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　　大平 雅雄　　松本 健一
奈良先端科学技術大学院大学　情報科学研究科
E-mail: {akinori-i,masao,matumoto}@is.naist.jp

概要

多くのオープンソースソフトウェア（*OSS*）開発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．しかし，障害の修正が効率的に行われていないことが多い．特に，障害が報告されてから修正に取りかかるまでの時間（未対応時間）が長いことが問題となっている．本研究の目的は，効率的な障害修正のための作業支援システムを構築することである．本稿ではその第一歩として，未対応時間の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分析を行う．分析の結果，(*i*) 重要度と優先度の指定方法が未対応時間に影響していること，*(ii)* 短期間に障害報告数が集中すると未対応時間が長くなることが分かった．

1　はじめに

OSS の大規模化や複雑化に伴い，障害の数が増加し，全ての障害について修正状況を把握することが困難になるという問題が生じている．そのため，多くの OSS 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 [1][3].

近年，OSS の信頼性向上のために滞留時間の短縮化が望まれており，障害管理システムを利用する OSS プロジェクトを対象に，障害管理の状況を把握するための一手段として，滞留時間の分析が盛んに行われている [5][6][9].

ユーザのために障害を素早く修正することが重要であるが，修正に長い時間を要していることが明らかにされている [2][7]．さらに，Wang らは Linux ディストリビューションの一つである Ubuntu を分析対象とし，障害の修正状況を分析した結果，多くの障害は修正する担当者を割り当てることさえ行われていないことが分かった [10]．つまり，障害の滞留時間のうち未対応時間が長い時間を占めていると考えられる．

本研究の目的は，効率的な障害修正のための作業支援システムを構築することである．本稿ではその第一歩として，未対応時間長い障害はどのような障害なのか明らかにするために分析を行う．そのために Apache を対象に滞留時間を障害が報告されてから修正が完了するまでの時間（未対応時間）と障害の修正を行っている時間（修正時間）の 2 つに分類し，障害の優先度及び重要度別に未対応時間の長さの違いを分析する．また，障害報告数による未対応時間の長い障害数の違いを分析する．

以降，2 章では OSS における障害管理について説明を行い，その後本稿で扱う障害の滞留時間についての定義を行う．3 章ではケーススタディで行う分析の方法について説明し，4 章で Apache を対象としたケーススタディを行い結果を報告する．5 章でケーススタディの考察を行い，6 章で関連研究について述べ本研究の立場を明らかにし，7 章で本稿のまとめと今後の課題を述べる．

2　準備

本章では障害管理システムを利用している OSS プロジェクトを対象にケーススタディを行う際，扱う用語の説明を行う．

2.1　OSS 開発における障害管理

多くの OSS プロジェクトでは障害管理を行うために障害管理システムを利用している [1][3]．障害管理システムはソフトウェア開発中に発生する障害をデータベー

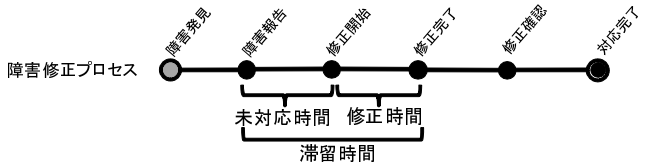

図 1. 滞留時間，未対応時間，修正時間の概略図

スなどに登録し，修正状況などを一元管理するシステムである．主な機能としては，障害報告時に重要度や優先度等を指定する機能があり，早急に直すべき障害の検索が可能である [8]．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) を行うことが可能である．掲示板は各障害が報告された後，いつでも投稿可能であり，障害の進捗状況の把握や修正活動のために利用されている．そのため，掲示板に記載されているコメントは障害の修正履歴の一つとみなすことができ，本稿では掲示板に 1 回目のコメントが投稿された時刻を障害修正の開始時刻と考える．

