データサイズを考慮したコミッター候補者予測モデルの検討

松本 明[1,a)]　伊原 彰紀[2,b)]　大平 雅雄[1,c)]

概要：オープンソースソフトウェア（OSS）プロジェクトには，プロダクトに加えた変更をバージョン管理システムにコミットする権限を持つ，コミッターが存在する．近年，このコミッターに適任である候補者を見つけ出すことを目的として，既存コミッターおよび一般開発者の過去の活動履歴データを分析し，コミッター候補者予測モデルを構築する研究がなされている．既存の予測モデルでは，現コミッターがコミッターに昇格する前の活動（例えば，パッチ投稿，パッチ検証，コメント投稿など）に基づいてコミッター候補者予測モデルを構築する．しかしながら，一般開発者がコミッターに昇格するまでの活動は常に一定であるとは限らない．一般開発者の全ての活動期間における活動量の中央値を用いた予測モデルでは，コミッター候補者が有する本来の能力を過小評価する可能性がある．そのため，予測モデルの予測精度を向上させるためには，コミッター候補者が最も活発に活動を行った時期に絞ったデータ選択を行うことが有効である可能性がある．本論文では，既存の予測モデルをベースとし，データサイズを考慮したコミッター候補者予測モデルについて検討を行う．

1. 研究の背景と目的

オープンソースソフトウェア (OSS) プロジェクトでは，プロダクトに加えた変更をバージョン管理システムにコミットする権限を持つ，コミッターと呼ばれる開発者が存在する [1]．OSS プロジェクトにおける技術的活動や社会的活動の履歴から，十分な能力があると認められた開発者がコミッターとして選出される．このコミット権限付与方式は多くのプロジェクトで採用されており，不具合を含むコードファイルがプロダクトに混入する機会を抑制するという品質保証上の利点がある．

その一方で，開発者から投稿される不具合修正パッチの検証作業を少数のコミッターが中心となって負担するという作業負担上の課題が発生する場合がある．OSS の社会的普及により，プロジェクトへの不具合報告の件数は年々増加している．特に大規模プロジェクトではコミッターにかかる負担は極めて大きい [2], [3]．一日当たり数百件の不具合が報告される Eclipse プロジェクトや Mozilla プロジェクトでは，コミッターへの過度な負担が原因となり，不具合修正が完了するまでの時間の長期化を引き起こしていることが報告されている [4], [5], [6].

[1] 和歌山大学
Wakayama University
[2] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] s151045@center.wakayama-u.ac.jp
[b)] akinori-i@is.naist.jp
[c)] masao@sys.wakayama-u.ac.jp

コミッターの負担を軽減させるためには，新規コミッターの数を増やすという手段が効果的であると考えられる．しかしながら，プロジェクトに参加している一般開発者の中から，コミッターに昇格すべき有能な開発者（コミッター候補者）を見つけ出すことは容易ではない．一般的に多くの OSS プロジェクトでは，コミッター候補者は，既存のコミッターからの推薦と承認を経て昇格する [7].

コミッター推薦のために，既存コミッターは，プロジェクトに対するコミッター候補者の過去の活動内容，具体的には機能拡張や不具合修正に関連する活動内容を総合的に評価している．この活動実績評価の必要性から，一般開発者がコミッター候補者として推薦されるには，通常 1 年以上の継続的な活動が必要であるとされている．そのため，有能な開発者であっても，継続的に活動する意欲を失ってしまい，コミッターに推薦される前にプロジェクトから離脱してしまうことが少なくない [8].

この問題を解決するためには，コミッターに昇格すべき有能な開発者を可能な限り早い段階で見つけ出し，コミッター候補者として推薦を行う必要がある．このコミッター候補者を見つけ出すことを目的として，コミッター候補者予測モデルを構築する研究がなされている [9].

