OSS を活用したエンタープライズソフトウェア開発向け メトリクスの抽出

Extraction of metrics in utilization of OSS in enterprise software development

株式会社富士通研究所　ソフトウェア研究所

Software Laboratories, FUJITSU LABORATORIES LTD.

○森田 純恵　菊池 慎司[1)]　水谷 拓人[2)]　伊藤 雅子[3)]　土屋 雅生[4)]　野中 誠[5)]

○Sumie Moirta Shinji Kikuchi[1)] Takuto Mizutani[2)] Masako Ito[3)] Masao Tsuchiya[4)] Makoto Nonaka[5)]

Abstract　In this paper, we discuss the extraction of metrics which are important to improve quality and speed in software development. Based on a questionnaire survey conducted to software development teams in an enterprise, we identified metrics related to the following three categories: (1) skills for identifying OSS suitable for the purpose of development, (2) skills for designing architecture of software combined with OSS and (3) skills for efficient software testing to keep quality of software. Through the measurements of these metrics in actual software development project, we confirmed that the extracted metrics are valid for assessing the quality and speed in software development.

1. はじめに

近年，ソフトウェア開発のスピード向上とコスト削減の有力な方法の一つとして,オープンソース・ソフトウェア(OSS)の利用が注目を集めている．OSS とは,オープンソースの概念に基づき,ソフトウェアのソースコードが無償で公開され，改良や再配布を行うことが誰に対しても許可されているソフトウェアである．OSS 導入にあたっての初期コストが不要であることや，特定企業の技術への依存を回避できるなどの理由から OSS のビジネスユースへの拡大が進んでおり，世界各国の政府調達などでも OSS の採用が進んでいる．

しかし，OSS を利用して製品やサービスを開発しても，結果的に，QCD(コスト(Cost)，品質(Quality),開発スピード(Delivery))の向上につながらないことが少なくない[1]．そのため,OSS を利用した製品やサービスを開発する上での QCD 向上要因を明らかにし，その状況を定量的な情報に基づいて把握することは重要な課題である．

株式会社富士通研究所　ソフトウェア研究所　主席研究員
Research Principal, Software Laboratories, FUJITSU LABORATORIES LTD.
〒211-8588 神奈川県川崎市中原区上小田中 4-1-1　Tel: 044-874-2124
e-mail:morita.sumie@jp.fujitsu.com
1-1, Kamiodanaka 4-chome, Nakahara-ku, Kawasaki, Kanagawa 211-8588, Japan
Tel:+81-44-874-2124

1) 株式会社富士通研究所　ソフトウェア研究所　主任研究員
Research Manager, Software Laboratories, FUJITSU LABORATORIES LTD
2) 株式会社富士通研究所　ソフトウェア研究所　研究員
Researcher, Software Laboratories, FUJITSU LABORATORIES LTD
3) 株式会社富士通　共通ソフトウェア技術本部　シニアプロフェッショナルエンジニア
Senior Professional Engineer, Software Technologies Unit, Fujitsu Ltd.
4) 株式会社富士通コンピュータテクノロジーズ　TMP・検証ソリューション部
Engineer, V&V Solution Dept. Test Methodology Producing Division,
FUJITSU COMPUTER TECHNOLOGIES LIMITED.
5) 東洋大学　経営学部　教授
Professor, Faculty of Business Administration, Toyo University

このような課題認識に基づき，本論文では，筆者らが過去に実施した「OSS を活用したエンタープライズソフトウェア開発における課題調査（以下，課題調査）」[2]に基づいて OSS を効果的に利用するためのメトリクスを抽出した結果と，それが有効であることを示す．まず，2 節において，事例から見た OSS 利用における課題と，課題調査[2]から明らかになった OSS 利用における課題を述べる．次に，これらの課題を踏まえて，高品質なサービスや製品をいち早く開発して世に送り出すため重要なメトリクスを，GQM（Goal- Question-Metrics）手法を用いて抽出した結果を 3 節で述べる．ここで抽出した「OSS の目利き力，設計力，品質保証力」に対応するメトリクスについて， 4 節でその有効性検証および考察を述べ，5 節でまとめと今後の課題を述べる．

