ソフトウェア定量的管理にかかわる学術研究事例

東洋大学 経営学部 教授　野中 誠
九州大学 システム情報科学研究院 准教授　亀井 靖高
和歌山大学 システム工学部 准教授　大平 雅雄

産業界でのソフトウェアの定量的管理に参考となり、役立つことが期待される学術研究の事例について、筆者ら3名がそれぞれかかわった最近の研究事例と、そこから読み取れる知見を紹介する。その上で、産業界でのソフトウェア定量的管理に期待することを最後に述べる。

1　はじめに

ソフトウェア工学の一分野として、ソフトウェア開発に関する大量かつ多様なデータの分析を通じて、提案手法の有効性を実証したり、データに見られる関係性を抽出して知見を得たりする学術研究が盛んに行われている。そうした学術研究は、オープンソースソフトウェア（OSS）のリポジトリを主な分析対象としたMSR（Mining Software Repositories）研究などを中心に、2000年代になってから多くの研究者が取り組んでいる。OSS開発と産業界におけるプロプライエタリなソフトウェア開発の境界が融合しつつある今日において、MSRなどで積み重ねられてきた学術研究の知見は産業界においても有用であると考えられる。

本稿では、産業界におけるソフトウェア定量的管理にとって参考となり、役立つことが期待される学術研究の事例について、筆者ら3名がそれぞれかかわった最近の研究事例と、そこから読み取れる知見を紹介する。まず、亀井がかかわったモダンコードレビュー記録の分析について紹介する。次に、大平及び亀井がかかわった不具合修正タスクの最適割り当て問題に関する研究を紹介する。続いて、野中がかかわった組織の能力成熟度別に見たリリース後品質の善し悪しを分けるポイントに関する研究を紹介する。そして、亀井がかかわった直近の研究である、ソフトウェア進化とバグ予測の関係に関する研究を紹介する。最後に、これらを踏まえた上で、産業界でのソフトウェア定量的管理に期待することを述べる。それぞれの研究事例の詳細については、参考文献に挙げた情報をもとに原文をご参照されたい。

2　モダンコードレビュー記録の分析

Shimagakiら[1]は、ソニーモバイルコミュニケーションズ株式会社でのソフトウェア開発における約20,000件のコミット記録を対象に、コードレビューを実施したものとそうでないものを比較し、その違いがリリース後欠陥数に影響しているかどうかを分析している。その結果、コードの規模、コミッターの人数、及びリリース前に検出したバグ数がそれぞれ多くなるとリリース後欠陥数は増える傾向にあるが、コードレビューが未実施であってもリリース後欠陥数が増えるわけではないという、ソフトウェア品質管理の観点からの期待感とは異なる結果を示している。一方で、OSSの外部コードをレビューせずに取り込んだ場合、リリース後欠陥数が増える傾向にあるという、こちらは品質観点からの期待感に沿った結果を示している。また、得られた分析結果とその含意を開発者に伝えたところ、約7割の開発者がこれに賛同したことを示している。

この分析結果から、「内部コードのレビューは不要である」ということを筆者らが主張したいのではないことに留意されたい。ここで述べたいのは、このような分析の実施基盤としてWeb上のコードレビューツールGerritが利用されており、データ入力

のための人手による追加コストが不要であるという点である。そして、分析の結果として得られた情報を、開発プロセス改善のためのコミュニケーションツールとして利用している点である。OSSの開発では、地理的・時間的に分散した開発者がGerritのようなツールを使ってコードレビューを行うことが一般的である。このようなコードレビューは、従来型の対面によるフォーマルなインスペクションに対比して「モダンコードレビュー」と呼ばれている。Gerritのようなツールを使えば、レビュー実施記録の基礎データを容易に収集することができ、定量的管理に利用できるだけでなく、開発プロセス改善への有用なインプットとして利用できる。この研究事例は、そうした改善サイクルをスムーズに回すためにツール利用の観点から貢献できることを示唆している。

