(18%)	
	normal	4(0.5%)	
全体		747(100%)	747(100%)

表 8 重要度別の初期対応時間が長いバグ数 (apache)

		重要度	初期対応時間		
			0 日以上	1 週間以上	1 ヶ月以上
滞留時間	1 週間以上	高い	143(100%)	49(34.3%)	23(16.1%)
		低い	328(100%)	163(49.7%)	109(33.2%)
	1 か月以上	高い	107(100%)	37(34.6%)	23(21.5%)
		低い	242(100%)	128(52.9%)	109(45.0%)

初期対応時間もしくは、実質対応時間のどちらかに時間がかかっているバグが多いことが分かった。

また、表 6 より、滞留時間が 1 ヶ月以上のバグ 349 件のうち、初期対応時間が 1 週間以上であるバグは約半数 (165 件) 存在することが分かった。

5.2 重要度，優先度の分析

- 重要度

本研究で分析対象としている Apache のバグ数を重要度別に表 7 に示す。また、重要度別に初期対応時間別のバグの数を表 8 に示す。

表 9 優先度別のバグ数 (apache)

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

本研究で分析対象としている Apache のバグのうち重要度が高い分類に入るのは 202 件 (27%), 重要度が低いバグは 545 件 (73%) であった.

滞留時間が 1 週間以上のバグにおいて, 重要度が高いバグは 143 件存在し, そのうち初期対応に 1 週間以上かかっているのは 49 件 (34.3%), 1 か月以上かかっているのは 23 件 (16.1%) である. 一方, 重要度が低いバグは 328 件存在し, そのうち初期対応に 1 週間以上かかっているのは 163 件 (49.7%), 1 か月以上かかっているのは 109 件 (33.2%) である.

さらに, 滞留時間が 1 か月以上のバグにおいて, 重要度が高いバグは 107 件存在し, そのうち初期対応に 1 週間以上かかっているのは 37 件 (34.6%), 1 か月以上かかっているのは 23 件 (21.5%) である. 一方, 重要度が低いバグは 242 件存在し, そのうち初期対応に 1 週間以上かかっているのは 128 件 (52.9%), 1 か月以上かかっているのは 109 件 (45.0%) である.

以上の結果から重要度が高いバグの方が初期対応時間が短い傾向にあることが分かる.

- 優先度

本研究で分析対象としている Apache のバグ数を優先度別に表 9 に示す. また, 優先度別に初期対応時間別のバグの数を表 10 に示す.

本研究で分析対象としている Apache のバグのうち優先度が高い分類に入る

表 10 優先度別の初期対応時間が長いバグ数 (apache)

		優先度	初期対応時間		
			0 日以上	1 週間以上	1 ヶ月以上
滞留時間	1 週間以上	高い	31(100%)	9(29.0%)	6(19.4%)
		低い	440(100%)	203(46.1%)	126(28.6%)
	1 か月以上	高い	22(100%)	6(27.3%)	6(27.3%)
		低い	327(100%)	159(48.6%)	126(38.5%)

のは 49 件 (6.6%), 優先度が低いバグは 698 件 (93.4%) であった。

滞留時間が 1 週間以上のバグにおいて, 優先度が高いバグは 31 件存在し, そのうち初期対応に 1 週間以上かかっているのは 9 件 (29.0%), 1 か月以上かかっているのは 6 件 (19.4%) である。一方, 優先度が低いバグは 440 件存在し, そのうち初期対応に 1 週間以上かかっているのは 203 件 (46.1%), 1 か月以上かかっているのは 126 件 (28.6%) である。

さらに, 滞留時間が 1 か月以上のバグにおいて, 優先度が高いバグは 22 件存在し, そのうち初期対応に 1 週間以上かかっているのは 6 件 (27.3%), 1 か月以上かかっているのは 6 件 (27.3%) である。一方, 優先度が低いバグは 327 件存在し, そのうち初期対応に 1 週間以上かかっているのは 159 件 (48.6%), 1 か月以上かかっているのは 126 件 (38.5%) である。