2. OSS 利用における課題

2.1 事例から見える課題

企業でのソフトウェア製品やサービスの開発における OSS の利用形態は，「OSS 単純利用」と「OSS 改変」の二つに大別できるが，いずれの場合においても考慮すべき課題がある．OSS 単純利用は，コミュニティで開発された機能をそのまま利用する形態であり，一般に広く用いられる．OSS の機能が十分な場合は，OSS 単純利用により保守性や開発効率などの向上が期待されるが，OSS 自体の品質を企業がコントロールできないため，品質面での懸念がある．一方，OSS 改変は OSS そのものを企業の手により改変して製品やサービスに組み込む形態であり，OSS の機能が不足しており，かつ改変に伴うライセンス条件（ソースの公開義務など）が許容できる場合に用いられる．しかし，不特定多数の開発者によって開発されたソースコードを理解し改変する必要があるため，OSS 単純利用に比べて開発効率の低下が懸念される．

このような OSS 利用形態の違いが開発効率の差として現れることを，社内で開発された事例を通して示す．図 1 は，OSS 単純利用に相当する 2 製品（破線で図示）と，OSS 改変に相当する 1 製品（実線で図示）について，製品の改版に伴う開発効率の変化をリリース時期ごとに示したグラフである．ここで，開発効率とは，開発したソースコードのステップ数を，改版にかかった工数で割った値である．OSS 単純利用の 2 製品は，それぞれアプリケーションサーバの OSS および運用監視ツールの OSS を改変なしで利用している．一方，OSS 改変の 1 製品は，アプリケーションサーバの OSS を改変している．このグラフからは，OSS 改変の方が OSS 単純利用に比べて総じて開発効率が低く，リリースを繰り返してもその効率は向上しにくいということがわかる．なお，OSS を改変したケースにおいては，2012 年 1 月のリリースだけ著しく開発効率が低くなっているが，これはリリースの開発規模が非常に小さかったことに起因するためである．

また，OSS 利用形態の違いは品質面にも差が現れる．図 2 の実線は，図 1 と同じ OSS 改変の 1 製品について，OSS 起因の障害件数と，それ以外の理由による障害件数についていずれも正規化した値を示している．一方，図 2 の破線は，図 1 と同じ OSS 単純利用の 2 製品について，OSS 起因の障害件数について正規化した値を示している．OSS 改変では，OSS に機能追加するとともに品質面の向上を図ったため，OSS 起因かどうかによらず，比較的短期間で障害が収束していることがわかる．一方で，OSS 単純利用では，OSS 起因の障害が多く，収束するまでに長い時間を要して

