社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

MetricsViewer: サービス指向リポジトリマイニングを活用したソフトウェアメトリクス可視化ツール

坂元　康好†　　柗本　真佑†　　中村　匡秀†

† 神戸大学 〒 657–8531 兵庫県神戸市灘区六甲台町 1–1
E-mail: †gen@ws.cs.kobe-u.ac.jp, ††{shinsuke,masa-n}@cs.kobe-u.ac.jp

あらまし　我々は先行研究においてサービス指向の考えを取り入れたリポジトリマイニングの実行フレームワーク SO-MSR（Service-Oriented Mining Software Repository）を提案した．さらに SO-MSR の考えに従い，様々なリポジトリの様々なソースコードからメトリクスを容易に取得できる Web アプリケーション MetricsWebAPI を開発した．本稿では，MetricsWebAPI のクライアントとして，個人の開発活動振り返り支援のための Web アプリケーション，MetricsViewer を提案し開発する．MetricsViewer はリポジトリ内からの対話的なファイル探索機能，及び自身の開発行動履歴の俯瞰を助けるためのメトリクス可視化機能を持つ．提案システムを用いることで，MSR に関する知識のないユーザでも手軽にリポジトリデータに基づいた，自己振り返りが可能となる．提案システムの予備実験として複数の被験者にシステムを利用してもらい，実際に自己振り返りに役立つかについて評価を行う．

キーワード　ソフトウェアリポジトリマイニング，サービス指向アーキテクチャ，ソフトウェアメトリクス，可視化，SO-MSR，MetricsViewer

MetricsViewer: A Software Metrics Visualization Tool Using Service-Oriented Mining Software Repository

Yasutaka SAKAMOTO†, Shinsuke MATSUMOTO†, and Masahide NAKAMURA†

† Kobe University, 1–1 Rokkoudai, Nada, Kobe, Hyogo, 657–8531 Japan
E-mail: †gen@ws.cs.kobe-u.ac.jp, ††{shinsuke,masa-n}@cs.kobe-u.ac.jp

Abstract　We have proposed a framework named SO-MSR: service-oriented mining software repository, which applied service oriented architecture to MSR. Following the SO-MSR, we have developed a web service, named MetricsWebAPI, for metrics calculation from a variety of software repositories and a variety source codes. In this paper, we develop and propose MetricsViewer, which is client of MetricsViewer and is a web application to support personal process improvement. MetricsViewer provides an interactive user interface for repository file exploring. Moreover the MetricsViewer visualizes change of source code metrics to support overhead view of personal process. End user can improve their development activities based on software repository data without MSR specific knowledge by using MetricsViewer. We have conducted a pilot study to evaluate the effect of proposed system for personal process improvement.

Key words　MSR, Service-Oriented Architecture, Software Metrics, Visualization, SO-MSR, MetricsViewer

1. はじめに

MSR（Mining Software Repository）とは，ソフトウェアの開発履歴データが蓄えられたリポジトリを発掘（マイニング）する行為や技術，あるいは研究分野のことを指す．MSR の実施により，定量的な根拠に基づいたプロセス改善や自己振り返りが実現できる．

我々は先行研究 [1] において，サービス指向アーキテクチャ（SOA: Service-Orieted Architecture）の考えを取り入れた MSR フレームワーク（SO-MSR: Service-Oriented MSR）を提案している．SO-MSR では，マイニングに係る様々な処理をマイニング実施者の視点から洗い出し，抽象的なサービスという単位でラップし公開する．個々のマイニング実施に必要な細かな技術やノウハウは，全てサービス内に隠蔽される．こ