2.2 滞留時間，未対応時間，修正時間の定義

本節では滞留時間と未対応時間と修正時間の定義を行う．定義を行う前に本稿で取得する時刻についての説明を行う．障害修正プロセスにおける滞留時間と未対応時間と修正時間を図 1 に示す．本稿では障害が報告されてから修正が完了するまでの時間（未対応時間）と障害の修正を行っている時間（対応時間）を求めるために 3 つの時刻を取得する．まず障害管理システムに障害情報を登録した時刻（障害報告），掲示板に 1 回目のコメントが投稿された時刻（修正開始），障害の修正が完了した時刻（修正完了）の 3 つの時刻を取得する．3 つの時刻を用いて，滞留時間，未対応時間，修正時間の 3 つの時間を定義する．

滞留時間: 障害報告を行った時刻から，修正完了の時刻までとする．

未対応時間: 障害報告を行った時刻から，修正開始の時刻までとする．

修正時間: 修正開始の時刻から，修正完了の時刻までとする．

3 分析方法

本章では未対応時間にどれくらいの時間を要しているか，また，未対応時間の長い障害がどのような障害なのか明らかにするための 3 つの分析について分析方法を説明する．

3.1 未対応時間と滞留時間

Wang らの研究では障害の滞留時間において未対応時間に長い時間を要していることが分かった [10]．本稿では障害の滞留時間における未対応時間と修正時間の関係を分析する．滞留時間が長い障害は，未対応時間と修正時間の両方が長いのか，未対応時間のみが長いのか，修正時間のみが長いのかを明らかにする．

3.2 重要度，優先度による未対応時間

重要度と優先度はそれぞれ障害の深刻度と修正するための優先順位を意味することから，重要度や優先度により修正に取りかかるまでの未対応時間が異なると考えられる．本稿では重要度/優先度が高い障害と低い障害のどちらが早く障害の修正に向けた対応が行われるのかを明らかにするためにそれぞれの未対応時間を分析する．

重要度別，優先度別の未対応時間は図 2 のように示す．横軸は未対応時間，縦軸は修正を開始し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) 障害の累積とする．一方，曲線 B は重要度 (または優先度) が低い (もしくは高い) 障害の累積とする．図 2 の場合，曲線 A は曲線 A に該当する障害のうち 4 日以内に修正を開始した障害が 50%存在することを意味し，曲線 B は曲線 B に該当する障害のうち 10 日以内に修正を開始した障害が 50%存在することを意味する．つまり，曲線 B よりも曲線 A の方が障害の修正開始が早いことを示す．

重要度は 7 種類 (Blocker，Critical，Major，Normal，Minor，Trivial，Enhancement) に分類されており，機能拡張の際に指定される Enhancement を除く障害を対象として，Blocker と Critical と Major を重要度の高い障害，Normal と Minor と Trivial を重要度の低い障害とする．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．多くの

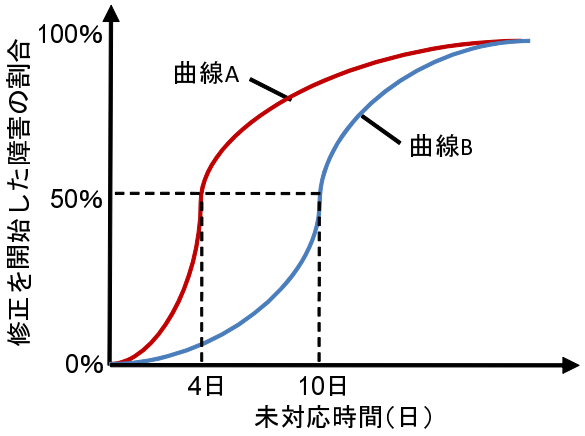

図 2. 重要度，優先度別の修正を開始した障害の割合を示す概略図

OSS プロジェクトでは優先度が P3 と指定される障害が最も多いことから，P1 や P2 の障害は相対的に見て優先的に直すべき障害と見なされていると考えられる．本稿では P1 と P2 を優先度の高い障害，P3〜P5 の障害を優先度の低い障害とする．