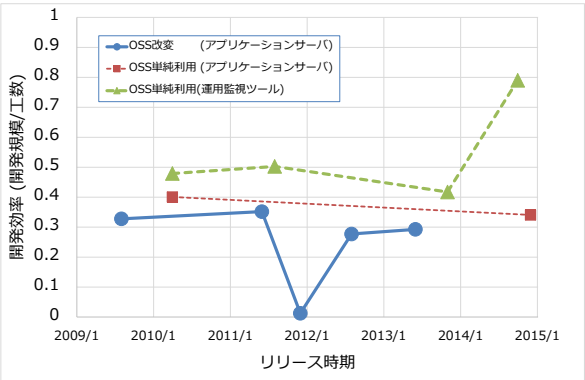

図 1　OSS 利用形態と開発効率

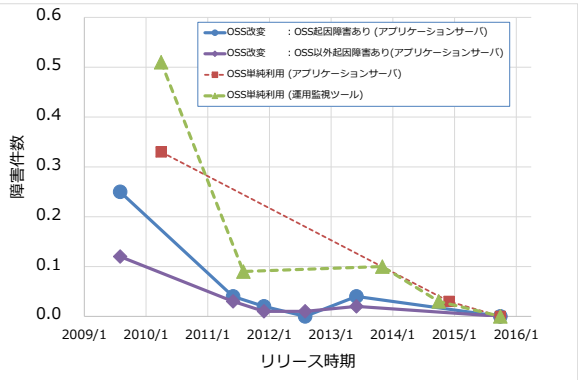

図 2　OSS 利用形態と品質

いることがわかる．

ここで紹介した事例では，OSS の利用形態によって開発効率と品質がトレードオフの関係になっている．ただし，このトレードオフは常に生じるものではなく，利用した OSS に大きく依存する．したがって，どのような OSS を選定するかという「OSS の目利き力」が課題となる．また，OSS 単純利用の場合は OSS の品質リスクが高いため，「品質保証」における工夫が求められる．

2.2 課題調査から見える課題

筆者らは，富士通株式会社において OSS を利用した製品やサービスの開発を行っているソフトウェア開発者約 130 名を対象に，OSS を利用したソフトウェア開発の実態とその問題点について調査を行った[2]．調査対象者はマネージャー層，リーダー層，担当開発者であり，さまざまなメンバーが含まれる．調査対象組織の開発プロセスは，CMMI 段階表現におけるレベル 3 相当である．プロジェクトの規模は 10～50 名である．詳細は文献[2]を参照されたい．

本調査によって明らかになった傾向は，主に下記 3 点にまとめられる．

(1) OSS を利用したソフトウェア開発においては，開発計画と保守の工程において QCD に関する問題が発生する割合が高い．
(2) OSS の機能やインタフェースを的確に理解することは，開発計画，設計，保守など，ソフトウェア開発における広範囲の工程に対して影響を与える．
(3) 開発計画時に最適な OSS を選定すること，設計における企業の OSS 活用戦略，アジャイル型の開発実践などがソフトウェア開発において重要になる．

3. メトリクスの抽出

3.1 GQM モデル

OSS を活用したソフトウェア開発における QCD 問題の要因を明らかにし，それらを改善するための施策を特定するために，これらの問題と関連するソフトウェアメトリクスの抽出を行う．メトリクスの抽出においては，GQM(Goal-Question-Metric)手法を参考にした．一般的な GQM 手法では，最終ゴール(G:Goal)をサブゴールに分割し，その達成状況を把握するための質問(Q:Question)を列挙する．そして，それらの質問に定量的なデータで答えるためのメトリクス(M:Metrics)を定める．本論文では，ゴールを達成するために把握しておきたいことを質問として挙げており，その具体的な方法をメトリクスに挙げている点に留意されたい．

本論文における GQM モデルの一部を図 3 に示す．ここでは，OSS を利用した製品の QCD 向上を最終ゴールとして設定した．QCD 向上という最終ゴールをサブゴールに分割する方法として，品質(Q)，コスト(C)，納期(D)を設定する方法が考えられる．しかし，2 節で示したように，ソフトウェアの開発効率と品質はトレードオフの関係にあるので，これらの要素を独立に扱って分析す

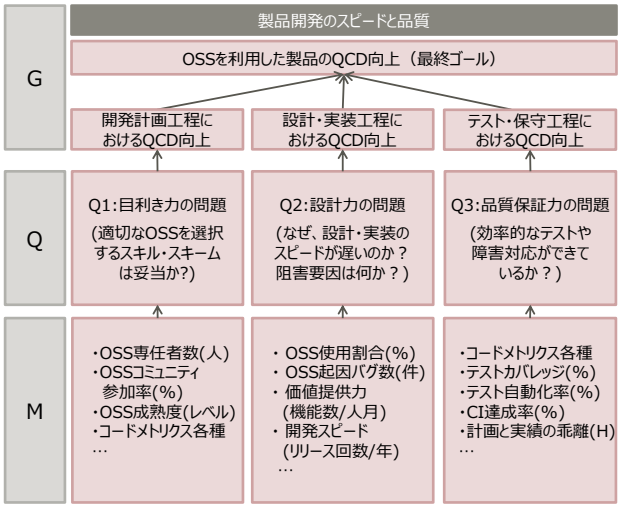

図 3　目利き力，設計力，品質保証力に関する GQM モデル（一部）

るのは不適切である可能性が高いと考えられる．そこで我々は，サブゴールとして 2.2 節の課題調査でも着目したソフトウェア開発工程(開発計画，設計・実装，テスト・保守)のそれぞれの QCD 向上をサブゴールとして設定することで，QCD 問題を分割することなく全開発工程における重要メトリクスを抽出することを試みた．これらのサブゴールに対して，それぞれの達成における問題点を示す質問(Q1,Q2,Q3)として，以下のものを抽出した．