以上の結果から, 優先度が高いバグの方が初期対応時間が短い傾向にあることが分かる。

5.3 報告時期の時系列分析

バグ報告数と初期対応時間が 1 週間以上の未対応バグ数の推移を図 11 に示す。未対応バグ数を折れ線グラフ, バグ報告数を棒グラフで表現する。横軸を期間とし, 未対応バグ数を左側の縦軸, バグ報告数を右側の縦軸に示す。

2002 年 4 月に 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 件のバグが報告されている。また, 未対応バグ数は 2002 年 10 月頃に 110 件程度

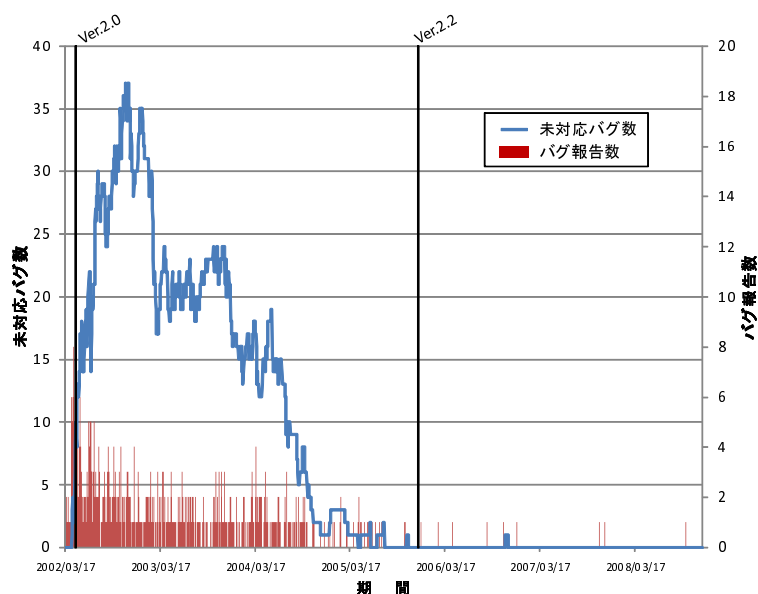


図 11 未対応バグとバグ報告数の時系列分析 (apache)

のバグを抱え、それから報告、修正が行われ、2005 年頃には未対応バグ数が 10 件程度まで修正されている。よって、2002 年 10 月頃までは修正されるバグ数よりも、バグを報告されるバグ数の方が多いことが分かる。また、2003 年 3 月中旬頃、2003 年末から 2004 年始めにかけて、多くのバグが一度に対応されていることが分かる。

さらに、未対応バグ数とバグ報告数の推移からバグ報告数が多い時期ほど、バグの初期対応時間は長くなる傾向にあることが分かる。

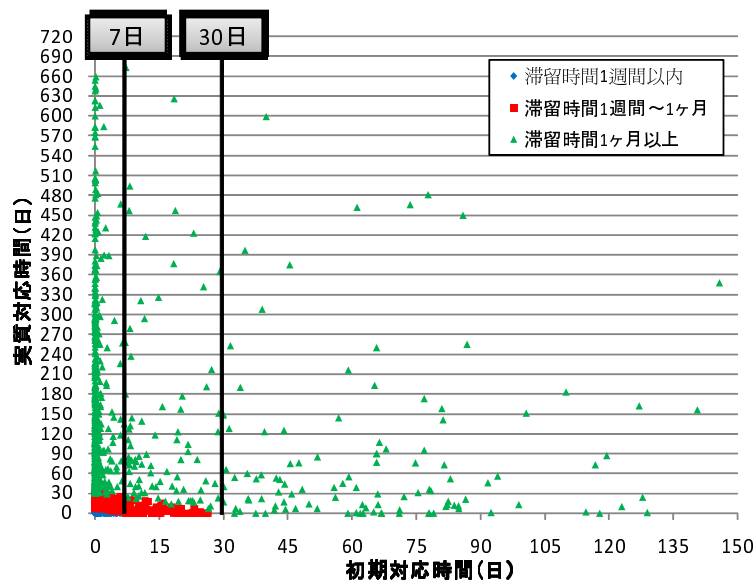


図 12 初期対応時間と実質対応時間の関係 (firefox)