れにより MSR 実施のハードルの低減を可能とするほか，他の MSR サービスとの組み合わせによる MSR の柔軟な拡張が期待できる．

さらに我々は，SO-MSR のフレームワークに従ったサービスの一つとして，MetricsWebAPI を開発している [1]．MetricsWebAPI は，MSR の代表的な方法の一つであるソフトウェアメトリクス計測のための Web サービス [2] である．このシステムでは，コミット差分の計算や抽象構文木の生成といったメトリクス計算のための処理はサービスの中に隠蔽される．またリポジトリの種類（SVN，CVS，Git 等）の違いや，ソースコードの言語の違いも全てサービス内で吸収される．．ユーザはこれらの情報を意識することなく，計測対象のリポジトリとソースコードへのパスといった最小限の情報を MetricsWebAPI に指定するだけで，簡単に必要なメトリクスを取得することができる．

MetricsWebAPI の次なる課題として，エンドユーザのためのクライアントの開発が挙げられる．MetricsWebAPI の提供形態である Web サービスは，XML 形式のテキストベースのプロトコルから呼び出せるため，様々な言語やプラットフォームから利用できる．また，他の Web サービスとの組み合わせ（マッシュアップ）も容易に実現できるため，その拡張が容易といった利点も持つ．これら Web サービスの利点は，主としてアプリ開発者への利益をもたらすものであり，エンドユーザがシステムを利用するためには，別途何かしらのクライアントを設けることが必須である．

この MetricsWebAPI のクライアントとしては，様々な目的のものが考えられる．例えば，ソースコードの成長度合いを可視化し，自己の開発活動の振り返りやプロセス改善を支援するためのクライアントが考えられる．さらに，開発者とソースコードの関係をグラフ化することで，特定ライブラリの利用に関する知識を持った開発者を探すといったコミュニケーション支援のためのクライアントも考えられる．ほかにも，システムの脆弱な部分やリファクタリング箇所の特定のためのクライアントなど，その応用の幅は広い．

本研究では，ソースコードメトリクスの可視化による自己振り返り支援に焦点を絞り，MetricsWebAPI のクライアントとなる Web アプリケーション，**MetricsViewer** を開発する．MetricsViewer は対話的なリポジトリ内のファイル探索機能を提供し，さらに自身の開発行動の俯瞰を助けるためのメトリクス可視化機能を持つ．MetricsViewer を利用することで，マイニングに関する知識を持たないユーザに対しても，気軽な自己振り返りが支援できると考える．可視化処理については Google の提供するグラフ生成のための Web サービス，Google Chart API とのマッシュアップで実現されている．システムの有効性を確かめるための予備実験を行った．実験では被験者 4 名に提案システムを利用してもらい，実際に自己振り返りに役立つかという観点から評価を行う．

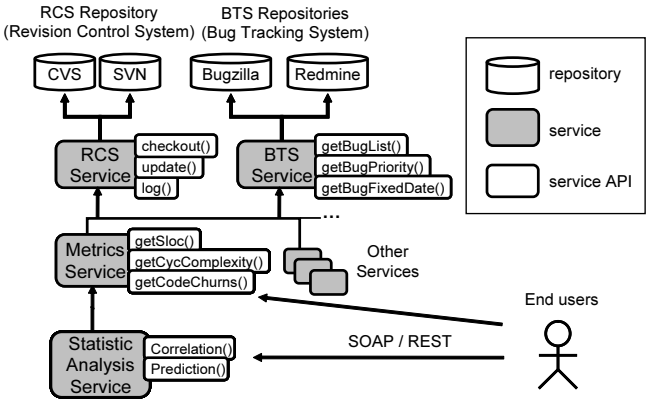

図 1　SO-MSR のアーキテクチャ

2. 準　　備

2.1 ソフトウェアリポジトリマイニング（MSR）

MSR とは，バージョン管理システムやバグ管理システムなどにより管理されるソフトウェアリポジトリから，開発履歴データを取り出し，データマイニングの技術を用いてソフトウェア開発に役立てる行為や研究分野のことを指す．MSR はその分析手段によって様々なものが研究されており，例えば，ソースコードの静的解析 [3], [4] やソースコードのパターンマイニング [5] など多岐にわたる．