(Q1) 開発計画：　適切な OSS を選択するスキル・スキームは妥当か？
(Q2) 設計・実装：　なぜ，設計・実装のスピードが遅いのか？
(Q3) テスト・保守：　効率的なテストや障害対応ができているか？

さらに，これらの質問を一般化し，開発計画，設計・実装，テスト・保守のそれぞれの工程において開発に求められる能力として，「目利き力」「設計力」「品質保証力」を以下のように定義する．但し，テスト工程に関しては，便宜上，「品質保証力」に分類したが，「設計力」「品質保証力」の両方に関係が深い．

(1) 目利き力：ソフト開発において，機能・非機能要件やライセンス，セキュリティや保守など，様々な観点に基づき，製品に採用すべき OSS を選定・採択の判断を行うことのできるスキル
(2) 設計力：保守の容易性や OSS 乗り換えの可能性，OSS のインタフェースや使用法を正しく理解した上で，適切なソフトウェア設計や実装を行うことのできるスキル
(3) 品質保証力：OSS の版数アップやパッチ適用，ブラックボックス化した OSS のテストや障害対応などを適切に行い，ソフトウェアとサービスの品質を保ち続けることのできるスキル

次節以降，これらの能力に関する質問・回答を定量化するためのメトリクスについて，その抽出過程での考え方を示す．

3.2 目利き力と品質保証力に関するメトリクス

開発計画時の OSS の選定における「目利き力」と，OSS 単純利用による「品質保証力」の課題を改善する施策として，OSS コミュニティから提供される情報，第三者から提供される情報を利用した OSS 事前評価のしくみを導入し，目利き力を向上させることが効果的であると考えられる．さらに，エンジニアが OSS コミュニティそのものに参加して，自社製品に必要な機能にいかに合った OSS を選定するか，OSS のパッチ貢献への影響力をもつか，社会貢献として OSS を改善する活動をしていくことも企業の役割であることが課題調査[2]や文献[3]からもわかる．これらの開発能力を示すメトリクスとして，まず，OSS の品質をソースコードの複雑度などから評価するコードメトリクスやテスト品質を評価するテストカバレッジを挙げ，「目利き力」は OSS 専任者数，OSS コミュニティ参加率，「品質保証力」は，テスト自動化率 および CI 達成率などを挙げることができる．

3.3 設計力に関するメトリクス

ソフトウェアの「設計力」として，QCD 問題発生の抑制のために，OSS のインタフェースや機能を理解した上でソフトウェア開発を行うことが重要である．正しい理解のないまま安易に OSS のブラックボックス利用をしてしまうと，テスト工数が増大し，ソフトウェア開発における広範囲の工程に対して影響を与える可能性がある．エンジニアのコード読解力も重要であるが，OSS の利活用による開発スピードと品質を高めるためには，ツールとテストの自動化を徹底したアジャイル型の開発スタイルが有効であることが課題調査[2]や文献[3]からわかる．

これらのことから「設計力」のメトリクスとしては，OSS 使用割合や OSS 起因バグ数，価値提供力(ストリーポイント/人月)，開発スピード(リリース回数/年)の他に，ツールを駆使するアジャイル型の開発スタイルを実践するための項目を挙げる[4]．

3.4 抽出したメトリクス一覧

以上の考え方に基づき，目利き力，設計力，品質保証力に関するメトリクスとして，従来から収集してきたソフトウェアメトリクスを目利き力，設計力，品質保証力の観点で再分類したメト

リクス一覧を表 1 に示し，今回の GQM 法から抽出した OSS 活用ソフトウェアメトリクス一覧を表 2 に示す. なお，紙幅の都合上，それぞれの詳細な説明は省略する.