表 11 滞留時間が長いバグのうち初期対応時間別のバグの数 (firefox)

		初期対応時間		
		0 日以上	1 週間以上	1 ヶ月以上
滞留時間	1 週間以上	1395(59%)	290(12%)	134(6%)
	1 ヶ月以上	884(37%)	214(9%)	134(6%)

6. ケーススタディ2:Mozilla Firefox Project の結果

6.1 滞留時間と初期対応時間の分析

各バグの初期対応時間と実質対応時間の関係を表す散布図を図 12 に示す。横軸を初期対応時間、縦軸を実質対応時間とする。グラフ中の縦の線は初期対応時間が7日と30日を示す。また、滞留時間が長いバグのうち初期対応時間別のバグの数を表 11 に示す。

図 12 から Apache に比べて初期対応時間が短いバグが多く見られ、初期対応時間に時間がかかっているにも関わらず、実質対応時間が短いというバグは少ない。

表 12 重要度別のバグ数 (firefox)

重要度		バグ数 (割合)	バグ数 (割合)
高い	blocker	67(3%)	445(19%)
	critical	109(5%)	
	major	269(11%)	
低い	normal	1652(70%)	1931(81%)
	minor	155(7%)	
	normal	124(5%)	
全体		2376(100%)	2376(100%)

表 13 重要度別の初期対応時間が長いバグ数 (firefox)

		重要度	初期対応時間		
			0 日以上	1 週間以上	1 ヶ月以上
滞留時間	1 週間以上	高い	211(100%)	17(8.1%)	5(2.4%)
		低い	1184(100%)	273(23.1%)	129(10.9%)
	1 か月以上	高い	108(100%)	8(7.4%)	5(4.6%)
		低い	776(100%)	206(26.5%)	129(16.6%)

また、表 11 より、滞留時間が 1 ヶ月以上のバグ 884 件のうち、初期対応時間が 1 週間以上というバグは約 24% の 214 件であることが分かる。

6.2 重要度，優先度の分析

- 重要度

本研究で分析対象としている Firefox のバグ数を重要度別に表 12 に示す。

また、重要度別に初期対応時間別のバグの数を表??に示す。

本研究で分析対象としている Firefox のバグのうち重要度が高い分類に入るのは 445 件 (19%)、重要度が低いバグは 1931 件 (81%) であった。

表 14 優先度別のバグ数 (firefox)

優先度		バグ数 (割合)	バグ数 (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(100.0%)	2376(100%)	

留時間が1週間以上のバグにおいて、が高いバグは211件存在し、そのうち初期対応に1週間以上かかっているのは17件(8.1%)、1か月以上かかっているのは5件(2.4%)である。一方、重要度が低いバグは1184件存在し、そのうち初期対応に1週間以上かかっているのは273件(23.1%)、1か月以上かかっているのは129件(10.9%)である。

さらに、滞留時間が1か月以上のバグにおいて、重要度が高いバグは108件存在し、そのうち初期対応に1週間以上かかっているのは8件(7.4%)、1か月以上かかっているのは5件(4.6%)である。一方、重要度が低いバグは776件存在し、そのうち初期対応に1週間以上かかっているのは206件(26.5%)、1か月以上かかっているのは129件(16.6%)である。

以上の結果から重要度が高いバグの方が初期対応時間が短い傾向にあることが分かる。

- 優先度

本研究で分析対象としている Firefox のバグ数を優先度別に表 14 に示す。また、優先度別に初期対応時間別のバグの数を表 15 に示す。

本研究で分析対象としている Firefox のバグのうち優先度が高い分類に入るのは535件(22.5%)、優先度が低いバグは174件(7.3%)であった。

滞留時間が1週間以上のバグにおいて、優先度が高いバグは349件存在し、

表 15 優先度別の初期対応時間が長いバグ数 (firefox)