MSR の一つの方法として，ソースコードメトリクスを利用した分析手段が数多く研究されている [6], [7]．メトリクスとはソフトウェアの構造や変更履歴から算出された定量的な値のことである．代表的なメトリクスとして，ソースコードの行数や複雑度，CK メトリクス，開発者の貢献度合いやなどが挙げられる．

2.2 先行研究：Service-Oriented MSR（SO-MSR）

SO-MSR とはサービス指向アーキテクチャ（Service-Oriented Architecture）の考えを取り入れた，MSR のためのフレームワークである．図 1 に SO-MSR のアーキテクチャを示す．ここでは版管理システムのリポジトリとして，CVS と SVN の 2 つが例示されている．このリポジトリ種類ごとのアクセス方法の違いは，RCS Service によって吸収される．RCS Service はリポジトリアクセスに関する操作を `checkout()` や `log()` などの API として公開している．バグ管理システムへのアクセスについても同様に BTS Service によってラップされており，`getBugList()` や `getBugFixedDate()` などの API を提供している．Metrics Service はメトリクス計算のためのサービスであり，RCS Service，BTS Service の両方のサービスを利用している．また統計解析のための Static Analysis Service は，この Metrics Service を利用してメトリクス値を取得し統計処理を施す．このように複数のサービスを組み合わせることで，MSR サービスの拡張が柔軟に実現可能である．

2.3 先行研究：MetricsWebAPI

我々は SO-MSR のフレームワークに従ったサービスの一つとして，MetricsWebAPI を開発している．MetricsWebAPI はソフトウェアメトリクス算出のための Web サービスであり，現

```
$curl http://metrics.web.api/registerReposi
tory?repo=http://path.to.repo/

  <id>1</id>
```

(a) registerRepository() API の呼び出し例

```
$curl http://metrics.web.api/getSlocs?repo=
1&src=PlayHNS.java

  <getSlocs>
    <revision>
      <revid>690</revid>
      <date>2010-08-27</date>
      <author>gen</author>
      <sloc>100</sloc>
    </revision>
    <revision>
      <revid>698</revid>
      <date>2010-08-30</date>
      <author>masa-n</author>
      <sloc>112</sloc>
    </revision>
    ...
  </getSlocs>
```

(b) getSlocs() API の呼び出し例

図 2　MetricsWebAPI を cURL コマンドで呼び出した結果の例

在以下に示す 3 種類の代表的なメトリクスを計測できる．

- コードメトリクス: 行数，CK メトリクス，複雑度等．
- 変更メトリクス: 変更行数，リビジョン回数等．
- 開発者メトリクス: 開発者数，開発者リスト等．

MetricsWebAPI を cURL コマンド(注1)を用いて REST プロトコルで呼び出した結果の一例を図 2 に示す．この図は，あるリポジトリ（http://path.to.repo/）に格納されているソースコード（PlayHNS.java）のリビジョンごとの行数一覧を取得する場合の API 呼び出し例を表している．図 2(a) では，MetricsWebAPI へリポジトリを登録する `registerRepository()` を呼び出しており，図 2(b) で，各リビジョンの行数の一覧を取得する `getSlocs()` を呼び出している．図 2(a) での呼び出しパラメタはリポジトリ URL である “http://path.to.repo” であり，その返り値としてリポジトリの登録 ID 番号 “1” が得られている．図 2(b) では先ほどの登録 ID 番号 “1” とソースコードの名前 “PlayHNS.java” が指定され，返り値として各リビジョンのメタ情報と行数が一覧として得られている．

このようにサービスの呼び出しには，XML 形式のテキストベースプロトコル（SOAP/REST）が利用される．そのため，複数のサービス間での連携（マッシュアップ）が比較的容易に実現可能であり，BTS Service と組み合わせたバグ予測サービスなど様々なサービスへの拡張が期待できる．