4. メトリクス有効性検証および考察

本節では，３節で示したメトリクスについて，実プロジェクトでの計測とその有効性検証の実証事例について紹介する．これらのメトリクスのうち，表 1 に示したメトリクスは，OSS 活用に依存しない従来の開発においても測定・活用してきているものであることから，その有効性検証はここでは割愛する．一方，表 2 に示した今回新しく定義したメトリクスは，随時，新規プロジェクトから計測を開始中であるが，測定期間がまだ短い．そのため, 全ての検証はできないが, 変化のみえてきているメトリクスにつき, 有効性検証を行い, その考察を述べる.

表 1 目利き力，設計力，品質保証力に関する従来のソフトウェアメトリクス一覧

分類	メトリクス	収集項目	単位
I_目利き力	OSSコミュニティからの提供情報	ソースコード規模	LOC
		ドキュメント量	-
		バージョン情報	-
		コードメトリクス各種	-
		ライセンス種別	-
	品質	BTSに登録されている欠陥数	件
		修正までの日数	日
		パッチリリース頻度	間隔（日）
	脆弱性	件数	件
		重要度	CR/MJ/MN
D_設計力	開発規模(ソフトウェア)	自社開発ステップ数(新規、改造)	LOC
		OSS規模(新規、改造)	LOC
	工数とコスト	設計・実装・テスト・保守	人時
		集合レビュー時間	時間（H）
Q_品質保証力	母体品質	未修正障害件数	件
	設計品質	レビュー指摘された欠陥数（集合レビュー）	件
		欠陥の重要度	CR/MJ/MN
	テスト品質	検出欠陥数（ストーリー完了後のみ）	件
		欠陥の重要度	CR/MJ/MN
		デグレード件数	件
	出荷後品質	出荷後問題発生件数	件

表 2 目利き力，設計力，品質保証力に関する OSS 活用ソフトウェアメトリクス一覧

分類	メトリクス	収集項目	単位
I_目利き力	コミュニティ活性度	ユーザー活性度（汎用ML量月次平均）	件/月
		参加者数(ユーザー)	人
		開発者活性度（開発者ML量月次平均）	件/月
		参加者数（開発）	人
	第三者公開情報(SCSK社/Bluckdug社等)	総合スコア	点
		サポート	-
		プロジェクト継続性	経過年数
		Web検索Hit数	件
		SlideShare数	件
	自社使用実績	OSS名一覧、有識者情報、製品名、製品数	-
	OSSの世界で通用するエンジニア数	OSS専任者数	人
		コミッター数	人
		コミット数	件
D_設計力	開発規模(ソフトウェア)	OSS 使用割合	%
	価値提供力	開発効率（ストーリーポイント）	SP/人月
	開発スピード	リリース頻度	回数/年
	計画力	計画と実績の乖離距離	時間（H）
Q_品質保証力	開発規模(テストコード)	ステップ数	LOC
	テスト自動化率	全テスト項目中、実行が自動化されている割合（LV1）	%
		全テスト項目中、結果判定が自動化されている割合(LV2)	%
	母体品質	バックログの内訳、内容	-
		OSS 未修正障害件数	件
	コードメトリクス（OSS、母体、設計分）	カバレッジ	%
		ネスト	ネスト数
		複雑度	複雑度数
	プロセス成熟度	非機能要件の達成度	件
		CI結果（成否推移）or CI成功率	-

4.1 「目利き力」に関するメトリクスの有効性検証

現在，主要 OSS のコード量や複雑度，障害件数などを一元的に閲覧可能な OSS 事前評価ツールを社内で構築して運用中である．このように，表 1 および表 2 に示したメトリクスの一部は実際に運用できており，利用可能であることを確認している．また，そのツールへのアクセス数や，OSS コミュニティへのコミッターやコミット数も確実に増加しており，海外拠点を含む OSS コミュニティで活躍するエンジニアも増えつつある．

