Codosseum: オープンなソフトウェア開発・分析支援 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，ソフトウェア開発データを分析，可視化し，エビデンスに基づく意思決定を支援するソフトウェアアナリティクスが注目されている．分析や可視化の技術は多数提案されているが，独立して開発されているため，実務者や研究者にとって様々な技術を統一的に活用することは容易ではない．そこで我々は，オープンなソフトウェア開発データを有用な形式で公開し，複数の可視化を提供する Web サービス，Codosseum を開発している．具体的には，現在，メソッドレベルの履歴の可視化・リポジトリ提供，コミュニティメンバの経験期間による人口構成可視化，ソースコード都市可視化サービスを提供している．こうしたサービスを Software as a Service として提供することで，一般ユーザや実証的ソフトウェア工学研究者が容易にかつ無料でデータや可視化サービスを利用することを可能にしている．

Software analytics aims at helping practitioners make decision making based on evidence from various software development data by analyzing and visualizing various software development data, and it has been attracting practitioners and software engineering researchers. However, since newly developed research tools and methodologies are presented separately, it is difficult to use those techniques together. To tackle this problem, we are developing a Web service, Codosseum to provide preprocessed data and various visualization services. Currently, we provide method-level history visualization and repositories, software development community population pyramid visualization, and city-like code visualization. By implementing these techniques as SaaS (software as a service), they can be used by practitioners and researchers freely and easily.

1 はじめに

ソフトウェア開発に関する多量かつ大量のデータを分析し，エビデンスに基づく意思決定に活用することを目指すソフトウェアアナリティクスが研究者のみならず産業界でも注目されている [7,15]．ソフトウェア開発履歴の分析や，ソフトウェア開発状況の可視化は，こうしたソフトウェアアナリティクスの重要な技術である．しかし，ソフトウェアアナリティクスのための多様な技術やそれに関するツールなどは独立して研究，提案されており，一般ユーザにとってソフトウェア工学の最新の知見を活用することは容易ではない．

我々はこれまで，Kataribe [3,12] という Web サービスを開発，公開してきた．これは，ソースコードをメソッドレベルで履歴を分析することを可能にするリポジトリ，historage [5] を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．

1.1 貢献

本システムと特に関連する先行論文には，高橋らのソフトウェア論文 [16] がある．これは，ソフトウェ

Codosseum: A Web Service for Supporting Development and Analysis of Open Source Software Projects

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

コンピュータソフトウェア, Vol.34, No.4 (2017), pp.1–8.

[ソフトウェア論文] 2017 年 10 月 26 日受付.

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

ア工学で活発に研究されているリポジトリマイニングのためにデータセットの作成を支援するツール，RepositoryProbe を提案している．我々の提案する Codosseum も同様の目的のために，データセット，分析のための可視化などを提供する．また，Codosseum は，こうした機能を SaaS（Software as a Service）として実装することで,「そのソフトウェアを一般読者が容易にかつ無料で使用できること」を実現している．

我々は，旧システムの Kataribe を研究コミュニティで発表することで，普及のための努力をしてきた [3,12]（二番目の発表は対応言語の拡大により機能拡張したものである）．ユーザコミュニティの形成まではできていないが，国内外からの問い合わせや，提供するデータセットを用いた研究などが発表されている．一方，Kataribe のシステム管理や保守，機能拡張といった運用の経験から，システムに問題点があることが判明してきた．そこで我々は，Kataribe を Codosseum へソフトウェア進化させる大幅な開発を行った．

2 Codosseum の目的

Codosseum は，オープンなソフトウェア開発プロジェクトのデータを有用な形式で公開，可視化することで，ソフトウェア開発者，ユーザ，またソフトウェア開発データを分析対象とする研究者の開発や分析を支援することを目的としている．その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．本提案システムが発展的に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．

これまでに多数のソースコードの分析および可視化に関する研究が報告されている。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 [13,14]．Balogh らはコンピュータゲームである MineCraft[†2] 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． また、研究成果をサービス化する研究も既に存在する．Rosen らは，ソースコードのリスク予測に特化したアナリティクスのための Web サービス Commit Guru を提供している [11]．Munaiah らは，ソフトウェア工学研究者のために，GitHub 上の分析対象となり得るプロジェクトを特定し公開する Web サービスを公開している [8]．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 [16]．研究結果の再現性を高めるためにこのような Web サービスによるデータの共有は，研究コミュニティにおける大きな貢献と考えられている．ただし，その内容を更新，維持することはコストが高く，古い情報がそのままとなっていることも多い．提案する Codosseum では，データの更新，サービスの長期運用，機能の拡張などを考慮したシステム設計をしている．