3.3　障害報告数と未対応障害数

OSS はソースコードが公開されており，リリース直後は障害報告が増加することが多い．障害報告が増加すると，開発者が全ての障害の修正を行えなくなり，未対応時間が長くなると考えられる．本稿では障害報告数は未対応時間が長くなる要因の一つであることを確認するために，障害報告数と未対応時間の長い障害数の推移を分析する．ただし，未対応時間の長い障害を未対応時間が 1 か月以上の障害をとみなす．

4　ケーススタディ

本章では，3 章で述べた分析方法を用いて Apache プロジェクトを対象としてケーススタディを行う．Apache の分析対象データの概要を表 1 に示す．分析対象とする Apache のバージョンは ver.1.3 系列，ver.2.0 系列，ver.2.2 系列とする．

表 1. 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

4.1　Apache プロジェクトにおける障害管理

Apache プロジェクトでは障害管理システムの Bugzilla を 2003 年 3 月頃から導入している．Apache プロジェクトでは障害を発見した際，障害の発見者が Bugzilla に障害に関する情報を投稿する前に，障害の情報をユーザ用のメーリングリストに投稿する．ユーザ用のメーリングリストで数日間返信がなかった場合，もしくは解決されなかった場合に Bugzilla に投稿するという報告プロセスが定められている．そして，Bugzilla に障害の情報を投稿後は，随時，開発者が障害の修正を行う．しかし，報告直後に修正されない障害が存在するため，開発者が不定期に修正されていない障害を検索し，修正作業を行っている．

4.2　分析結果

4.2.1　滞留時間，未対応時間，修正時間の統計量

滞留時間，未対応時間，修正時間の各統計量を表 2 に示す．また，滞留時間毎の障害数の分布と未対応時間毎の障害数の分布を図 3 に示す．左のヒストグラムが滞留時間に関するグラフで，横軸を滞留時間とする．右のヒストグラムが未対応時間に関するグラフで横軸を未対応時間とする．ヒストグラムのデータ区間は 1 日である．未対応時間が非常に長い (90 日以上) の障害については 1 つの階級にまとめている．滞留時間の中央値は 24.13 日であることから障害の多くは 1 か月以内に修正が完了し，未対応時間の中央値が 0.81 日であることから障害の多くは 1 日以内に修正を開始していることが分かる．また，未対応時間に関するグラフから 3 か月以上修正が開始されていない障害は 77 件存在し，中には 2 年以上修正が開始されていない障害が存在していることが分かった．

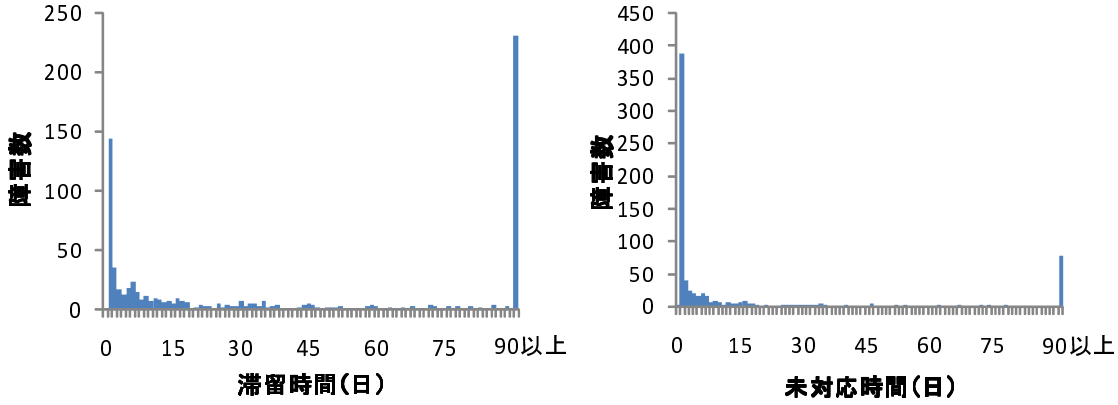

図 3. 滞留時間毎と未対応時間毎の障害数の分布

表 2. Apache における障害の滞留時間

	滞留時間	未対応時間	修正時間
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

4.2.2 未対応時間と修正時間

障害の未対応時間と修正時間の関係を図 4 に示す．横軸を未対応時間，縦軸を修正時間とする．未対応時間，修正時間ともに 900 日以内の分布を示す．未対応時間が 1 日以内で，かつ修正時間に長い時間を要する障害と，修正時間が 1 日以内で，かつ未対応時間の長い時間を要する障害に偏っていることが分かった．つまり，修正時間が 1 日以内で，かつ未対応時間に長い時間を要している障害に関しては修正活動を開始するまでに長い時間を要していることが分かる．