目利き力に挙げたメトリクスのうち，OSS 専任者とコミュニティ参加者について，これらが QCD 問題と関連があることを以下に示す．表 3 に，2.2 節で示した課題調査[2]で OSS 起因の QCD 課題があると回答のあったプロジェクトにおける，OSS 専任者比率とコミュニティ参加者比率の平均値を示す．この表からは，OSS 起因の QCD 課題があると回答したグループは，無いと答えたグループと比較して，OSS 専任者比率とコミュニティ参加者比率のいずれも高い値を示している．このことは，OSS 専任者やコミュニティ参加者が多いと QCD 課題の発生確率が高まるということを意味しているのではなく，専任者やコミュニティ参加者が多いプロジェクトほど，OSS に起因する課題を的確にとらえる能力が高いことを示唆していると考えられる．この結果より，OSS の目利き力を測るためには，これらのメトリクスが有効であると考えられる．

4.2 「設計力」に関するメトリクスの有効性検証

従来，ソフトウェア開発力を定量的に示す指標として開発量(LoC/人月)を基礎としたメトリクスが長らく利用されてきた．しかし，今日では OSS を活用してより少ない開発量でより多くの価値を提供することが求められる．そのため，表 2 に示したメトリクスでは価値提供力(ストーリーポイント/人月)や開発スピード(リリース回数/年)などがより重視すべき指標であると考えられる．ただし，文献[4]でも指摘されているとおり，これらのメトリクスは性質の異なる複数のプロジェクトを比較するのに適していないことは，運用上で注意すべきことである．

図 4 と図 5 は，2.2 節で示した課題調査からわかった２つの傾向を示したものである．図 4 は，OSS を活用する開発において，なんらかの工程に QCD 課題の意識ありと回答したチームと，QCD 課題は無いと回答したチームにおけるテスト自動化率の差異を示したものである．この図からは，QCD 課題の意識ありと回答したチームの方が，課題なしと回答したチームより自動化率が高いということがわかる．このことも 4.1 節と同様に，自動化率が高いほど問題が発生するということを示しているのではなく，自動化率が高いチームの方が，QCD 課題に対して敏感で，高い問題意識を持っていることを示唆していると考えられる．図 5 は，今後 2〜3 年後に使いたいと思っている領域ごとの新しい OSS の回答数を３つに分類し，そのチームにおける開発ツール導入率の差

表 3　OSS 専任者と OSS コミュニティへ参加者の比率

メトリクス	OSS起因のQCD課題		全体平均
	あり	なし	
OSS専任者比率 平均値(%)	8.9	3.7	6.0
OSSコミュニティ参加比率 平均値(%)	1.4	0.9	1.1

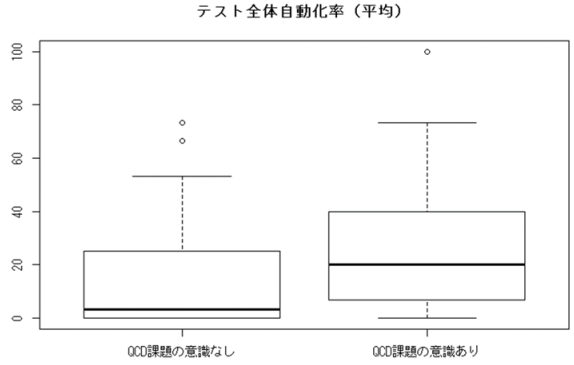

図 4 QCD 課題の有無とテスト全体自動化率の関係

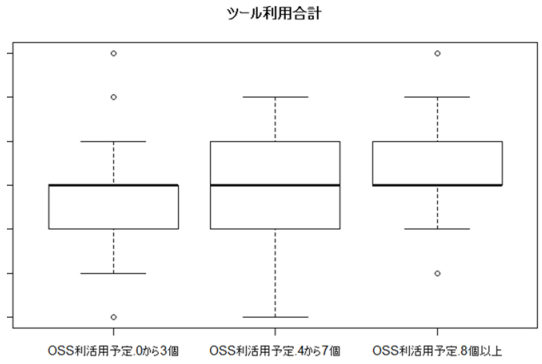

図 5 新 OSS への興味とツール導入率の関係

異を示したものであるが，新しい OSS によりチャレンジしようとしているチームのツール導入率が高いことがわかる．

なお，「設計力」に関するメトリクスについては，2.2 節で示した課題調査から OSS 利活用による QCD に関する問題が発生する割合はあまり高くないため，OSS 使用割合や OSS 起因障害数についてのみ測定するものとして，アジャイルメトリクス[4]に準じることができると考える．