現在の実装で展開を試みている機能は以下である．

- メソッドレベルの版管理システム historage [5] の提供．メソッドレベルの不具合予測などへの応用がある [4]．
- コミュニティの人的リソースを明らかにする，人口ピラミッド可視化 [9,10]．
- ソフトウェアの構造を視覚化する，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

†2 MineCraft, `http://minecraft.net/`

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．

例えば、Codosseum のプロジェクト一覧ではリポジトリの規模などを俯瞰して参照することができる。また、それらの Historage やコミュニティの人的リソースを提供をしている。ソフトウェア工学研究者はこれらの情報を分析対象とするリポジトリを選択する指標として活用することができる．また，OSS 利用者は，自身が利用したいと考えているプロジェクト毎のページを参照し，コミュニティの人的リソースやメソッドごとの変更頻度などの情報から利用する OSS の安定性，成熟度などを検討することができる．OSS 開発者も同様に，自身が開発に関わるプロジェクト毎のページを参照することで，ソースコード都市可視化や，メソッドごとの変更頻度などの情報からプロジェクトの概観を把握する手がかりとすることができる．

3 要件

先行システム Kataribe で判明した問題点を議論し，その問題点を踏まえて定めた要件について説明する．

3.1 先行システム Kataribe の問題

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．

しかし，先行システム Kataribe には 3 つの問題点があった．第一に，提供されるデータが古いという問題点である．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り，継続してデータが更新されていなかった．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，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．

第二に，セキュリティ対策のためのアップデートが困難であるという問題点がある．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．

第三に，新しい機能を拡張するのが困難であるという問題点がある．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．

3.2 提案システム Codosseum への要件

新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．

1. データの最新性
2. サービスの継続性
3. 機能の拡張の容易性

以降，各要件について詳細を述べる．

1. データの最新性. 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．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．

2. サービスの継続性. 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．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

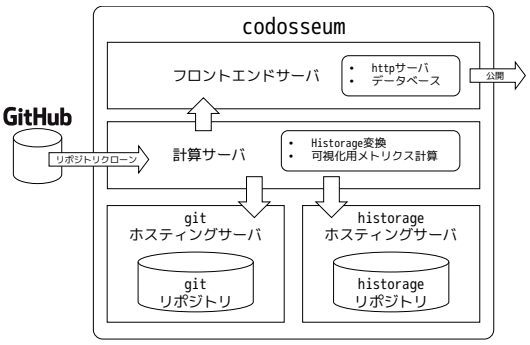

図 1　Codosseum の概要図

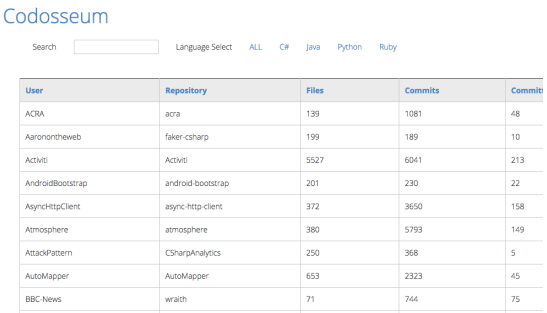

図 2　Codosseum のプロジェクト一覧

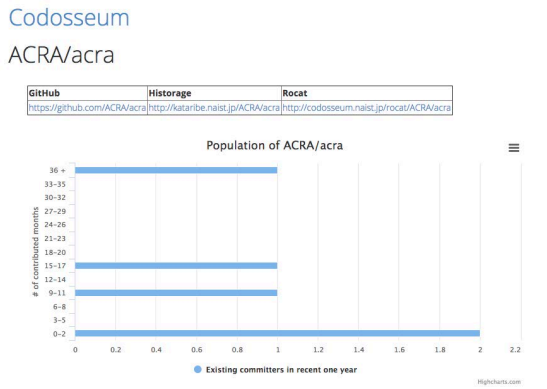

図 3　Codosseum のダッシュボード

る必要がある．

3. 機能拡張の容易性．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．具体的には，GitLab を改変せずに既存機能を利用するために，GitLab をラップする形でサービスを構築する．

