Codosseum: OSS プロジェクトモニタリング Web サービス

上村 恭平　中才 恵太朗　大神 勝也　畑 秀明　一ノ瀬　智浩
松本 健一　飯田 元

ソフトウェア工学において，ソフトウェアの開発状況やソフトウェアの安定性などの理解を促すための，開発履歴データを可視化する試みが多数行われている．可視化されたデータは，オープンソースソフトウェア（OSS）を開発，あるいは活用する際の指標として活用される．しかし，これまでに提案されている可視化の仕組みは独立しており，利用者は独自にデータを収集し，可視化システムに入力することで活用する必要がある．そこで，我々は収集した OSS プロジェクトの開発データの分析及び可視化を行い，提供する Web サービスである Codosseum を開発した．本論では Web サービスに求められる要件の整理及び，実際に構築したシステムについて説明する．

1　はじめに

ソフトウェアの開発状況や安定性の理解を促すことを目的に，開発に関わる情報の可視化に関する多数の研究が行われている．このような可視化された情報は，OSS のプロダクトを利用する際や，OSS の開発に携わる際の指標として利用される．しかし，既存の可視化を目的としたシステムは独立しており，利用者は個別に対象プロジェクトを可視化し，確認する必要がある．

我々はこれまで，Kataribe [3,9] という Web サービスを開発，公開してきた．これは，historage [5] という，メソッドレベルでの版管理できるリポジトリのホスティングするサービスであった．この Kataribe を発展させ，可視化機能の追加やメンテナンス性の強化などの拡張をし，Codosseum という新しい Web サービスとして提案，公開する †1．

Kyohei Uemura, Keitaro Nakasai, Katuya Ogami, Hideaki Hata, Ichinose Tomohiro, KenichiMatsumoto, Hajimu Iida, 奈良先端科学技術大学院大学, Nara Institute of Science and Technology.

†1　Codosseum: `http://codosseum.naist.jp/`

2　Codosseum の目的

Codosseum は，OSS をモニタリングし，開発への参加やプロダクト利用あるいは研究対象の収集を行う際の指標を提供する．開発においては以下を目指している．

- 研究成果の展開
- GitHub 以上にリッチな情報の提供
- 健全かつ活発なソフトウェア開発への貢献

研究において提案，開発したツールや可視化システムなどは，それぞれ独立に公開するだけでは使われにくい．特にソフトウェア開発プロジェクトの分析という同一の目的をもつ研究成果を同じ Web システムでサービスとして提供することで，研究者だけでなく実務者にも研究成果を展開しやすくなると期待できる．一方，研究としては発表しにくいサービスや分析であっても，それらが GitHub などになく，有用であれば実装し，提供したいと考えている．目指すべきところは，健全かつ活発なソフトウェア開発への貢献であり，研究などを実開発へ展開する実験の場である．

現在の実装で展開を試みているのは以下である．

- メソッドレベルの版管理システム historage [5]．メソッドレベルの不具合予測などへの応用がある [4]．

- 人口ピラミッド可視化 [7,8].
- ソースコード都市可視化 [6].

こうしたリポジトリや可視化といったデータや機能を Web サービスとして統一的に提供することで，誰もが Web ブラウザから手軽に利用することができるようにし，実際のソフトウェア開発への支援基盤となるとともに，研究のプラットフォームとなることを目指す．

3 システムに求められる要件

本章では本研究において提案システムを開発するにあたり定めた要件について述べる．まず，先行研究で開発した Kataribe について述べる．その後，Kataribe での問題点を踏まえて定めた要件について説明する．

3.1 先行研究での開発

我々はこれまでの研究において細粒度なソフトウェアリポジトリをホスティングする web サービスである Kataribe を開発した．Kataribe は 1 つのプロジェクト毎に 2 種類のソフトウェアリポジトリを扱う．1 つはファイル毎に変更を記録している，標準的な git リポジトリである．また，それに加え，メソッド単位で変更を記録した git リポジトリである historage を提供している．これらの 2 種類のリポジトリを，600 件のプロジェクトを対象に構築し，公開している．Historage は標準的な git リポジトリを変換することで構築するが，これには時間がかかるため，あらかじめ別環境で構築したデータをシステム上に転送している．これらのシステムは，OSS で開発されている git ホスティングサーバである gitlab を改造することで開発した．