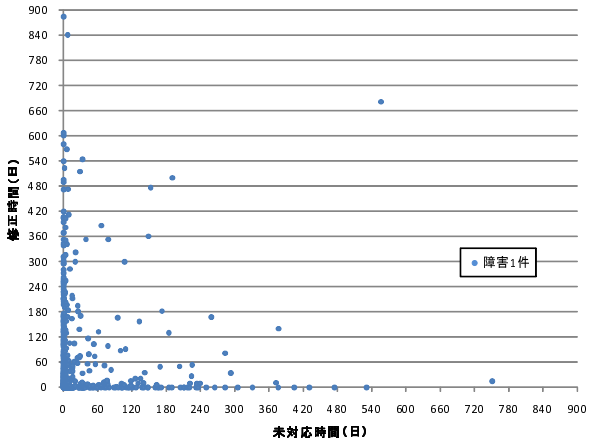

図 4. 障害の未対応時間と修正時間の関係

4.2.3 重要度，優先度による未対応時間

- 重要度

  重要度別の障害数を表 3 に示す．また，重要度別の修正を開始した障害数の累積グラフを図 5 に示す．横軸を未対応時間，縦軸を修正を開始した障害の割合とする．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 日である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%，重要度が低い障害のうち約 50%は報告されてから 1 日以内に修正を開始していることが分かった．また，重要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．つまり，重要度が高い障害ほど優先的に修正が行われ，低い障害ほど修正の開始が遅いことが分かった．

- 優先度

  優先度別の障害数を表 4 に示す．また，優先度別の修正を開始した障害数の累積グラフを図 6 に示す．横軸を未対応時間，縦軸を修正を開始した障害の割合とする．ただし，重要度の分析と同様に，半

表 3. 重要度別の障害数

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%)	
	trivial	4(0.5%)	
全体		747(100%)	747(100%)

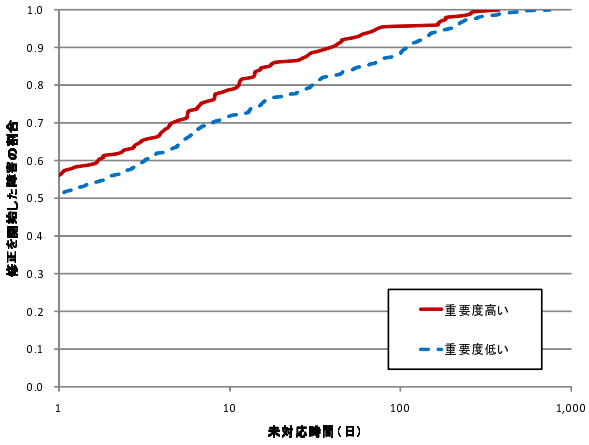

図 5. 重要度別の修正を開始した障害数の累積グラフ

表 4. 優先度別の障害数

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%)	698(93.4%)
	P4	1(0.1%)	
	P5	7(0.9%)	
全体		747(100%)	747(100%)

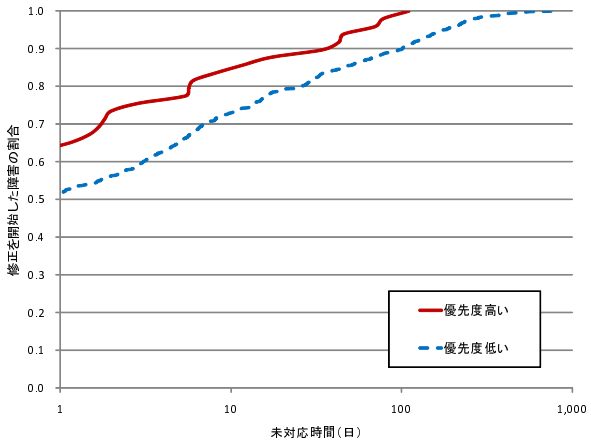

図 6. 優先度別の修正を開始した障害数の累積グラフ

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 日である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%，優先度の低い障害のうち約 51%は報告されてから 1 日以内に修正を開始していることが分かった．また，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．つまり優先度が高いほど優先的に修正が行われ，低い障害ほど修正の開始が遅いことが分かった．

