自発的ソフトウェア進化における プロジェクトの成長過程可視化ツールの試作

樋口 凱斗[1]　玉田 春昭[1]　戸田 航史[2]　中村 匡秀[3]

概要：今日の開発においてソーシャルコーディングが一般的になりつつある．ソーシャルコーディングとは，ソフトウェアへの機能追加を上位下達ではなく，開発者自身から自発的に提案される方法である．このような開発形態を自発的ソフトウェア進化と呼ぶ．この進化の様子を可視化するための方法が提案されているものの，必要なデータ取得には複雑な操作，そして大きなコストを要する．本稿では，このデータ取得，グラフ描画を自動化して，プロジェクトの成長過程を可視化するためのツールの作成を目指す．

1. はじめに

近年，開発者自身が自発的に提案できるソーシャルコーディングという新たなソフトウェア開発方式が注目されている．ソーシャルコーディングでは，開発者から自発的にコードの変更が提案される．そして，管理者が提案を承認することでソフトウェアに機能が追加されたり，バグが修正されたり，ソフトウェアが進化・洗練されていく．これを自発的ソフトウェア進化と呼ぶ．

従来，ソースコードの編集履歴を記録・再生する手法が提案されている [1]．しかしながら，プロジェクトをどのように管理すれば，活発なコミュニティが形成され，ソフトウェアが良い方向に進化するかはわかっていない．プロジェクトでの開発の様子を記録・再生する方法が提案されているものの，環境構築に大きなコストを要し，また，分散開発という今日では当たり前になった状況に対応しているとは言い難い [2]．今日では，GitHub などのプラットフォームでの開発が当たり前になっており，それらプラットフォームからのデータによりプロジェクト管理の指針の導出を狙う．Toda らは自発的ソフトウェア進化の過程を表現するためにスマートシティのコンセプトを導入した [3]．そして，スター数やコード量，Issue 数などの推移を街の進化になぞらえることで，自発的ソフトウェア進化の可視化を試みた．さらに，GitHub で公開されているプロジェクト Atom，Brackets，VSCode の 3 つのエディタを対象としてケーススタディを示した．

しかしながら，GitHub からこの分析用のデータを取得するために複雑な GraphQL のクエリを作成する必要がある．また，データは膨大になるため一度に取得できず，何度もリクエストを送る必要がある．そして，単位時間内に送信可能なリクエスト数には上限がある．その結果，プログジェクトの規模によっては，1 つのメトリクスを抽出するのに数十分から数十時間を要し，様々なプロジェクトに対して，同様の分析を気軽に行える状態ではない．

そこで本研究では，ソーシャルコーディングによって開発されるプロジェクトに対して，プロジェクトの状態を性質づける基本的な属性（コード量，プルリクエスト率，イシュー数等）のデータを取得し，成長過程を示したグラフを自動出力する Argo を開発し，そのプロジェクトの状態推定の容易化を目指す．今日，ソーシャルコーディングのプラットフォーム（SCP）は数多くリリースされているが，本研究ではプロジェクト数が多く，API などが充実している GitHub を対象とする．

2. プロジェクトの成長過程可視化ツール Argo

本研究の目的は Toda ら [3] が提案したメトリクスによるプロジェクトの状態の可視化を容易に行えるようにすることにある．そのために，プロジェクトの成長過程のデータ抽出，可視化を自動的に行うツール Argo を開発する．Argo では自発的ソフトウェア進化が見られるプロジェクトの状態を性質づける基本的な属性（コード量，プルリクエスト率，イシュー数等）の成長過程のデータを取得し，グラフに描画するまでを行う．

[1] 京都産業大学
Kyoto Sangyo University, Kyoto 603–8555, Japan
[2] 福岡工業大学
Fukuoka Institute of Technology, Fukuoka 811–0295, Japan
[3] 神戸大学
Kobe University, Hyogo 657–8501, Japan

2.1 Argo の機能要求

Argo は SCP 上で行われる自発的ソフトウェア進化によるプロジェクトの成長過程を可視化するツールである．機能要求を以下に示す．

R1. プロジェクトメトリクスを取り出し，保存できる．
R2. 保存したメトリクスの差分のみを更新，取得できる．
R3. 保存したメトリクスからグラフを描画できる．
R4. 複数プロジェクトのメトリクスを 1 つのグラフに描画できる．

Argo の目的である可視化のためには，SCP から，もしくは版管理システムから必要なデータを取り出す必要がある（R1）．また，データの多くは API 経由で取得する必要があるため，取得には大きな時間がかかる[*1]．必要な時に毎回取得することは現実的ではない．そのため，キャッシュしておくことが必要になる．一方で，キャッシュされた情報に加え，日々情報が蓄積されていく．それらの差分のみを取得できることを表しているのが，R2 である．