4.3 「品質保証力」に関するメトリクスの有効性検証

テスト自動化率について，自動化への取り組み方法を具体的に紹介し，その適用効果を述べる．我々は，テストの自動化率を以下の 2 レベルに定義して計測した．ここでは，投入した工数と，それを実行したことによる削減工数について有効性検証を行う．

(1)　自動化率の Lv1 は全テスト項目中，テスト実行が自動化されている割合
(2)　自動化率の Lv2 は全テスト項目中，テスト結果の判定も自動化されている割合

この自動化率は，Lv1，Lv2 の両面において向上させることで，投資対効果という意味でコストの削減に効果があることを図 6 に示す．図 6 は，Lv1 と Lv2 を合算した自動化率(最大値は 200%)が 100% 未満の場合と以上の場合において，自動化に取り組んだ 1 年間における，各プロジェクトでの自動化に費やした工数と，それによる削減効果の比を表している(例えば，年間 100 万円の自動化投資で年間 150 万円の削減効果があった場合，投資効率は 150%とする)．この分布をみると，自動化率が小さい(合計 100%未満)の方は投資効果の中央値が 50%であるのに対し，自動化率が大きい(合計 100%以上)場合は投資効果の中央値が 100%を超えるなど，自動化率が高い程投資効果が大きくなるという傾向があることがわかる．

他にも，新しい OSS にチャンレンジしているチームは，積極的にコミュニティに参加するとともに，OSS 専任者をおいてより成熟度の低い（新しい）OSS の活用に対してしっかりとしたプロセス成熟度の中でチャレンジすることで開発のスピードをあげようとしている傾向があることが，課題調査[2]からわかっている．このことから，OSS 利活用による開発は，このテスト自動化を徹底しアジャイル型の開発スタイルで CI/CD(継続的インテグレーション，継続的デリバリー）を実装する環境により，変化の激しい OSS への追従を実現していくことが必須であると考えられる．

5. まとめと今後の課題

本論文では，ソフトウェア開発のスピード向上とコスト削減のための OSS の利活用課題を明らかにすることを目的に，最初に，筆者らが過去に実施した「OSS を活用したエンタープライズソフトウェア開発における課題調査[2]」から見える課題と OSS 利用事例から見える課題を示した．次に，その課題からソフトウェア開発における OSS 利用では，開発計画と保守工程の QCD 課題の割合が多く，その要因は，目利き力，設計力，品質保証力にあり，従来とは異なるスキルが要求

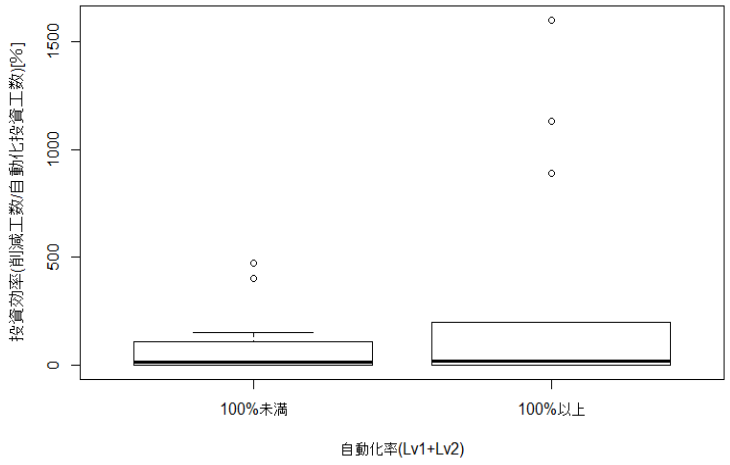

図 6　テスト自動化における投資効率

されていることが明らかになった.これら3つの軸がOSSを利用したソフトウェア開発における重点課題と捉え，OSSを利用した製品のQCD向上を上位ゴールとしたGQM (Goal-Question-Metrics)モデルを検討し，この 3つの軸に対応するソフトウェアメトリクスの抽出,有効性検証を行った.