既存の予測モデルでは，現コミッターがコミッターに昇格する前の活動（例えば，パッチ投稿，パッチ検証，コメント投稿など）に基づいてコミッター候補者予測モデルを構築する．コミッター昇格前の全ての期間の各活動（一般開発者として初めて活動を開始してから，コミッターに昇

格するまでの全ての活動）の平均値（中央値）を用いるのが一般的である．

しかしながら，一般開発者がコミッターに昇格するまでの活動は常に一定であるとは限らない．例えば，プロジェクト参加初期には積極的にパッチ投稿を行っていても，必ずしもコミッター昇格直前にも同様にパッチ投稿を行っているとは限らない．一般的に，一般開発者がコミッターに昇格されるまでには，約 1 年以上の期間を要するため，一般開発者の全ての活動期間における活動量の中央値を用いた予測モデルでは，コミッター候補者として適任の開発者を早期に発見する事は困難である．

また同時に，長期間の活動量の中央値は，コミッター候補者が有する本来の能力を過小評価する可能性がある．そのため，コミッター候補者予測モデルの予測精度を向上させるためには，コミッター候補者が最も活発に活動を行った時期に絞ったデータ選択を行うことが有効である可能性がある．

本論文では，既存の予測モデルをベースとし，データサイズを考慮したコミッター候補者予測モデルについて検討を行う．

以降 2 章では，関連研究について述べ，本論文の立場を明らかにする．3 章では，本論文で提案する手法について詳しく述べる．4 章では，実験に用いるデータセットと実験の手順について述べる．5 章では，実験結果と結果の考察について述べる．最後に 6 章で，本論文のまとめと今後の課題について述べる．

2. 関連研究

2.1 コミッター昇格プロセスに関する研究

OSS の社会的普及により，開発者への修正依頼，各パッチの品質検証，構成管理，ユーザサポート，ドキュメントの作成など，様々な活動を行うコミッターの役割の重要性は年々高まっている．このような背景から近年，コミッター昇格プロセスに関して調査する研究が数多く行われている．

Jensen ら [7] は，OSS 開発者にインタビューを行い，既存コミッターがコミッター候補者をどのように決定しているのかを調査している．Apache プロジェクトでは，開発者の技術的な活動 (パッチの投稿など) を参考に，プロジェクトに多く貢献している開発者をコミッター候補者として，既存コミッターがプロジェクト管理委員 (PMC Member) に推薦する．その後，PMC Member がコミッター候補者の貢献を認めるとコミッターに昇格することができる．Mozilla プロジェクトも同様に，既存コミッターは開発者の技術的な活動に加えて，開発者が行う社会的な活動 (開発の指示など) を参考に，コミッター候補者を見つけ出している．

Bird ら [8] は PostgreSQL プロジェクトにおける開発者の活動期間に関する分析を通して，約 1 年間の活動実績がある一般開発者はコミッターに昇格する開発者として適切であると結論づけている．開発者の活動期間が 1 年よりも短い場合，コミッターとして適切であるかどうかを判断するのは難しく，活動期間が 1 年より長い場合，開発者は継続的な活動に取り組む意欲を失ってしまうリスクが高いことを挙げている．いずれの OSS プロジェクトでも，新たなコミッターを見つけ出すために，開発者の活動量や活動期間を参考にしているが，どの程度の活動を行っている開発者をコミッターとして推薦するのかを示す目安や指針は提示されていない．

現状では，プロジェクト管理者や既存コミッターは，客観的なデータではなく，彼ら自身の経験に基づいて開発者の活動量と活動期間を評価し，コミッターに昇格する開発者を決定せざるを得ない．そこで，藤田ら [10] は，パッチの投稿・検証回数とパッチの編集・検証時間に関して，PostgreSQL プロジェクトに参加しているコミッター候補者と一般開発者の違いを分析している．その結果，パッチの投稿回数とパッチの検証回数については，コミッター候補者と一般開発者で違いがあることが分かった．しかし，これらの活動量を用いて，どの程度コミッター候補者を見つけ出すことができるかについては定かではない．

2.2 コミッター候補者予測に関する研究

伊原ら [9] は，Eclipse platform プロジェクト，Mozilla Firefox プロジェクトにおいて，それぞれコミッター候補者と一般開発者の活動量と活動期間を分析し，活動履歴を用いて構築したコミッター候補者予測モデルによってどの程度の精度で予測することができるかを調査している．その結果，Eclipse platform プロジェクトでは再現率が約 70%，Mozilla Firefox プロジェクトでは約 80%の精度でコミッターに昇格する開発者を予測することができることが分かった．