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．このような構成にすることで，各可視化機能が独立し，機能追加の容易性やサービスの継続性を高めることができる．なお，本システムは，Windows 7 以降，または OS X 10.10 以降の OS の Edge, Chrome, Firefox または Opera のいずれかのブラウザでの利用を想定している．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．web フレームワークの選定基準として，軽量でフレームワーク自体の学習コストが低いことを重視した．この要件を満たすものには Pyramid の他，Flask などが挙げられるが，今後の拡張性の高さを考慮し，より柔軟性の高い 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．提案システムで利用するデータはリアルタイムに更新されるものではないため，可搬性を重視し sqlite3 をデータベースとして採用した．

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．規模の小さいプロジェクトでは 3 分程度で終わるものも存在するが，現在提供している OSS 全ての変換を行うには 80 時間要する．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．今後，提供する OSS の増加に従い，1 週間に一度の更新でも間に合わなくなる可能性がある．その際は，計算サーバを並列化することでデータの最新性を保つことができる．historage の変換は変換するプロジェクト毎に独立しているため，並列化対応は容易に行うことができる．

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，機能を追加する際の既存サービスへの影響を抑えている．また，データベースは共有しているため，新機能の追加時にはデータを再利用することで，計測に関する実装の手間を軽減することができ，機能追加の容易性を高めている．

5 可視化機能

Codosseum はソフトウェア工学における研究成果である分析や可視化を実用的に活用できる場としてユーザーに提供する事を目的としている．そのため，Katribe と比較して機能追加を容易とすることを要件とした．Codosseum の公開にあたりまず，ソース

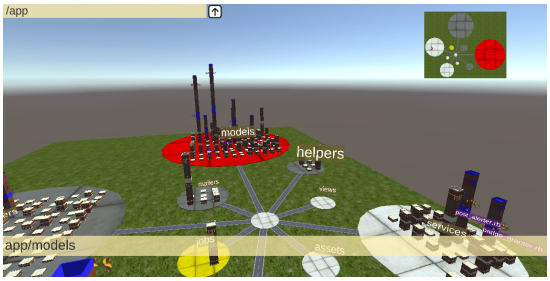

図 4　rocat のスクリーンショット

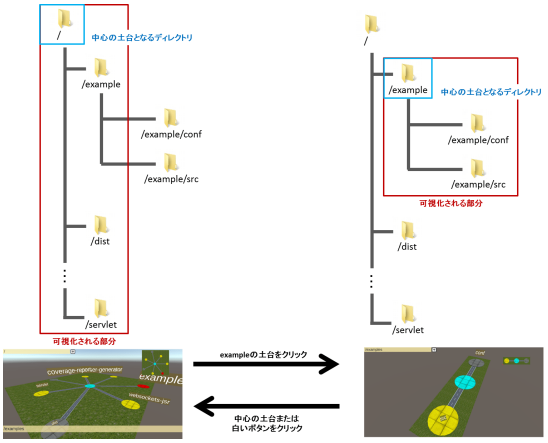

図 5　サブディレクトリ内部への都市の再構成

コード都市とコード編集量，コミュニティ人口構成の 3 種類の可視化機能を実装することとした．ソースコード都市は，OSS 開発者に対して, プログラム構造の理解を促すことを目的とした可視化である．コード編集量の可視化は, プロジェクトの開発の活発さや安定度を表し，プロジェクト自体に加えて，クラスやメソッド毎の開発状況がわかる．このような情報は OSS 利用者が利用する OSS を選択する際の指標として活用できる．コミュニティ人口はプロジェクトに関わっている開発者の参加期間を可視化するもので，これも OSS 利用者が指標として活用できるものである．以降，機能毎に機能の詳細と実装について述べる．

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 [13, 14]．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 [14]．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[†3] 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

†3　Unity: https://unity3d.com/

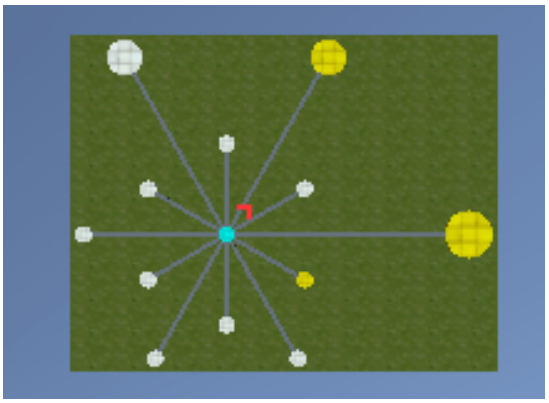

図 6 マップ

図 7 コード編集量の可視化

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ドの編集量の遷移を表す．緑線が追加の行数，赤線が削除の行数に対応しており，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行数の合計，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 [9,10]．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