そして、モダンコードレビューと呼ばれる最近の取り組みを、組織的な定量的管理の文脈に位置付けることの必要性を述べておきたい。組織的な定量的管理の仕組みは、ひとたび確立されると組織の慣性力が働くために、データ収集や分析の枠組みを変えていくのが難しくなりがちである。一方で、OSSの開発においては、モダンコードレビューのようにツールや開発スタイルの進化が著しい。定量的管理の仕組みが技術の進化に取り残されることのないよう、技術動向や学術研究の動向に目を向けておく必要がある。そのような意味合いからも、この研究事例を捉えていただけると良い。

3 不具合修正タスクの最適割り当て

柏ら[2]は、不具合管理システムに報告された不具合の修正タスクを担当者に割り当てるときに、特定の担当者に修正タスクが集中することなく、かつ、プロジェクト全体での不具合修正時間をできるだけ短縮できるようなタスク割り当て手法を提案している。具体的には、不具合の生じたコンポーネントの分類、対応の優先度、その分類における開発者の過去の不具合修正時間などを入力情報として、整数計画法によりタスク割り当て案を作成するというものである。この手法をMozilla Firefox及びEclipse Pluginプラットフォームなどの大規模OSS開発プロジェクトの実績データに適用してシミュレーションを行った結果、一部の開発者に修正タスクが集中するという問題を緩和でき、プロジェクト全体での不具合修正時間を従来の方法に比べて約35～50％削減できる可能性があることを示している。

この手法は、TracやRedmineなどのタスク（チケット）管理システムがあれば適用可能であり、応用範囲は広いと言える。そのほかに、この手法を適用するベースとして、それぞれの開発者がそれぞれのコンポーネントの分類に対してどのくらいの不具合修正時間を必要としたのかといったパラメータ値が必要である。ただし、提案手法ではその中央値のみを用いていることから、典型的な修正時間の推定値を初期のパラメータ値とし、手法を運用しながら修正時間の実測値に基づいてパラメータ値を適宜更新していけば良いだろう。このように、この研究事例で示された手法は、実際のソフトウェア開発に適用する上でのハードルが低いことも特徴である。

実際のソフトウェア開発において、不具合修正タスクが特定の担当者に集中し過ぎないようにタスクを割り当てるといったこと自体については、日常的に行われているだろう。しかし、この手法が示すように、定量的なデータに基づいて不具合修正のタスク割り当て案を作成し、これを実際のタスク割り当ての際の参考情報として意思決定しているといった取り組みは、筆者らの主観的観測としてはほとんどないように思える。ソフトウェアの定量的管理はアクションに結び付けることができてこそ意味がある。そのため、ソフトウェア開発におけるデータに基づく意思決定を支える一つの方法として、このような研究に目を向けていただけると良い。

この研究事例では、OSSを対象に手法の有効性を検証している。筆者がこのような研究事例を産業界の実務者に紹介したときに、過去には「OSSの開発と産業界でのソフトウェア開発は異なるため、参考にならない」という批判をしばしば頂戴した。しかし、それは一昔前の話である。今日においてはOSSを取り入れたソフトウェア開発が当たり前となり、OSSとプロプライエタリなソフトウェアの関係性をソフトウェア開発戦略の中に位置付けることが重視される。より積極的には、OSSへの貢献を含めたオープンイノベーションも重要な戦略になるであろう。「OSSで有効性が示されたことは実務においても有効である可能性が高い」という認識を持つことが必要であろう。

4 組織の能力成熟度別に見た品質の善し悪しを分けるポイント

柳田ら[3]は、日本電気株式会社における522件のソフトウェア開発プロジェクトデータを対象に、CMMI（Capability Maturity

Model Integration）のソフトウェア能力成熟度レベル1及び2のそれぞれについて、リリース後品質の善し悪しを分ける要因を分析している。その結果、リリース後品質の善し悪しを分ける最初の特徴的な要因は、いずれの成熟度レベルにおいてもソフトウェア開発規模であるが、その閾値はレベル1よりもレベル2のほうが4倍以上大きいことを示している。つまり、レベル1の組織に比べて、レベル2の組織では、ソフトウェア開発規模が多少大きくなっても品質問題が生じるリスクを低く抑えられることを示している。