3.2 今回の開発において定めた要件

システムを開発するにあたり必要な要件を，先行研究における Kataribe の開発の経験も踏まえて，以下のように整理した．

- データの最新性
- サービスの継続性
- 機能の拡張の容易性

以降，各要件について詳細を述べる．

3.2.1 データの最新性

Kataribe で提供していたのは，ある時点で GitHub からクローンしたデータであった．しかし，OSS の多くは日々開発が進められており，変更が加えられている．そのため，提案システムで可視化するデータが古いまま更新されなければ，実際のソフトウェア開発への支援基盤として利用することができない．したがって，本研究で提案するシステムには，OSS の開発の進捗に従い，可視化する対象のデータを更新し，最新の情報を提示することが求められる．

3.2.2 サービスの継続性

前述のように，Kataribe は既存の OSS である gitlab を改変することで実現していた．しかし，この改変はシステム全体への影響が大きく，gitlab がバージョンアップした場合に，その変更内容を反映するのが困難であり，セキュリティ対策のため必要なアップデートの適用も困難であった．継続してサービスを提供するためには，セキュリティ対策などのアップデートは必要不可欠である．従って，本研究で提案するシステムには，継続したサービスの提供のため，利用する OSS などには極力手を加えず，アップデートへの追従性を高める必要がある．

3.2.3 機能拡張の容易性

前項と同様，gitlab を改変してシステムを構築しているため，Kataribe の機能の追加時には，gitlab 全体のコードを把握し，他の機能と干渉することがないよう調査するなどの作業が必要であった．これらの作業には労力を要し，後から新しい機能を拡張するのが困難であった．

本研究で提案するシステムは，実際のソフトウェア開発の支援に利用することに加え，ソフトウェア工学における研究成果を活用するためのプラットフォームとする目的がある．そのためには，システムへの機能追加が容易に行える必要がある．

4 提案システムの機能と実装

4.1 システム概要

図 1 にシステムの概要図を示す．提案システムは，フロントエンドサーバと計算サーバ，及び 2 つのリ

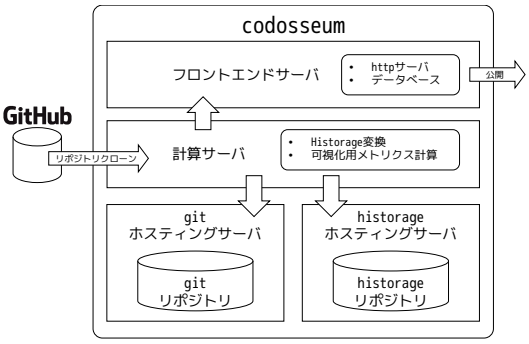

図 1　Codosseum の概要図

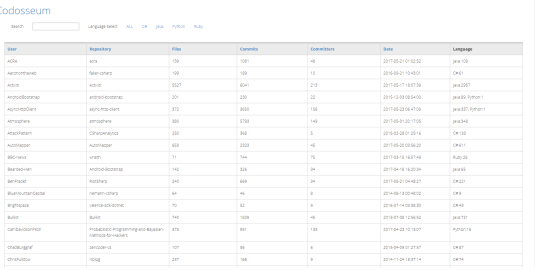

図 2　Codosseum のプロジェクト一覧

ポジトリホスティングサーバで構成される．システムは，まず，計算サーバが提供する対象となる開発データを GitHub から収集する．計算サーバは GitHub からクローンした git リポジトリを historage に変換し，元の git リポジトリと historage をそれぞれリポジトリホスティングサーバに登録する．また，git リポジトリと変換した historage を分析し，計算した可視化に用いるメトリクスをフロントエンドサーバ上のデータベースに登録する．フロントエンドサーバは，データベースに登録されているデータをもとに web ページを動的に生成し，http を通じて公開する．