2.4 課　　題

MetricsWebAPI の課題の一つとして，対話型インタフェースを持ったエンドユーザ向けクライアントの開発が挙げられる．図 2 からも確認できるように，Web サービスを呼び出すための XML 形式の通信プロトコル（SOAP/REST）はマシン間通信のために設けられたプロトコルであり，厳密ではあるものの可読性は低く，ユーザからの直接利用は想定されていない．MSR 知識を持たないエンドユーザに対するメトリクス活用の支援のためには，対話的な操作やメトリクス算出結果の可視化などの機能を持ったクライアントが必須である．

このクライアントはメトリクスの利用目的によって様々なものが考えられる．まず，開発プロセスの把握や改善という観点からは，開発したソースコードのメトリクスを収集し，グラフとして可視化することで，プロセスに関する反省や気づきを直感的に提示できるクライアントが一例として挙げられる．また，ソースコードと開発者の関係をソーシャルグラフとして描画することで，担当開発者を特定したり，特定ライブラリの利用に関する知識を持った開発者を探すといったコミュニケーション支援のためのクライアントも考えられる．ほかにも，システムの脆弱な部分の推定やリファクタリング箇所の特定などの，システム把握や改善に関するクライアントも提供できると考えられる．

3. MetricsViewer

3.1 概　　要

本稿ではメトリクスの活用方法の一つとして，個人の活動の振り返り支援に焦点を絞り，自己振り返りのためのメトリクス可視化システム **MetricsViewer** を開発する．MetricsViewer は MetricsWebAPI のクライアントとなる Web アプリケーションであり，Web ブラウザ上で動作する．本システムは MetricsWebAPI と Google Chart Tools の 2 つのサービスを組み合わせたマッシュアップアプリケーションとして実装されている．マイニング実施者はシステムの利用環境に縛られずどのような環境からでも，対話的な操作を通じて手軽にメトリクスの可視化結果を得ることが可能となる．

3.2 要 件 定 義

提案システムの開発に当たり，ソースコードメトリクスに基づく自己振り返りの支援のための要件を以下の 2 つに定めた．

- **要件 R1**：リポジトリ内の任意のファイルを容易に探し出せる対話的な操作を提供すること．
- **要件 R2**：選択された任意のファイルについて自己振り返りの手助けとなるメトリクスとその他情報の可視化結果を提示すること．

リポジトリ内には数多くのソースコードやドキュメントが保管されており，その中から任意のソースコードを見つけ出すことは容易ではない．ユーザが手軽に自己振り返りを行う際には，要件 R1 のような対話的なファイルエクスプローラ機能が必要となる．また要件 R2 で可視化されるメトリクスとしては，以下の 4 種類を検討している．