また、レベル1においては、開発規模の次に位置付けられる特徴的な要因はレビュー指摘数であり、これが「少ない」ほどリリース後品質が良いという、ソフトウェア品質管理の観点からの期待感とは逆の結果が得られた。すなわち、レビューで多くのバグを指摘できていることよりも、そもそもバグの混入数が少ないことのほうがリリース後品質の良さに影響しているものと思われる。一方、レベル2においては、開発規模の次に位置付けられる特徴的な要因はテストバグ数であり、これが少ないほどリリース後品質が良いという結果が得られた。こちらについては、品質管理の観点からの期待感と整合した結果であると言えよう。

この研究事例を通して述べたいこととして、第一に、データの現れ方は分析対象のセグメントによって異なり、それに伴って定量的管理の着眼点や基準値も変える必要があるということである。この研究事例では、それぞれの能力成熟度レベルのセグメントにおいて、どの程度の規模を超えたときにプロジェクト管理レベルを高める必要があるかを示唆している。また、その次に着目すべきポイントは、レベル1の組織ではバグ混入を抑制することであり、レベル2の組織ではテスト重視であることを示唆している。改善施策を異なるセグメントに対して一律に展開しても必ずしもうまく行かず、それぞれの状況に合わせた施策が必要である。この研究事例は、それを定量的に示す一つの方法を示している。

第二に、データ解析手法はプロジェクトの実態把握に役立つということである。この研究事例では、分類木と呼ばれる手法を適用することで、リリース後品質の善し悪しを分ける要因の階層的な分析を行っている。分類木により影響要因を階層的に示すことで、分析結果を直観的に理解することが容易になる。ソフトウェアの定量的管理においては、データで把握した結果を組織へとフィードバックすることが必要であり、結果の「分かりやすさ」も重要な要素の一つである。

第三に、実際にデータ分析をしてみると、期待とは異なる関係が見いだされることがある。この研究事例では、成熟度レベル1における第二の要素、すなわちレビュー指摘数が少ないほどリリース後品質が良いという関係が見いだされている。これを因果関係として捉えるべきではないことは言うまでもないが、このような事前の期待とは異なる結果は、開発プロセス改善に向けた議論の契機となり得る。

5 進化するソフトウェアにおけるバグ予測に対してどの範囲のデータを使用すべきか

McIntoshら[4]は、OSSの中でも変更が多く進化の激しいQTとOpenStackを対象に、ソースコードのコミットに対してバグが含まれるかどうかの予測を、約3年半にわたる、およそ38,000件のコミットデータを対象に実施している。分析の結果、OSSが進化していくにつれて、バグ予測の精度が大幅に低下していくという現象を示している。分析により得られた知見として、進化の激しいソフトウェアにおいては、直近3カ月程度のデータに基づいてバグ予測モデルを構築することが望ましいことなどを示している。なお、多くのバグ予測研究ではファイルを予測の単位としているが、この研究ではより細かな変更を予測の単位としている。

この研究事例から得られる知見は、過去データをどこまで使用して予測モデルを構築するべきかという問いについて、OSSの事例を通して実践的に示している点である。そして、進化の激しいOSSの変更履歴データに基づくバグ予測の精度を保つには、3カ月程度が望ましいという一定の見解を示している点である。もちろん、この期間は予測対象の変化の度合いや、得られるデータサイズにも依存するが、一定の目安として有用な知見と言える。

なお、このような予測モデルへの入力データ範囲をどこまでとすべきかについては、工数見積もりの分野でも研究が行われており（例えば[5]）、一般にウィンドウィングと呼ばれている。ただし、ソフトウェア開発プロジェクトを対象とした工数見積もりモデルと、ソースコードを対象としたバグ予測とでは、入力データの連続性と変化量の観点で大きく異なる。工数見積もりモデルにおけるウィンドウィングに興味がある方は、天嵜[5]の論文などを参照されたい。