既存の予測モデルでは，現コミッターがコミッターに昇格する前の活動（例えば，パッチ投稿，パッチ検証，コメント投稿など）に基づいてコミッター候補者予測モデルを構築する．コミッター昇格前の全ての期間の各活動（一般開発者として初めて活動を開始してから，コミッターに昇格するまでの全ての活動）の平均値（中央値）を用いるのが一般的である．

しかしながら，一般開発者がコミッターに昇格するまでの活動は常に一定であるとは限らない．例えば，プロジェクト参加初期には積極的にパッチ投稿を行っていても，必ずしもコミッター昇格直前にも同様にパッチ投稿を行っているとは限らない．一般的に，一般開発者がコミッターに昇格されるまでには，約 1 年以上の期間を要するため，一般開発者の全ての活動期間における活動量の中央値を用いた予測モデルでは，コミッター候補者として適任の開発者を早期に発見する事は困難である．

また同時に，長期間の活動量の中央値は，コミッター候補者が有する本来の能力を過小評価する可能性がある．そのため，コミッター候補者予測モデルの予測精度を向上させるためには，コミッター候補者が最も活発に活動を行った時期に絞ったデータ選択を行うことが有効である可能性がある．

本論文では，既存の予測モデルをベースとし，データサイズを考慮したコミッター候補者予測モデルについて検討を行う．

3. 提案手法

本論文では，先行研究で伊原らが提案したコミッター候補者予測モデル構築手法をベースとし，データサイズを考慮した予測モデル構築手法を提案する．本章では，先行研究における伊原らの提案手法の概要と本論文の提案手法について述べる．

3.1 先行研究における予測モデル構築手法

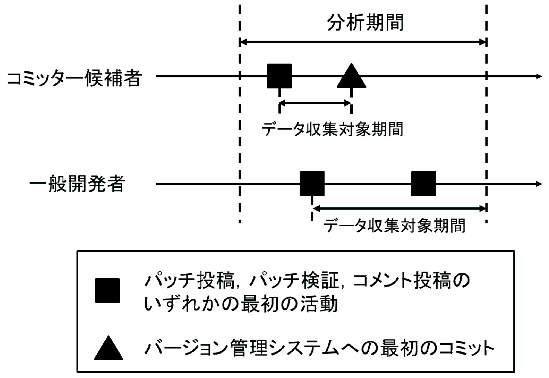

図 1 各分類の開発者の活動例

先行研究では，既存コミッターがコミッター候補者を推薦する際に参考にしている開発者の技術的・社会的活動（パッチ投稿および検証，開発に伴う議論など）の履歴を用いて，コミッター候補者予測モデルを構築している．

予測モデル構築に用いている活動履歴データを抽出する期間は，コミッター候補者の場合，活動を開始してから初めてコミットを行うまでの期間を，一般開発者の場合，活動を開始してから分析期間終了時までの期間をそれぞれ対象としている．各開発者のデータ収集対象期間を図 1 に示す．また，予測モデル構築のために収集する，開発者の活動メトリクスの概要を表 1 に示す．

予測モデルの構築において，多くの研究で利用されているロジスティック回帰分析 [11], [12], [13] を用いている．コミッターに昇格するか否かを目的変数とし，予測モデルの出力値（0 から 1 の連続値）が 0.5 以上のときにコミッターに昇格すると判別する．

3.2 本論文で提案する予測モデル構築手法

本論文では，3.1 章で述べたコミッター候補者予測モデル構築手法をベースとし，データサイズを考慮した予測モデル構築手法を提案する．データサイズを考慮するために，活動履歴データを 1 ヶ月単位で細分化し，活動開始時から N ヶ月をデータセット期間として，予測モデルを構築する．パラメータ N の取り得る範囲は 1〜12 とする．

本論文でも，先行研究と同様に，ロジスティック回帰分析を予測モデルの構築に適用する．モデルの評価は適合率（Precision），再現率（Recall），F1 値 [14] を用いる．