- 最新リビジョンのメタ情報（日時，リビジョン ID，コミッター，コメント等）．
- リビジョンの履歴一覧．
- ソースコードメトリクス値の推移グラフ．

(注1)：コマンドラインから URL を呼び出すためのツール．

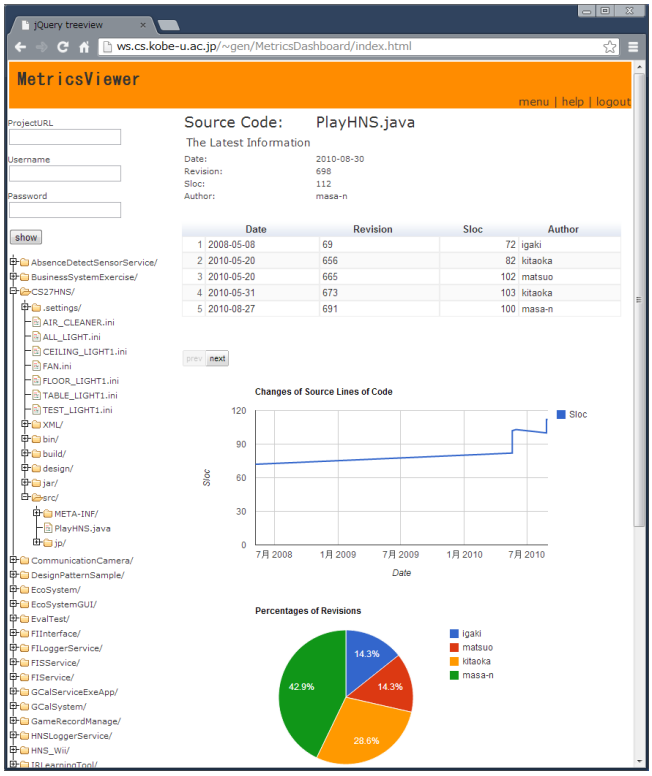

図 3　MetricsViewer のスクリーンショット画面

- ファイルに関与したユーザのコミット数の割合．

自分の問題点や改善点を探し出す際には，活動内容の全体的な情報から異常，あるいは特異な点を探しだし，その詳細をさらに追って確認するという流れが一般的であると考えられる．この一連の振り返りの流れを支援するためには，まず活動全体の俯瞰を助けるようなメトリクスの可視化が必要である．上記の 4 種類のメトリクスは，開発活動に関する情報の中でも直感的に理解しやすい単純な情報である．これらの情報を一つの画面に提示することで自己振り返りの支援を試みる．より具体的には，マイニング対象のソースコードの成長過程をメトリクスの推移グラフによって確認し，個々のリビジョンの詳細をリビジョン履歴情報から確認することができる．

3.3　主 要 機 能

MetricsViewer のスクリーンショットを図 3 に示す．画面左部分が要件 R1 に対応するファイル探索のための機能であり，画面右部分が要件 R2 に対応するメトリクスの可視化とその他情報の提示を行う部分である．MetricsViewer の持つ 5 つの主要機能を次節以降で説明する．

3.3.1　Package Explorer

指定されたリポジトリ内のファイル探索機能を提供する．ユーザがマイニング対象のリポジトリを選択すると，自動的にリポジトリ内のファイル一覧を取得し，エクスプローラビューを生成する．ユーザはこのエクスプローラからファイルを選択することで，対話的に次節以降の可視化機能を利用できる．新規リポジトリを登録する際には，左上部に対象リポジトリの URL とリポジトリのユーザ ID，及びパスワードを指定すればよい．これらの機能は MetricsWebAPI の提供する，`lsr()` と

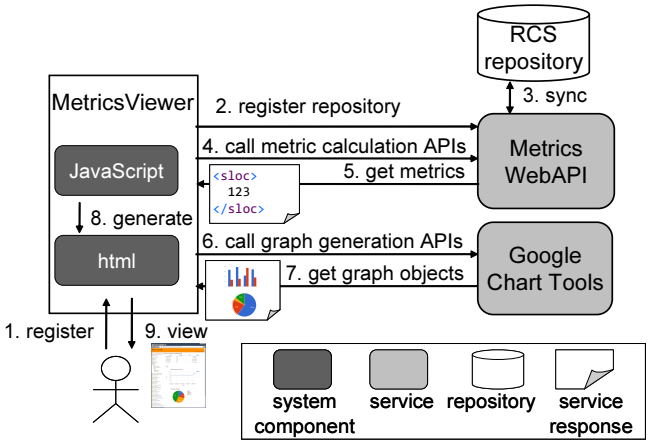

図 4　MetricsViewer の処理の流れ

`registerRepository()` の 2 つの API を利用している．

3.3.2　Latest Information

指定されたファイルに関する，最新リビジョンのメタ情報（リビジョン番号，日付，行数，コミット者）を表示する．この機能により，指定ファイルの最新状態を把握することができる．上記のメタ情報取得には，`getLatestRevision()` と `getSloc()` を利用している．

3.3.3　Revision History

指定されたファイルに関する，リビジョン履歴の一覧を表形式で表示する．本機能により，メトリクスの可視化グラフでは正確に読み取りにくい詳細な情報を確認することができる．利用 API は `getRevisions()` である．

3.3.4　SLOC Line Chart

指定されたファイルに関する，コード行数の推移を折れ線グラフとして可視化する．このグラフは縦軸が行数，横軸が日付となっている．グラフを確認することで，対象のソースコードがどのように成長しているかを直感的に把握できる．行数メトリクスの取得には `getSlocs()` を利用している．この機能は，将来的に行数以外のメトリクスに対応できるよう拡張する予定である．