分析対象データの中で重要度，優先度の高い障害は低い障害に比べて修正開始が早いことが分かった．しかし，重要度や優先度が高いにも関わらず，修正開始が遅れている障害も存在した．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．

- 修正方法の指摘
- 類似する障害，もしくは関連する障害を発見し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ていたが，他の開発者がパッチの作成を行わなかったため，報告者が修正パッチを作ったケースがあった．その他に，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ったケースが存在した．

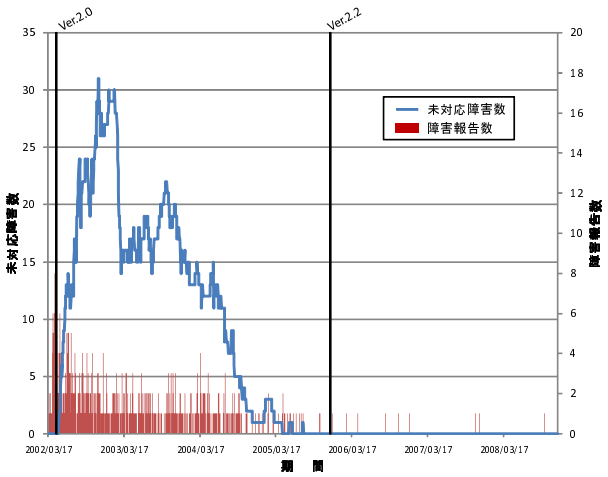

図 7. 障害報告数と未対応時間が 1 か月以上の障害の推移

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 7 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．グラフ中の縦線はリリース時期を示す．2002 年 4 月に ver.2.0 がリリースされた時期から 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．一方，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．

つまり，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．さらに，障害報告数が最も多い時期と未対応障害数の最も多い時期は必ずしも一致しているとは言えないことが分かった．

5 考察

5.1 障害の未対応時間と滞留時間

図 4 より修正時間が 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．このような障害は早急に修正が開始されていれば障害の滞留時間が長くなることを避けることができた可能性がある．

また，Apache では開発者が修正されていない障害を不定期に検索し，修正作業を行っている．定期的に修正作業を行うことで，未対応時間の長い障害を減少することができたと考える．

5.2 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から，開発者は障害管理システムの意図に則って，重要度や優先度を参考にどの障害を優先的に直すべきか選んでいることが分かる．そのため，重要度や優先度が未対応時間を左右する要因の一つであることが考えられる．つまり，適切でない重要度や優先度を指定してしまうと修正が遅れてしまうことが考えられることから，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必要がある．滞留時間短縮のためには，未対応時間の長い障害の優先度を上げることで，修正開始が早くなると考えられる．

さらに，重要度や優先度が高いにも関わらず未対応時間が長い障害を詳しく分析した結果，報告は確認していたが修正を後回しにしたのか，それとも報告に気付かず修正が遅れたのか判断することはできなかった．しかし，未対応時間の長い障害の多くは開発者からの返答が無く未対応時間が長くなったケースであったことが分かった．障害報告直後に障害に対する指摘などのコメントを送っていれば，早く修正が開始され，滞留時間の長さが変わる可能性があると考える．

5.3 障害報告数と未対応障害数

障害報告数が最も多い時期と未対応障害数の最も多い時期は必ずしも一致しているとは言えないことが分かった．しかし，2002 年 3 月頃，障害報告数が最も多く，その後，未対応障害数が増加しているため，障害報告数の増加により開発者の修正が間に合わなくなり，修正開始できない障害が増加したと考えられる．つまり，障害報告数は未対応時間を左右する一つの要因として考えられる．