適合率は，コミッターに昇格すると判断された開発者のうち，実際にコミッターに昇格した開発者の割合を示す．また再現率は，全コミッターの中で，コミッターに昇格すると判別されたコミッターの割合を示す，ただし，適合率と再現率はトレードオフの関係にあるため，両方の評価指標が高いとき，性能の高い予測モデルが構築できたことになる．

そこで本論文では，適合率と再現率に加えて，適合率と再現率の調和平均で与えられる F1 値を評価指標の一つとして用いる．F1 値は以下の式で定義され，値が大きいほどモデルの判別精度が高いことを示す．

$$F1 - measure = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

4. 実験

表 2 実験対象データの概要

分析期間	2011/9 - 2014/12
全コミッター人数	47
コミッター候補者人数	43
一般開発者人数	248

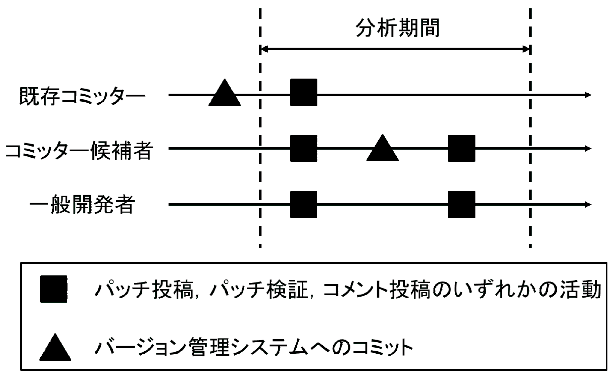

図 2 各分類の開発者の活動例

表 1　開発者の活動

活動	メトリクス名	内容
パッチ投稿	総パッチ投稿数	各開発者がパッチを投稿した総回数
	月パッチ投稿数	各開発者が 1 ヶ月にパッチを投稿した回数の中央値
パッチ検証	総パッチ検証数	各開発者がパッチを検証した総回数
	月パッチ検証数	各開発者が 1 ヶ月にパッチを検証した回数の中央値
開発に伴う議論	総コメント投稿数	各開発者がコメントを投稿した総回数
	月コメント投稿数	各開発者が 1 ヶ月にコメントを投稿した回数の中央値
活動期間	活動期間	各開発者の活動が実施された月数

4.1 概要

本実験は，開発者の過去の活動量に基いてコミッター候補者予測モデルを構築し，データサイズの大小によって，予測精度に違いが見られるかを確認することを目的としている．3 章で述べた提案手法を適用して予測モデルを構築し，先行研究の予測モデルを含めて予測精度の比較を行う，

4.2 対象データ

OSS プロジェクトである Apache Ambari プロジェクトを対象としてケーススタディを行う．本論文では，パッチ投稿，パッチ検証，開発に伴う議論に関する情報を収集するために，不具合管理システム Jira[*1]に 2011 年 9 月から 2014 年 12 月の間に報告された不具合票データを用いる．また，コミッター候補者と一般開発者を区別するために，バージョン管理システム git[*2]の 2014 年 12 月までのコミットログデータを用いる．本論文で対象とする分析期間，各分類の開発者数を表 2 に示す．　また，各分類の開発者の活動例を図 2 に示す．

分析期間の終了時点までに一度でもバージョン管理システムにコミットを行った開発者をコミッターと呼び，そのうち活動を開始してから初めてコミットを行うまでをコミッター候補者と呼ぶ．そして，分析期間終了時までにコミッターに昇格していない開発者を，一般開発者と呼ぶ．一方で，開発者の中にはプロジェクトに携わり始めたときからコミッターである場合もある．本論文では，分析期間以前からコミッターである開発者（既存コミッター）については，分析期間中の活動がその開発者のコミッター昇格に直接関係していないと考えられるため，除外する．

本対象データでは，47 人のコミッターを確認したが，4 人はパッチの投稿，パッチの検証，開発に伴う議論を行う前にバージョン管理システムにコミットしていたため，プロジェクトに参加した時点で既にコミット権限をもっていたと考えられる．そのため，本論文では残りの 43 人をコミッター候補者として実験を行う．