3.3.5　Contributor Pie Chart

指定されたファイルに対する追記・削除・編集に関わったユーザの，貢献度の割合を円グラフを用いて表示する．この貢献度はコミット回数の割合から算出されている．共同作業したユーザの貢献度の割合を直感的に把握できる．`getContributors()` を利用している．

3.4　可視化までの処理の流れ

MetricsViewer による処理の流れを図 4 に示す．

Step1: ユーザは対象のリポジトリを登録する．

Step2: JavaScript によりリポジトリ登録 API が実行される．

Step3: MetricsWebAPI は指定リポジトリをサーバ上に同期する．

Step4: JavaScript によりメトリクス計測 API が実行される．

Step5: MetricsWebAPI は同期したリポジトリから対象ソースコードを取り出し，メトリクスを計測し返す．

Step6: JavaScript により Google Chart Tools のグラフ生

質問1. 以下の3つの点に対して4段階で評価してください.
| 4:とても満足 | 3:満足 | 2:不満 | 1:とても不満 |
・使いやすさ [4 3 2 1]
・情報の見やすさ [4 3 2 1]
・実行時間 [4 3 2 1]

質問2. 執筆活動振り返りで得られた「自己発見」を記述してください.
質問3. 執筆活動に関する「改善点や反省点」を記述してください.
質問4. システムに対する「不満や改善点」を記述してください.
質問5. その他気づいた点について記述してください.

図 5 アンケートの内容

成 API を呼び出す.

Step7: Google Chart Tools から生成されたグラフオブジェクトを取得する.

Step8: グラフとその他メタ情報をまとめた HTML が生成される.

Step9: 生成された HTML がブラウザ上に描画される.

3.5 実 装

HTML と CSS, 及び JavaSciprt を利用して MetricsViewer を実装した. 主な処理については jQuery を利用して実現されている. 開発工数には約 2 人月を要した. プログラムステップ数は 430 行である.

4. 評 価 実 験

4.1 概 要

MetricsViewer が実際に自己振り返りに役に立つかを確かめるために, 被験者に提案システムを利用してもらう予備実験を行った. 実験の題材には, ソフトウェアの開発活動ではなく, 研究会原稿の執筆活動を取り上げた. その理由は, ソフトウェア開発活動の履歴データと被験者を確保できなかったためである. しかしながら, 提案システムによる自己振り返りの効果を確認するという観点では, ソフトウェア開発のデータに限定する必要はなく, 原稿の執筆履歴 (正確には tex ファイルの変更の推移) でも十分であると考えられる.

被験者は大学院修士の学生 4 名である. 実験の流れとしては, まず被験者が初めて執筆した研究会 (全被験者学部 4 年生の時) の原稿の tex ファイルを, 提案システムを利用して可視化してもらう. 次に, 最も直近の執筆原稿の tex ファイルを同様に可視化してもらう. 最後に, これら 2 つの可視化結果を比較し, 自己の執筆活動の振り返りに関するアンケートに答えてもらった.

アンケートの内容を図 5 に示す. アンケート内容には使いやすさ, 情報の見やすさ, 実行時間に関する定量的な評価項目と, 自己発見や改善点などの自己振り返りに関する質問と, 提案システムの不満点などの質問が含まれている.

4.2 結 果

アンケートの結果を図 6 に, 可視化結果の一例としてある被験者の最新の執筆活動の可視化結果を図 7 示す. 自己発見に関する質問 2 からは, ほとんどの被験者から「執筆間際の追い込みが激しい」という点が挙げられた. また, 自己の改善点・反省点については,「余裕を持って執筆に取り組めるよう計画を立

質問1:4段階評価
・使いやすさ 平均3.5点
・情報の見やすさ 平均3.3点
・実行時間 平均2.3点

質問2:自己発見
・直近の執筆活動は締め切り間際の追い込みが激しい.
・執筆を助けてもらったことを忘れていたが思い出せた.
・初めての執筆活動は時間に余裕を持って進めていた.
・全体を先に執筆し終えてから, 細かな部分を調整するスタイルであることに気付いた.