以降，各サーバの動作および構成について詳細を述べる．

4.2　フロントエンドサーバ

フロントエンドサーバはプロジェクト一覧ページの表示および，リポジトリ毎のダッシュボードページを有する．図 2 はプロジェクト一覧ページのスクリーン

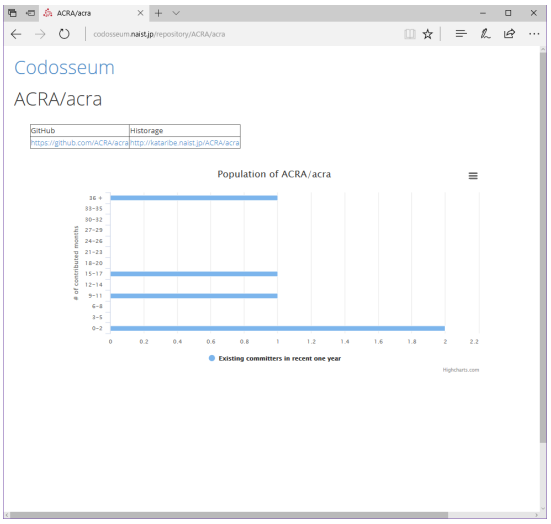

図 3　Codosseum のダッシュボード

ショットである．このページでは可視化情報が提供されているプロジェクトの一覧を表示する．また，ファイル数やコミット数，コミッターの人数，及び利用されている言語などのプロジェクトの概要も合わせて表示している．プロジェクト一覧からプロジェクトをクリックすることで，そのプロジェクトのダッシュボードページに遷移する．

ダッシュボードページの一例を図 3 に示す．ダッシュボードの上部には，元の GitHub へのリンクや，historage を提供するリンクを表示している．加えて，後述する可視化機能による可視化データも表示される．可視化機能はダッシュボード上に実装されるものと，別ページで実装し，ダッシュボード上にはリンクのみを表示しているものがある．

フロントエンドサーバの機能は Python を用いた Web アプリケーション開発フレームワークである Pyramid を採用した．Pyramid のテンプレート機能により，各ページはアクセスされた際にデータベースからデータを読み込み，動的に生成される．データベースには sqlite3 を利用した．

4.3　リポジトリホスティングサーバ

リポジトリホスティングサーバは，ソースコードの標準的な git リポジトリを提供するものと，それを変

換した細粒度なリポジトリである historage を提供するものの，2 つが存在する．標準的な git リポジトリを提供するサーバは，システム内部の可視化機能で利用することを目的としたもので，直接外部へ公開することを目的としたものではない．一方，historage を提供するサーバは，システム内部で利用に加え，historage を開発者や研究者へ提供することも目的としている．

リポジトリホスティングサーバには OSS で開発されている gitlab を採用した．先行研究では，標準の git リポジトリと historage の 2 種類のリポジトリを 1 つのサーバで公開するために，gitlab を改変して利用していた．本論で提案するシステムでは，それぞれのリポジトリ毎にサーバを分離させることで，改変することなく gitlab を利用している．これにより，gitlab のアップデートへの追従性を高く保っている．

4.4 計算サーバ

計算サーバは以下の 3 つの処理を担当する．

- GitHub から開発データのクローン
- git リポジトリの historage への変換
- 可視化に用いるメトリクスの計測

提供するデータの最新性を保つため，これらの処理は定期的に実行する必要がある．しかし，historage の変換にかかる時間は git リポジトリの大きさに依存し，開発期間が長いプロジェクトや，ファイル数が多いプロジェクトなどでは時間かかる．例えば，django の変換には，12 コア，メモリ 16GB のマシン上で 6 時間程度を要する．したがって，前述の処理をリアルタイムに行い続けるのは現実的ではない．そのため，提案システムでは 1 週間に一度，リポジトリの更新を行うこととした．

計算サーバで変換された historage や，計測されたメトリクスはそれぞれリポジトリホスティングサーバや，フロントエンドサーバ上のデータベースに登録される．このように，データの構築と外部への提供を分離し，各サーバの機能を最小限にすることで，機能拡張の容易性を向上させている．