		優先度	初期対応時間		
			0 日以上	1 週間以上	1 ヶ月以上
滞留時間	1 週間以上	高い	349(100%)	71(20.3%)	37(10.6%)
		低い	153(100%)	42(27.5%)	16(10.5%)
		－	893(100%)	177(19.8%)	81(9.1%)
	1 か月以上	高い	238(100%)	56(23.5%)	37(15.5%)
		低い	116(100%)	30(25.9%)	16(13.8%)
		－	530(100%)	128(24.2%)	81(15.3%)

そのうち初期対応に 1 週間以上かかっているのは 71 件 (20.3%), 1 か月以上かかっているのは 37 件 (10.6%) である。一方, 優先度が低いバグは 153 件存在し, そのうち初期対応に 1 週間以上かかっているのは 42 件 (27.5%), 1 か月以上かかっているのは 16 件 (10.5%) である。

さらに, 滞留時間が 1 か月以上のバグにおいて, 優先度が高いバグは 238 件存在し, そのうち初期対応に 1 週間以上かかっているのは 56 件 (23.5%), 1 か月以上かかっているのは 37 件 (15.5%) である。一方, 優先度が低いバグは 116 件存在し, そのうち初期対応に 1 週間以上かかっているのは 30 件 (25.9%), 1 か月以上かかっているのは 16 件 (13.8%) である。

以上の結果から, 優先度が高いバグの方が初期対応時間が短い傾向にあることが分かる。

6.3 報告時期の時系列分析

バグ報告数と初期対応時間が 1 週間以上のバグの未対応バグ数の推移を図 13 に示す。未対応バグ数を折れ線グラフ, バグ報告数を棒グラフで表現する。横軸を期間とし, 未対応バグ数を左側の縦軸, バグ報告数を右側の縦軸に示す。Firefox と Thunderbird の開発を分離された 2002 年中旬頃からバグ報告は行われ始め, 2005 年のバグ報告はやや少なかったが (173 件), 2003 年, 2004 年, 2006 年は年間

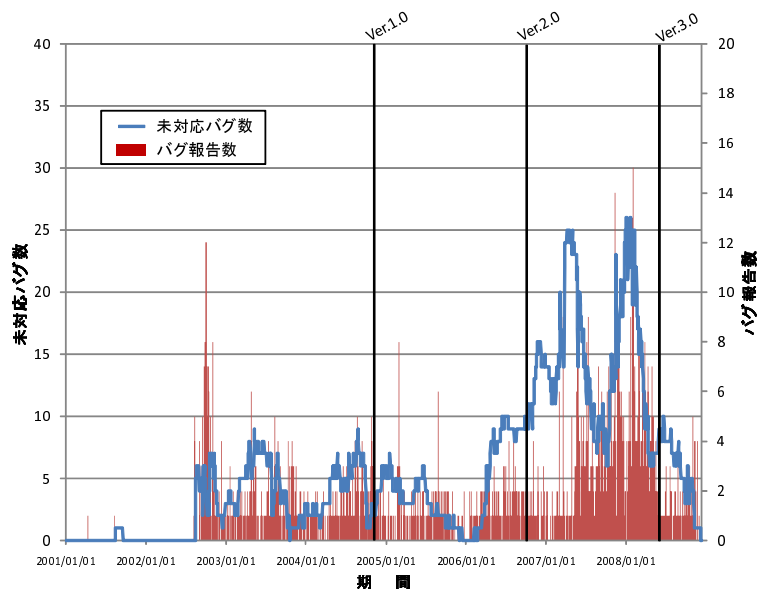


図 13 残留バグとバグ報告数の時系列分析 (firefox)