質問3:自己の改善点・反省点
・余裕を持って執筆できるように計画を立てておくべき.
・直近の執筆活動を振り返ったところ, 締め切り間際でもなんとかなるだろうという甘えが見られた.
・振り返りという観点ではこまめにコミットを行うべきであった. その時の考えや悩みなどを正確に記録しておけば過去の把握や問題点の洗い出しに役立ちそう.

質問4:システムの不満・改善点
・同時に複数ファイルを触りたい.
・ファイル単位ではなく人単位で可視化してほしい.
・過去の振り返りだけでなく予定との比較などの機能が欲しい.
・各リビジョンのコメントも併記してほしい.

質問5:その他の意見
・自身の執筆活動が直感的に可視化されていて面白い.

図 6 アンケート結果

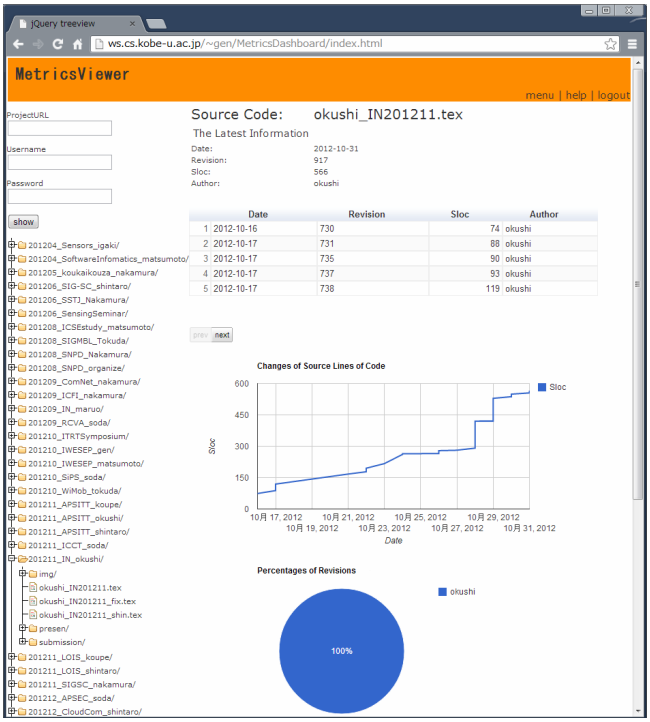

図 7 可視化結果の一例 (ある被験者の最新の執筆活動の可視化結果)

ておくべき」という意見や,「締め切りに対する甘えが見つかった」などの意見が得られた. システムに対する不満としては,「同時に複数のファイルを比較したい」,「ファイル単位ではなく人単位での活動の振り返りを行いたい」などの意見が得られた. その他の意見からは, 振り返りという行為そのものに対して面白い・興味深いという意見も得られた.

5. 考　　察

5.1 提案システムの有用性

定量的な質問項目からは，使いやすさや見やすさについては平均 3.5 点と 3.3 点であり，概ね高評価が得られた．また，その他の意見からも自己振り返りという行為そのものに関して面白いという意見も得られた．また定性的なアンケートからも，実際にいくつかの自己発見や改善点・反省点の発見があった．これらのことから，提案システムの有用性が確認できたと考えられる．

さらに，ほとんどの被験者から直近の執筆活動の方が締め切り間際の追い込みが激しいという意見が得られた．特に図 7 で示したある被験者については，締め切りまでまだ余裕があるという，執筆活動に対する甘えが確認できるという意見があった．研究室（あるいはチーム）単位での作業という観点からは，余裕を持って執筆できるよう事前にスケジュールを立てるべきであるという改善点が発見出来たといえる．このように自己振り返りに限らず，チーム単位での振り返りにも役立てられると考えられる．