また本論文では，開発者が初めてバージョン管理システムにコミットを行った日時をコミッター昇格日時と定義し，コミッター候補者はコミッター昇格日時までの活動履歴データのみを抽出する．

[*1] https://issues.apache.org/jira/browse/ambari/
[*2] https://github.com/apache/ambari

表 3　コミッター候補者の予測結果

データセット期間	適合率	再現率	F1 値
1 ヶ月	0.90	0.43	0.58
2 ヶ月	0.52	0.81	0.63
3 ヶ月	0.80	0.76	0.78
4 ヶ月	0.68	0.81	0.74
5 ヶ月	0.74	0.81	0.77
6 ヶ月	0.65	0.81	0.72
7 ヶ月	0.67	0.86	0.75
8 ヶ月	0.67	0.86	0.75
9 ヶ月	0.67	0.86	0.75
10 ヶ月	0.67	0.86	0.75
11 ヶ月	0.67	0.86	0.75
12 ヶ月	0.67	0.86	0.75
先行研究の予測モデル	0.68	0.81	0.74

4.3 実験手順

3 章で述べた提案手法を適用し，各開発者の活動履歴データセット期間を，1〜12 ヶ月にそれぞれ区別し，各期間での活動メトリクスを算出する．コミッター候補者の活動履歴データ群と一般開発者の活動履歴データ群はそれぞれ 2 等分し，一方は予測モデルの構築のための学習データセット（コミッター候補者 22 人，一般開発者 146 人），もう一方はモデルを評価するためのテストデータセット（コミッター候補者 21 人，一般開発者 145 人）とする．そして，コミッター候補者予測モデルの構築を行い，先行研究の予測モデルを含めて予測精度の比較を行う．

5. 実験結果と考察

5.1 実験結果

表 1 で示す全ての活動メトリクスを用いて構築した，活動履歴データセット期間ごとにおける予測モデルの適合率，再現率，F1 値を表 3 に示す．また，本論文の予測モデルの予測精度と比較するために， 先行研究の予測モデルで予測した場合の結果を示す．実験の結果，データセット期間を

7 ヶ月，8 ヶ月，9 ヶ月，10 ヶ月，11 ヶ月，12 ヶ月としたときの予測精度は同じ精度であることが分かった．適合率が最も高いのは，データセット期間を 1 ヶ月としたときで，90%であった．再現率が最も高いのは，データセット期間を 7 ヶ月，8 ヶ月，9 ヶ月，10 ヶ月，11 ヶ月，12 ヶ月としたときで，86%であった．F1 値が最も高いのは，データセット期間を 3 ヶ月としたときで，78%であった．F1 値に着目し，先行研究の予測モデルと本論文の予測モデルの予測精度を比較すると，データセット期間を 3 ヶ月，4 ヶ月，5 ヶ月，7 ヶ月，8 ヶ月，9 ヶ月，10 ヶ月，11 ヶ月，12 ヶ月としたとき，先行研究の予測モデルと同程度以上の予測精度であった．

本実験では，ロジスティック回帰分析の出力値が 0.5 以上の場合にコミッター候補者として判定した．コミッターとして適当な開発者を正確に抽出したい場合はしきい値の値を大きく，多くのコミッター候補者を抽出したい場合はしきい値を小さくすることが可能である．ただし，しきい値を大きくしすぎると，コミッター候補者と判断される開発者数が減少するため，適合率は向上する一方で再現率は低下する．反対にしきい値を小さくしすぎると，コミッター候補者と判断される開発者数が増加するため，再現率が向上する一方で適合率は低下する．この点において注意する必要がある．

5.2 考察

実験の結果，データサイズの大小によって，予測精度に違いが見られることが分かった．しかしながら，データセット期間が 7 ヶ月よりも長い場合の予測精度は同じ精度であった．これは，学習データセット，テストデータセットそれぞれにおいて，8 ヶ月以上継続的に活動した開発者の数が，146 人中 4 人，145 人中 5 人と少なかったためであると考えられる．また，適合率が最も高いのは，データセット期間を 1 ヶ月としたときであった．追加分析した結果，プロジェクト参加開始時の 1 ヶ月間において，一般開発者のパッチ投稿数は平均値 1.07，中央値 0 で，コメント数は平均値 1.64，中央値 0 であった．それに対して，コミッター候補者のパッチ投稿数は平均値 7.05，中央値 3.5 で，コメント数は平均値 2.95，中央値 1.5 であった．この結果から，将来コミッターに昇格する開発者の多くはプロジェクト参加開始時から他の一般開発者に比べて，活発に活動していることが分かった．この顕著な活動量の差によって，高い適合率が算出されたと考えられる．