目指すゴールは，開発計画時にOSS事前評価データを活用して最適なOSSを選定し,企業のOSS活用戦略に則った設計を行い，アジャイル型の開発スタイルをもって設計力を高めながら，進化し続けるOSSに対して静的解析ツールや可視化するツールを活用して品質保証を可能とすることである．オープンな世界においては，国際標準化の動きと並行して開発し，デファクトを狙うという従来の企業のOSS戦略モデルは消えていくと考えられる．代わりに，OSSコミュニティで実際にエンジニアが開発したコードが標準スペックとなり，ドキュメントが後を追うというスタイルになっているため,このような現実を鑑みて戦略を決めていかねばならない.加えて，開発の自動化には技術だけでなく開発文化やプロセス,開発者自身も変わる必要がある.たとえば，同じプロセスで自動化だけをしても大幅な時間短縮ができず，プロセスと文化を変えることに目をむけ,導入にむけたポリシーを策定することで，フルパイプライン試験を実施すると，44日かかっていた試験が2時間まで短縮できたという事例報告もある[5]．いまこうしたことが北米の西海岸を中心に急速に展開されている.

従来では，ソフトウェア開発力を定量的に示す指標(KPI: Key Performance Indicator)は開発量(行/人月)だったのに対し,本論文ではソフトウェア開発力を示すKPIを再定義した．そこでは,目利き力，設計力，品質保証力として，ROI(Return On Investment：投資対効果)や価値提供数（ストーリーポイント）,ソース解読能力，利用品質，製品品質，保守効率等を考慮した． 加えて，アジャイル型の開発スタイルで重要な CI/CD（継続的インテグレーション，継続的デリバリー）の開発基盤として先行してテスト自動化に取組み，本論文にてその効果を数値で示した.

一方で，欧米でのOSS事情は,LinuxやAndroid成功の例が示すように，欧米では企業内で開発されたソースコードよりオープンソースの方が品質がよいことは歴然たる事実となっている．現在では，無償であることの価値はOSSの一番の魅力ではなく，むしろ“開発人員の拡大”にその価値があり，ある意味で開発コスト削減（その拡大された開発人員を無償で雇える）にある．そして，OSS化を前提に欧米企業はソフトウェアを考えており，寄稿などを通して企業がOSSコミュニティを積極的に誘導する形へと急激に変化している.

今後は，GQM法で設定したゴールに向けて，抽出したメトリクスを測定，継続的に社内プロジェクト適用を推進し,目標を達成していくが，このためには，前述のプロセスと文化を変えることも必要であることを念頭において，その効果の定量的評価をしていく予定である.

謝辞　本稿の執筆にあたり，富士通研究所・金澤裕治主任研究員に多くのご助言を頂いた.ここに謝意を表する.

参考文献

[1] Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A., & Ubayashi, N. (2013). A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering*, 39(6), 757-773.

[2] 森田純恵, 菊池慎司, 宗像一樹, 伊藤雅子, 土屋雅生, 小林達樹, 力武達, 野中誠. (2016). OSS利用エンタープライズソフトウェア開発における課題調査. ソフトウェアエンジニアリングシンポジウム2016論文集, 2016 (to appear).

[3] 伊原彰紀,大平雅雄.(2016). オープンソースソフトウェア工学, コンピュータソフトウェア, 33(1), 28-40.

[4] Kammelar, J. (2015). Agile-Metrics. http://nesma.org/2015/04/agile-metrics/ (2016-06-20参照).

[5] Jonathan Bryce at OpenSack Summit Austin 2016. http://www.slideshare.net/VirtualTech-JP/openstack-summit-austin-2016-openstack-20165 (2016-06-20参照).