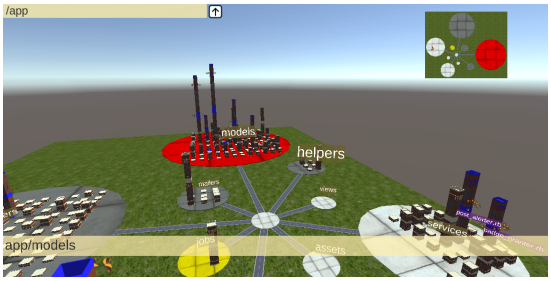

図 4　rocat のスクリーンショット

5　可視化機能

5.1　ソースコード都市

ソースコードの可視化として，都市をメタファとした可視化システムの CodeCity というものが提案されている [10, 11]．CodeCity の評価実験では，Eclipse と Excel による情報源に比べ，CodeCity による可視化はプログラム構造の理解や設計上の問題を解答するタスクの正確性が 24% 高く，解答にかかる時間が 12% 少ないという結果が報告されている [11]．我々は，rocat というソースコード都市の可視化を Kataribe に実装した [6]．Kataribe から Codosseum へ発展するにあたって，rocat も開発を進めた．

ソースコード都市 rocat はゲームエンジンの Unity[†2] で実装された GUI プログラムであり，リポジトリ解析で得られたプロジェクトの都市情報を読み込んで可視化する．rocat のスクリーンショットを図 4 に示す．円盤状の土台はプロジェクト内のディレクトリを，土台の上に建っているビルはそのディレクトリ直下に存在するソースコードファイルを表す．また，ビルの高さはコードの行数（LOC）を表し，横幅はビルのレイアウトを考慮し固定値とした．ビルはソースコードファイルの名前でソートして配置した．以下に詳細を説明する．

ディレクトリ：土台は現在可視化しているディレクトリを中心とし，そのディレクトリのサブディレクトリの土台が周囲に円形状に配置される．ディレクトリの親子関係を示すために，中心の土台と周囲の土台の

†2　Unity: https://unity3d.com/

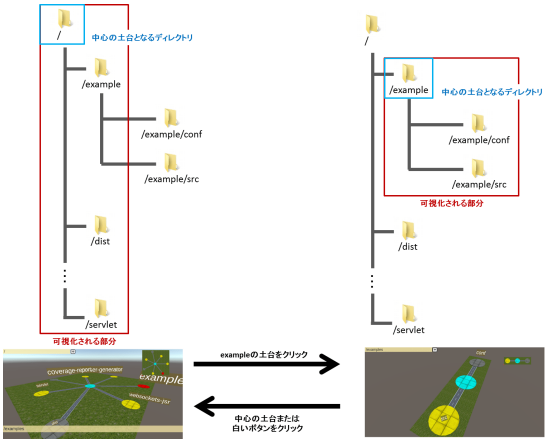

図 5　サブディレクトリ内部への都市の再構成

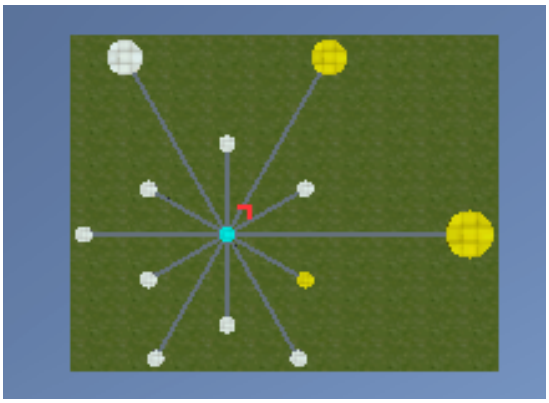

図 6　マップ

図 7　コード編集量の可視化

間には道路が作られる．周囲の土台が配置される位置はディレクトリ数やディレクトリ内にあるファイル数に依存し，土台同士が重ならないように決定される．

ディレクトリ移動に応じたソースコード都市の再構成：サブディレクトリの土台をクリックした際に，そのディレクトリ内にファイルなどが含まれる場合は，そのサブディレクトリを中心としてそのディレクトリ内のファイルやサブディレクトリを都市として再構成する．中心の土台または画面上部の白いボタンをクリックすることで 1 つ上のディレクトリを中心として都市が再構成される．都市の再構成の概念図を図 5 に示す．土台の上部にはその土台が示すディレクトリの名前が表示される．表示の大きさは視点との位置で変化するため，開発者は視点の遠くに土台があっても，その土台が示すディレクトリの名前を知ることができる．画面左上には現在可視化しているディレクトリのフルパスが表示される．