R3 は保存したデータから Toda らが示したようなグラフを描画できることを表している．参考までに Toda らの描画したグラフを図 1 に示す．図 1 には，Atom, Brackets, VSCode の 3 つのエディタのスター数の推移が示されている．このように複数のプロジェクトのデータを同一グラフに描画することも Argo の機能要求の一つである（R4）．

2.2 Argo の非機能要求

Argo の非機能要求として，データ取得中に Argo が終了したとしても再度 Argo を起動することで問題なくデータ取得が再開できることが挙げられる．SCP の一つである GitHub では，リポジトリの情報を取得するために，REST API と GraphQL の 2 種類の方法がある．Argo では GraphQL を用いて情報を取得する．この時，GraphQL でのクエリには次に挙げる制限（上限）が課せられる[*2]．

L1. 1 回のクエリで得られる項目数
L2. 単位時間当たりのクエリ数

具体的な制限内容として，L1 では 1 回あたりのクエリで得られる項目の上限は 100 であり，例えば，Microsoft の VSCode のスター数のデータを取得しようとすると，2021 年 7 月 16 日現在，118,727 件のスター数であるため，1,188 回のクエリを投げなければ全件取得できない．L2 では，1 時間当たりに受け入れられるクエリ数の上限は 5,000 回であり，それ以上のクエリを投げるとエラーを返す．このため，L2 の上限を超えた場合，制限のリセット後に再度クエリを投げる必要がある．L1，L2 で定められた上限に達した場合や途中で終了した場合でも，問題なくデータ取得を再開できるようにしておく必要がある．

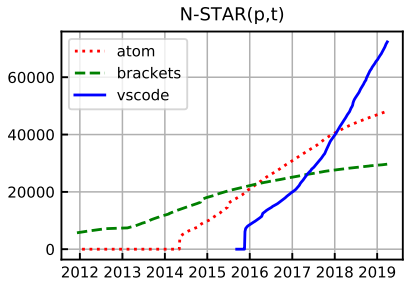

図 1 スター数の推移 [3]

[*1] 一番単純な star の履歴を microsoft/vscode から取得するのに 10 分 34 秒を要した（2021-07-16 時点で 118,727 件）．
[*2] https://docs.github.com/ja/graphql/overview/resource-limitations

2.3 Argo のワークフロー

ここでは Argo のワークフローを示す．最初にユーザは調査したいリポジトリ名と描画したい属性を Argo に渡す (1)．次に Argo は与えられた情報を元に GraphQL クエリを生成し (2)，GitHub GraphQL API にリクエストを送信する (3)．そして，得られたレスポンスから必要な情報をキャッシュに保存する (4)．最後に，保存したキャッシュからグラフを描画して結果として出力する (5)．

3. まとめ

今日のソフトウェア開発では，開発者の自発的な活動によりソフトウェアが成長していく．そのような成長を可視化するためには，複雑な操作を必要とし，多大なコストを要する．この問題に対して，グラフ出力を自動化するツール Argo を提案した．Argo により，調査が容易になることが期待できる．Argo 完成後には様々な分野のプロジェクトに対して同様の調査を行い，自発的進化のベストプラクティス，アンチパターンをまとめ，プロジェクトのより良い進化のための提案について検討する予定である．

謝辞 本研究の一部は JSPS 科研費 JP19H01138, JP20H05706 の助成を受けた．

参考文献

[1] 大森隆行，丸山勝久：プログラム開発履歴調査のための編集操作再生器，コンピュータソフトウェア，Vol. 28, No. 4, pp. 4_371–4_376 (2011).
[2] Goto, K., Hankawa, N. and Iida, H.: Project Replayer —An Investigation Tool to Revisit Processes of Past Projects, *Software Process Workshop*, pp. 72–79 (2006).
[3] Toda, K., Tamada, H., Nakamura, M. and Matsumoto, K.: Characterizing Project Evolution on a Social Coding Platform, *Proc. 20th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2019)*, pp. 525–532 (2019).

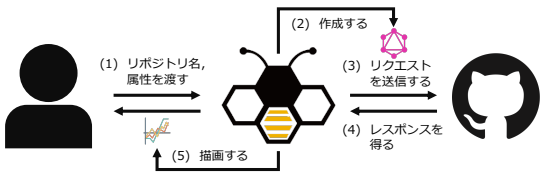

図 2 Argo のワークフロー