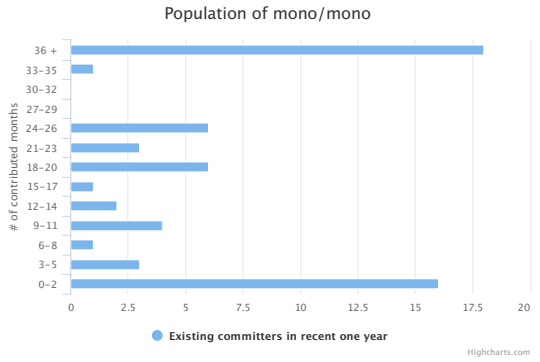

図 8　プロジェクト Mono の人口ピラミッド

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　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．このモデルにより，管理者らによる新機能の拡張性を高めることができた．例えば，可視化用のメトリクスを追加したい場合，計算サーバでメトリクスの計算に関する実装を行い，フロントエンドサーバではそのメトリクスの可視化に関する実装を行う．このようにシンプルなモデルになっているため，機能の拡張が容易である．

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．さらに，Codosseum 上で使用されるメトリクスはブラウザ上に表示されるのみであるが，今後，メトリクスを CSV や，WebAPI により提供予定である．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．

謝辞　これまでのシステムの開発に関わった，豊田工業高等専門学校 藤原賢二助教，槇原絵里奈氏，藤原雄介氏，中山直輝氏，藤原新氏，川島尚己氏，齊藤雄輔氏，田中大樹氏に感謝の意を表明する．

参 考 文 献

[1] Balogh, G. and Beszédes, Á.: 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., Beszédes, Á., and Gyimøthy, 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] Menzies, T. and Zimmermann, T.: Software

Analytics: So What?, *IEEE Software*, Vol. 30, No. 4(2013), pp. 31–37.
[8] Munaiah, N., Kroh, S., Cabrey, C., and Nagappan, M.: Curating GitHub for engineered software projects, *Empirical Software Engineering*, Vol. 22, No. 6(2017), pp. 3219–3253.
[9] 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.
[10] 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.
[11] Rosen, C., Grawi, B., and Shihab, E.: Commit Guru: Analytics and Risk Prediction of Software Commits, *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, New York, NY, USA, ACM, 2015, pp. 966–969.
[12] 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.
[13] 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.
[14] 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.
[15] Zhang, D., Han, S., Dang, Y., Lou, J.-G., Zhang, H., and Xie, T.: Software Analytics in Practice, *IEEE Software*, Vol. 30, No. 5(2013), pp. 30–37.
[16] 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.

上村恭平

平成 26 年大阪府立大学工業高等専門学校専攻科 総合工学システム専攻修了．平成 28 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．修士（工学）．現在，同大学院博士後期課程に在学．コードクローンを中心に，ソフトウェア工学分野の研究に従事．

中才恵太朗

平成 28 年近畿大学理工学部情報学科卒業．平成 30 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．修士（工学）．現在，同大学院博士後期課程に在学．ソフトウェア工学，特にソフトウェアリポジトリマイニングに関する研究に従事．

大神勝也

平成 28 年大阪市立大学工学部情報工学科卒業．平成 30 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．修士（工学）．ソフトウェアの可視化に関する研究に従事．

畑 秀明

平成 19 年大阪大学工学部電子情報エネルギー工学科卒業．平成 24 年同大学大学院博士後期課程修了．博士（情報科学）．平成 25 年奈良先端科学技術大学院大学助教．ソフトウェアエコシステムデザインの研究に従事．

一ノ瀬智浩

平成 27 年奈良工業高等専門学校専攻科電子情報工学専攻修了．平成 29 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．修士（工学）．ゲーミフィケーションによる作業支援に関する研究に従事．

松本健一

昭和 60 年大阪大学基礎工学部情報工学科卒業．平成元年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．平成 5 年奈良先端

科学技術大学院大学助教授．平成 13 年同大学教授．博士 (工学)．エンピリカルソフトウェア工学，特に，プロジェクトデータ収集／利用支援の研究に従事．電子情報通信学会, 日本ソフトウェア科学会，プロジェクトマネジメント学会各会員,IEEE Senior Member.

飯 田 元

昭和 63 年大阪大学基礎工学部情報工学科卒業．平成 3 年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．平成 7 年奈良先端科学技術大学院大学情報科学センター助教授．平成 17 年同大学情報科学研究科教授．博士 (工学)．ソフトウェアの開発支援環境や開発プロセスの研究に従事. IEEE，ACM 各会員.