全体像のガイド：画面右上には都市を真上から見たマップが表示される．マップの拡大図を図 6 に示す．マップ上の赤い目印は視点の位置と方向を示しており，開発者は現在自分が都市のどこにいるのかを一目で知ることができる．マップ上の土台をクリックすることでも都市の再構成が可能であり，開発者は都市が大きくなっても任意のディレクトリの探索ができる．なお，マップの見やすさやクリックの操作性を考慮し，ビルはマップ上に表示していない．

操作：視点の操作はマウスとキーボードを利用する．Esc キーを入力することで中心の土台に視点を動かすことができる．また，Tab キーを入力することで，カメラを周囲の土台の近くに動かすことができる．Tab キーを連続で入力すると，視点を土台が示すディレクトリのアルファベット順に動かすことができる．これらの入力により，都市の大きさが大きくなっても，開発者は任意の場所に容易に移動することができる．

5.2　コード編集量

図 7 はコードの編集量に関する可視化の一例である．可視化画面は，上部の年数が書かれたボタン，中央の線グラフ，下部のテーブルによって構成される．

中央の線グラフは，プロジェクト全体におけるコー

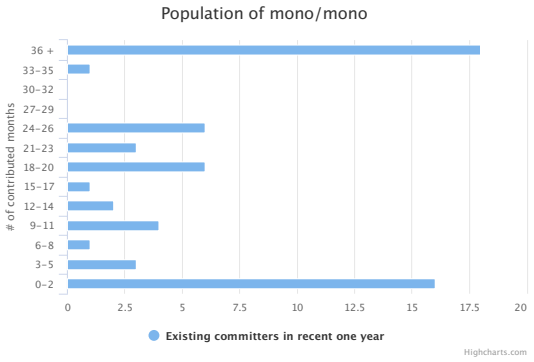

図 8　プロジェクト Mono の人口ピラミッド

ドの編集量の遷移を表す．緑線が追加の LOC，赤線が削除の LOC に対応しており，グラフは 1 週間ごとにプロットされる．初期状態では最近の年における遷移のみ表示されるが，上部にあるボタンを操作して過去の年へ遡ることが可能である．

下部のテーブルは，中央の線グラフが示す期間中に追加と削除が行われた部分の内訳のリストを示す．リストには変更された部分の名称，期間中に追加および削除された LOC の合計，追加と削除の遷移を表す線グラフが含まれる．初期状態ではファイルレベルでのリストが生成されるが，テーブルのヘッダに格納されたプルダウンメニューから，クラスレベルおよびメソッドレベルに変更することが可能である．クラスレベルおよびメソッドレベルの情報を表示する機能は，historage が持つ細粒度性によって実現されている．また，中央の線グラフの月ラベル（Jan など）をクリックすることにより，期間を特定の月に限定することも可能である．

5.3　コミュニティ人口構成

プロダクトだけでなく，ソフトウェア開発コミュニティの人的側面も，OSS プロジェクトのモニタリングにおいては重要な観点である．Onoue らは，開発コミュニティを人口ピラミッドで可視化，分析することで人口構成とその将来性を分析している [7,8]．この開発コミュニティの人口ピラミッド可視化を Codosseum においても実装する．

図 8 は，プロジェクトの人口構成の可視化例である．最終コミットから一年以内にコミットしたコミッターを現在のプロジェクトの人口として可視化している．人口構成要素としてはコミット経験月数を使用している．

横軸はコミッター数，縦軸は 3 ヶ月ごとのコミット経験月数を表示している．

6　関連研究

CodeCity は，ソースコードのクラスとパッケージの構造を都市のように 3D で可視化するシステムである [10, 11]．Balogh らはコンピュータゲームである MineCraft[†3] を用いてソースコードを都市のように可視化する CodeMetropolis を提案し，ソフトウェアテストに関連するメトリクスを可視化している [1, 2]．Balogh らのシステムでは関連のあるテストケースとソースコードを並べて配置して可視化することで，開発者の理解を支援している．