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,「報告された障害の内，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，これまで修正を開始していなかった障害の約 50%が修正を開始している．これは障害管理システムに登録された障害のうち，修正を開始していない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だけでなく，定期的に修正されていない障害を検索して修正を開始していない障害の対処を行う必要がある．

6 関連研究

6.1 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いる [4][10]．OSS の進化を計測するための指標として，Wang らは OSS の進化を計測するための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，修正を開始する以前の段階で修正が進んでいないということは明らかにされたが，修正が開始し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に長い時間を要しているかについても明らかにされていなかった．本稿では未対応時間の長い重要度や優先度，また障害が未対応時間の長い障害が報告された時期の報告数を分析を行い，未対応時間の長い障害がどのような障害なのか明らかにした．

6.2 OSS 開発における障害の滞留時間

OSS 開発における障害の滞留時間についての研究として Mockus らの研究 [9]，Herraiz らの研究 [5] がある．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 [9]．ユーザにとって障害が素早く修正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．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．

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，重要度，優先度ともに簡略化が可能であることが分かった．

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，本稿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考え，障害の未対応時間に着目し分析を行った．分析の結果，滞留時間の中で未対応時間が長い時間を占めている障害が多く存在していることが分かった．

7 まとめ

本論文では障害の滞留時間を短縮する手段を提案するための第一歩として，未対応時間が長い障害がどのような障害なのか明らかにすることを目的として分析を行った．Apache プロジェクトを対象に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，修正開始までに長い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．

- 重要度や優先度を低く指定すると未対応時間が長くなることが分かった．
- 障害報告が増加傾向にある時に障害報告すると修正開始が遅れる可能性があることが分かった．

本稿で得られた知見を用いることで，障害管理システムを用いる 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 OSS をケーススタディとして分析する必要がある．

謝辞

本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また，本研究の一部は，文部科学省科学研究補助費（若手 B：課題番号 20700028）による助成を受けた．

参考文献

[1] Canfora, G. and Cerulo, L.: Impact Analysis by Mining Software and Change Request Repositories, *In Proceedings of the 11th IEEE International Software Metrics Symposium*, IEEE Computer Society, p. 29 (2005).

[2] Chou, A., Yang, J., Chelf, B., Hallem, S. and Engler, D.: An empirical study of operating systems errors, *In Proceedings of the eighteenth ACM symposium on Operating systems principles*, ACM, pp. 73–88 (2001).

[3] Fischer, M., Pinzger, M. and Gall, H.: Analyzing and Relating Bug Report Data for Feature Tracking, *In Proceedings of the 10th Working Conference on Reverse Engineering*, IEEE Computer Society, p. 90 (2003).

[4] Godfrey, M. W. and Tu, Q.: Evolution in Open Source Software: A Case Study, *In Proceedings of the International Conference on Software Maintenance*, pp. 131–142 (2000).

[5] Herraiz, I., German, D. M., Gonzalez-Barahona, J. M. and Robles, G.: Towards a simplification of the bug report form in eclipse, *In Proceedings of the 2008 international working conference on Mining software repositories*, pp. 145–148 (2008).

[6] Kim, S. and Whitehead, Jr., E. J.: How long did it take to fix bugs?, *In Proceedings of the 2006 international workshop on Mining software repositories*, ACM, pp. 173–174 (2006).

[7] Michail, A. and Xie, T.: Helping users avoid bugs in GUI applications, *In Proceedings of the 27th international conference on Software engineering*, ACM, pp. 107–116 (2005).

[8] Michlmayr, M. and Hill, B. M.: Quality and the Reliance on Individuals in Free Software Projects, *In Proceedings of the 3rd Workshop on Open Source Software Engineering*, pp. 105–109 (2005).

[9] 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).

[10] Wang, Y., Guo, D. and Shi, H.: Measuring the evolution of open source software systems with their communities, *SIGSOFT Softw. Eng. Notes*, Vol. 32, No. 6, p. 7 (2007).