250 件前後のバグが報告されている。しかし、2006 年 10 月に ver.2.0 がリリースされて行こうバグ報告の数が増加し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バグ報告数が減少していることが分かった。また、残留バグ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残留バグは 50～100 件であったが、その後、残留バグ数は増加し、2008 年 1 月頃に 200 件程度のバグを抱えていた。その後、残留バグは減少していることが分かった。よって、2002 年中旬頃から 2006 年 10 月の ver.2.0 がリリースされるころまでは、報告と修正が繰り返され、残留バグ数が安定しているが、ver.2.0 がリリースされて以降、修正されるバグ数よりも報告数の方が多いことが分かる。

さらに、未対応バグ数とバグ報告数の推移からバグ報告数が多い時期ほど、バグの初期対応時間は長くなる傾向にあることが分かる。

7. 考察

7.1 滞留時間と初期対応時間の分析

Apache については滞留時間が1週間以上、1ヶ月以上のバグの中で、初期対応に1週間以上かかっているバグがそれぞれ約半数存在していることが分かった。また、初期対応に1か月以上かかっているバグは全体の18%存在していることが分かった。一方、Firefox については Apache に比べて初期対応が早いバグの割合が多く、滞留時間が1週間以上、1ヶ月以上のバグの中で、初期対応に1週間以上かかっているバグはそれぞれ約20%存在していることが分かった。また、初期対応に1ヶ月以上のバグは全体の約6%存在していることが分かった。

これらの結果から Apache よりも Firefox の方が初期対応が早いことが分かる。これは Apache に比べ Firefox の方がバグ報告後、掲示板に書き込む時間が早く、Firefox の方が掲示板を利用する意識が高いことが考えられる。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を明らかにするために、各バグに対するコメント数の分布を分析する。

Apache, Firefox において各バグに投稿されるコメント数の分布をそれぞれ図14, 図15にヒストグラムで示す。横軸はコメント数とし、区間をコメント1回とする。縦軸はバグ数とする。また、Apache, Firefox のコメント数に関する統計量を表16に示す。

Apache, Firefox のコメント数の平均からも Firefox の方がコメント数が多いことが分かる。また、各バグに対するコメントの最頻値が 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。よって、Apache よりも Firefox の開発者よりも掲示板を利用する意識が高いことが考えられる。

7.2 重要度分析

バグの多くは重要度が Normal に設定されており、Apache, Firefox 共に重要度が高いバグは低い割合になっていることが分かった。また、滞留時間、初期対応時間が長いバグに関しても、重要度が低いバグ数よりも高いバグ数の方が少ない

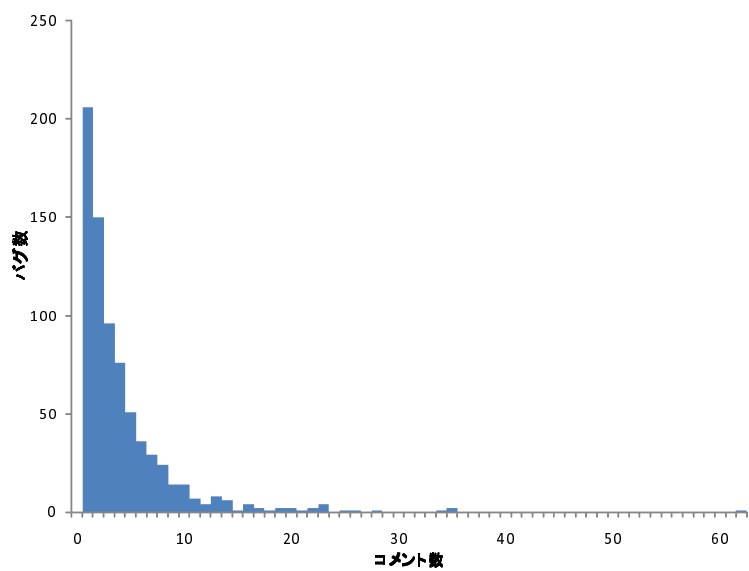


図 14 コメント数のヒストグラム (apache)