高澤らは，リポジトリマイニングのためのデータセット作成を支援するツールである RepositoryProbe を提案している [12]．

7　おわりに

本研究では，OSS プロジェクトモニタリング Web サービスである Codosseum を開発した．Codosseum は細粒度なソフトウェアリポジトリである Historage の公開と，それを用いた可視化機能を持ち，OSS 開発の支援と，ソフトウェア工学の研究成果を展開するプラットフォームとしての役割を担うことを目的としている．

Codosseum の開発にあたり，先行研究で開発したシステムでの経験から，システム構成に求められる要件を整理し，データの最新性，サービスの継続性，機能の拡張性が必要であることを確認した．この要件を満たすため，Codosseum は機能毎にサーバを分割するモデルで構築した．

現在，Codosseum 上には 500 件程度のプロジェクトを対象に，3 種類の可視化機能を実装している．可視化機能は今後も追加し，ソフトウェア工学の研究成

†3　MineCraft, http://minecraft.net/

果を公開するプラットフォームとして活用を進めていく予定である．また，OSS の開発支援として利用できるよう，対象とするプロジェクトも増やしていく予定である．

参 考 文 献

[1] Balogh, G. and . Beszdes: CodeMetropolis - code visualisation in MineCraft, *2013 IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, Sept 2013, pp. 136–141.

[2] Balogh, G., Gergely, T., . Beszdes, and Gyimthy, T.: Using the City Metaphor for Visualizing Test-Related Metrics, *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 2, March 2016, pp. 17–20.

[3] Fujiwara, K., Hata, H., Makihara, E., Fujihara, Y., Nakayama, N., Iida, H., and Matsumoto, K.: Kataribe: A Hosting Service of Historage Repositories, *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, New York, NY, USA, ACM, 2014, pp. 380–383.

[4] Hata, H., Mizuno, O., and Kikuno, T.: Bug prediction based on fine-grained module histories, *2012 34th International Conference on Software Engineering (ICSE)*, June 2012, pp. 200–210.

[5] Hata, H., Mizuno, O., and Kikuno, T.: Historage: Fine-grained Version Control System for Java, *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution*, IWPSE-EVOL '11, New York, NY, USA, ACM, 2011, pp. 96–100.

[6] Ichinose, T., Uemura, K., Tanaka, D., Hata, H., Iida, H., and Matsumoto, K.: ROCAT on KATARIBE: Code Visualization for Communities, *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science Engineering (ACIT-CSII-BCD)*, Dec 2016, pp. 158–163.

[7] Onoue, S., Hata, H., and Matsumoto, K.: Software Population Pyramids: The Current and the Future of OSS Development Communities, *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '14, New York, NY, USA, ACM, 2014, pp. 34:1–34:4.

[8] ONOUE, S., HATA, H., MONDEN, A., and MATSUMOTO, K.: Investigating and Projecting Population Structures in Open Source Software Projects: A Case Study of Projects in GitHub, *IEICE Transactions on Information and Systems*, Vol. E99.D, No. 5(2016), pp. 1304–1315.

[9] Uemura, K., Saito, Y., Fujiwara, S., Tanaka, D., Fujiwara, K., Iida, H., and Matsumoto, K.: A hosting service of multi-language historage repositories, *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, June 2016, pp. 1–6.

[10] Wettel, R. and Lanza, M.: CodeCity: 3D Visualization of Large-scale Software, *Companion of the 30th International Conference on Software Engineering*, ICSE Companion '08, New York, NY, USA, ACM, 2008, pp. 921–922.

[11] Wettel, R., Lanza, M., and Robbes, R.: Software Systems As Cities: A Controlled Experiment, *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, New York, NY, USA, ACM, 2011, pp. 551–560.

[12] 高澤亮平, 坂本一憲, 鷲崎弘宜, 深澤良彰: RepositoryProbe: リポジトリマイニングのためのデータセット作成支援ツール, コンピュータソフトウェア, Vol. 32, No. 4(2015), pp. 103–114.