6. おわりに

本論文では，既存の予測モデルをベースとし，データサイズを考慮したコミッター候補者予測モデルについて検討を行った．実験の結果，データサイズの大小によって，予測精度に違いが見られることが分かった．また，データサイズが小さい場合でも，先行研究の予測モデルと同程度以上の予測精度を算出できる事が分かった．今後は，本論文で構築したデータサイズを考慮したコミッター候補者予測モデルの有用性を確認するために，異なるプロジェクトを対象として実験を行う．

謝辞 本研究の一部は，文部科学省科学研究補助金（基盤 (C): 15K00101）による助成を受けた．

参考文献

[1] Fogel, K.: *Producing Open Source Software: How to Run a Successful Free Software Project*, O'Reilly Media, Inc. (2005).

[2] Canfora, G. and Cerulo, L.: Supporting Change Request Assignment in Open Source Development, *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pp. 1767–1772 (2006).

[3] Shibuya, B. and Tamai, T.: Understanding the Process of Participating in Open Source Communities, *Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, FLOSS '09, pp. 1–6 (2009).

[4] Bettenburg, N., Premraj, R., Zimmermann, T. and Kim, S.: Duplicate bug reports considered harmful ... really?, *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, pp. 337–345.

[5] Nurolahzade, M., Nasehi, S. M., Khandkar, S. H. and Rawal, S.: The Role of Patch Review in Software Evolution: An Analysis of the Mozilla Firefox, *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops*, IWPSE-Evol '09, pp. 9–18 (2009).

[6] Ihara, A., Ohira, M. and Matsumoto, K.-i.: An Analysis Method for Improving a Bug Modification Process in Open Source Software Development, *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops*, IWPSE-Evol '09, pp. 135–144 (2009).

[7] Jensen, C. and Scacchi, W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, *Proceedings of the 29th International Conference on Software Engineering*, ICSE '07, pp. 364–374 (2007).

[8] Bird, C., Gourley, A. and Devanbu, P.: Detecting Patch Submission and Acceptance in OSS Projects, *Proceedings of the 4th International Workshop on Mining Software Repositories (MSR '07)*, No. 26 (2007).

[9] 伊原彰紀, 亀井靖高, 大平雅雄, 松本健一, 鵜林尚靖：OSS プロジェクトにおける開発者の活動量を用いたコミッター候補者予測，電子情報通信学会論文誌， Vol. 95, No. 2, pp. 237–249 (2012).

[10] Fujita, S., Ohira, M., Ihara, A. and ichi Matsumoto, K.: An Analysis of Committers Toward Improving the Patch Review Process in Oss Development, *Supplementary Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering (ISSRE2010)*, pp. 369–374 (2010).

[11] Basili, V. R., Briand, L. C. and Melo, W. L.: A Validation of Object-Oriented Design Metrics As Quality Indicators, *IEEE Trans. Softw. Eng.*, Vol. 22, No. 10, pp. 751–761 (1996).

[12] Takagi, Y., Mizuno, O. and Kikuno, T.: An Empiri-

cal Approach to Characterizing Risky Software Projects Based on Logistic Regression Analysis, *Empirical Software Engineering*, Vol. 10, No. 4, pp. 495–515 (2005).

[13] Liang, G., Wu, L., Wu, Q., Wang, Q., Xie, T. and Mei, H.: Automatic Construction of an Effective Training Set for Prioritizing Static Analysis Warnings, *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pp. 93–102 (2010).

[14] Herlocker, J. L., Konstan, J. A., Terveen, L. G. and Riedl, J. T.: Evaluating Collaborative Filtering Recommender Systems, *ACM Trans. Inf. Syst.*, Vol. 22, No. 1, pp. 5–53 (2004).