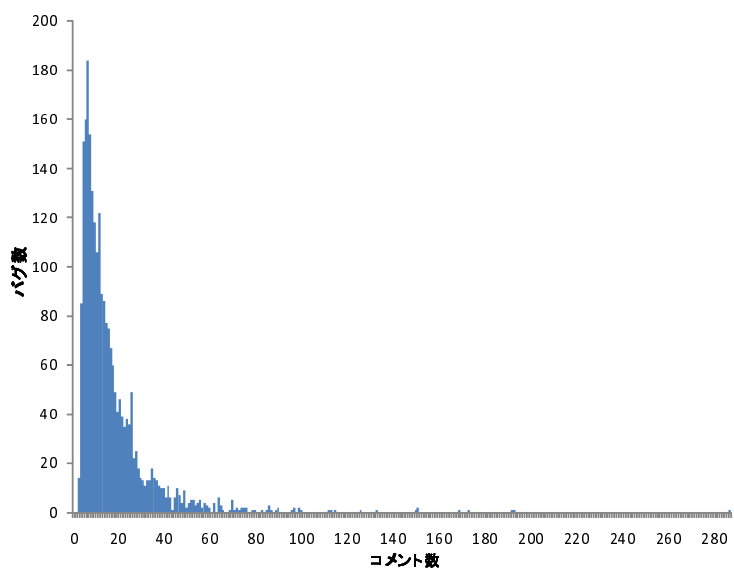


図 15 コメント数のヒストグラム (firefox)

表 16 コメント数の統計量

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

ことが分かった。しかし、滞留時間、初期対応時間が長いバグにも重要度が高いバグが存在している。バグ報告者は多くのバグを重要度 Normal に設定されているにも関わらず、Normal よりも高い重要度に設定しているということは早急に直すべきバグであると考えられる。しかし、重要度が高いが次回のリリースまでに修正すべき内容であることから滞留時間が延び、初期対応時間が延びた可能性もある。そこで、滞留時間、初期対応時間が1か月以上で重要度が高いバグの内容を分析する。

7.3 報告時期の時系列分析

8. 関連研究

8.1 OSS における情報共有についての研究

OSS では掲示板、ML などを利用して情報共有を行っている

8.2 Bugzilla で行われるコメントについての研究

□

8.3 滞留時間についての研究

Empirical Analysis of the Bug Fixing in Open Source Projects@OSS2008[]
デバッグの活動を継続性・効率性に着目して分析している
デバッグプロセスにはいくつか（非）継続性・（非）効率性で4つに分類できる
本研究との違いはCLOSEDで修正終了とするか、FIXEDで修正終了とするかの違いである
CLOSEDだとFIXされてからレビューが入るから分類される,

8.4 Bugzillaの優先度や重要度による滞留時間の違いについての研究

Towards a Simplification of the Bug Report from in Eclipse@MSR'08[]
レポーターが決定する重要度が滞留時間にどのような影響を及ぼしているかを分析している。本研究との違いは重要度によって初期対応時間に影響しているかまで考慮されていない

9. 今後の展望

地理的な違いがあるから今後考慮する必要がある
(Distance, Dependencies, and Delay in a Global Collaboration@CSCW2000)
(An Empirical Study of Global Software Development: Distance and Speed)

謝辭

Thank you. Thank you.

これはおまけの図です。

図 16 おまけの図

付録

A. おまけその1

これはおまけです。これはおまけです。これはおまけです。これはおまけです。
これはおまけです。これはおまけです。これはおまけです。これはおまけです。
これはおまけです。これはおまけです。これはおまけです。これはおまけです。
これはおまけです。これはおまけです。これはおまけです。これはおまけです。

B. おまけその2

これもおまけです。これもおまけです。これもおまけです。これもおまけです。
これもおまけです。これもおまけです。これもおまけです。これもおまけです。
これもおまけです。これもおまけです。これもおまけです。これもおまけです。
これもおまけです。これもおまけです。これもおまけです。これもおまけです。

参考文献

- [1] 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, May 2007.
- [2] Wang Y., Guo D. and Shi H. Measuring the evolution of open source software systems with their communities. SIGSOFT Softw. Eng. Notes 32, 6 (Nov. 2007)

- [3] 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(MSR '08). 145-148.