5.2 課　　題

質問 1 での評価として，実行時間に対する不満が多かった．その理由は，メトリクス算出 API が呼び出される度に，リポジトリに対する Update 処理とメトリクス算出処理が実行されており，またその処理結果の再利用も一切されていないためである．この点については，提案システムそのものではなく MetricsWebAPI へのキャッシュ等の改善が必要であると考えられる．

振り返りの支援という観点からは，各リビジョンの変更理由を確認するためにコミットのコメントも表示してほしい，複数のファイルのメトリクスを同時に可視化・比較してほしい，といった提案システムへの改善点が得られた．これらの機能はすでに MetricsWebAPI 側で提供されているため，比較的容易に拡張可能である．

現在はファイル単位の可視化のみを支援しているが，自己の振り返りという観点からは開発者単位のメトリクスの可視化が必須である．例えば指定した開発者についての，編集ファイルの一覧や，共同開発者の一覧，その開発者のコミット数といった情報を可視化することで，より効果的な自己振り返り支援ができると考える．現在の MetricsWebAPI で提供される API は，基本的にファイルを主体とした情報へのアクセスのみであるため，この開発者単位のメトリクス収集機能に関しては支援できていない．開発者単位のメトリクス可視化のためには，SO-MSR のフレームワークに従い，開発者単位でのメトリクスを収集可能な新たなサービスを開発する必要がある．

6. ま と め

本稿では，個人の振り返り支援を目的としたメトリクス可視化 Web アプリケーション MetricsViewer を開発した．MetricsViewer を利用することで，MSR の専門的な知識がないユーザでも，容易にリポジトリデータに基づいた振り返りを実施可能となる．4 名の被験者による予備実験により，提案システムを用いることで自身の改善点や反省点などを発見できることを確認した．

今後の課題は，まずアンケート結果から得られたシステムの不足機能の追加が挙げられる．コミット時のコメントの表示機能や，複数ファイルの比較機能，ファイル単位ではなく人単位での活動履歴の可視化機能を追加・拡張することで，より自己振り返りに役立てられる．さらに，全体の俯瞰を助ける方法としては，パッケージ単位やリポジトリ単位の可視化が必要であり，俯瞰結果からより細かな情報を掘り下げる方法として，メトリクスだけに限らずコード自体の中身を提示するといった機能も検討中である．

謝辞

この研究の一部は，科学技術研究費（基盤研究 C 24500079, 基盤研究 B 23300009），および，関西エネルギー・リサイクル科学研究振興財団の助成を受けて行われている．

文　　献

[1] S. Matsumoto and M. Nakamura, “Service oriented framework for mining software repository,” Proceedings of the Joint Conference of the 21st International Workshop on Software Measurement (IWSM) and the 6th International Conference on Software Process and Product Measurement (Mensura), pp.13–19, 2011.

[2] E. Cerami, Web Services Essentials, O’Reilly, 2002.

[3] C.C. Williams and J.K. Hollingsworth, “Automatic mining of source code repositories to improve bug finding techniques,” IEEE Transactions on Software Engineering, vol.31, no.6, pp.466–480, 2005.

[4] R.W. Selby, “Enabling reuse-based software development of large-scale systems,” IEEE Transactions on Software Engineering, vol.31, pp.495–510, 2005.

[5] H. Kagdi, Y. S, and J.I. Maletic, “Mining sequences of changed-files from version histories,” Proceedings of the 3rd International Workshop on Mining Software Repositories (MSR ’06), pp.47–53, 2006.

[6] A.P. Nikora and J.C. Munson, “Understanding the nature of software evolution,” Proceedings of the 19th International Conference on Software Maintenance (ICSM ’03), pp.83–93, 2003.

[7] A.G. Koru, D. Zhang, K.E. Emam, and H. Liu, “An investigation into the functional form of the size-defect relationship for software modules,” IEEE Transaction on Software Engineering, vol.35, no.2, pp.293–304, 2009.