6 産業界でのソフトウェア定量的管理への期待

本稿で紹介した研究事例を踏まえた上で、産業界におけるソフトウェア定量的管理への期待として、第一に、開発手法やツールの進化を定量的管理へと取り入れて、定量的管理の仕組みを進化させることが挙げられる。産業界におけるソフトウェアの定量的管理は、品質管理や品質保証の部門、若しくはプロジェクトマネジメントオフィスなどが担当することが多い。こうした部門は開発部門に比べて開発支援ツールを直接的に利用する機会が少なくなる傾向にあるため、開発現場でのツール利用と定量的管理の仕組みが乖離してしまいがちである。ツールの利用状況を把握し、ツールからのデータ抽出スキルを身に付けるなどして、開発と一体となって進化していくことのできる定量的管理の実現が期待される。あるいは、開発現場での工夫により生み出されたツールと連携した定量的管理の仕組みを、プロジェクトレベルや組織レベルへと引き上げ、組織内へと展開することが期待される。

第二に、意思決定を支援し、アクションに結び付く定量的管理の実現である。そして、その対象を、開発チームでの活動から組織的改善まで幅広く捉えることである。3節で紹介した不具合修正タスクの最適割り当ては主にプロジェクト単位で行われる活動であり、4節で紹介した品質の善し悪しを分けるポイントから得られた知見は組織的なプロセス改善にかかわる活動である。ソフトウェアの定量的管理と言えば後者の組織的な取り組みの例がなじみ深いと思われるが、プロジェクト単位やチーム単位での活動にも着目できると良い。ただ、そうした単位に着目してデータ分析が行えたとしても、適切なアクションへと結び付けることは容易なことではない。しかし、アクションにまで結び付けなければ、定量的管理を行う意味は薄れてしまう。チームレベルから組織レベルまでの範囲に応じた、開発プロセスの改善や品質向上につながるアクションの実践が求められる。

そして第三に、ソフトウェア工学研究コミュニティとの交流である。本稿は、野中・亀井・大平の3名による記事だが、このほかにも国内にはソフトウェア定量的管理にかかわる研究を行っている研究者がいる。もちろん、国外にも多数いる。こうした研究者らとの交流は、国内であれば、例えば情報処理学会ソフトウェア工学研究会のソフトウェアエンジニアリングシンポジウムなどに参加すれば意見交換の機会が得られる。逆に、学術研究に携わる研究者が、例えばIPA/SECの活動や日本科学技術連盟のソフトウェア品質シンポジウムに出向くなどして、産業界と交流するための場作りが求められる。まずは、参照可能な論文や発表資料を相互に学び合うところから始めると良いだろう。

7 おわりに

本稿では、産業界でのソフトウェアの定量的管理に参考となり、役立つことが期待される学術研究の事例について、筆者らがかかわったものを中心に紹介した。本稿の内容が、産業界における学術研究への関心につながり、ソフトウェア定量的管理更なる進化への足がかりとなり、産官学での交流がよりさかんになることを期待する。

参考文献

[1] Shimagaki, J., Kamei, Y., McIntosh, S. Hassan, A. E. and Ubayashi, N., A Study of the Quality-Impacting Practices of Modern Code Review at Sony Mobile, Proceedings of International Conference on Software Engineering (ICSE2016), Software Engineering in Practice (SEIP), pp. 212-221 (2016).

[2] 柏祐太郎, 大平雅雄, 阿萬裕久, 亀井靖高, 大規模OSS開発における不具合修正時間の短縮化を目的としたバグトリアージ手法, 情報処理学会論文誌, Vol. 56, No. 2, pp. 669-681 (2015).

[3] 柳田礼子, 野中誠, 誉田直美, CMMI成熟度レベル別に見たソフトウェア品質の良否にかかわる要因の複合的分析, SEC journal, Vol. 13, No. 1, pp. 8-15. (2017).

[4] McIntosh, S. and Kamei, Y., Are Fix-Inducing Changes a Moving Target? A Longitudinal Case Study of Just-In-Time Defect Prediction, IEEE Transactions on Software Engineering. (To Appear)

[5] 天嵜聡介, モデルに基づいた工数見積もりにおける重み付きMoving Window 法の評価, 情報処理学会ソフトウェア工学研究報告, Vol. 2012, No. 24, pp. 1-8 (2012).