Vol.2018-CE-143 No.26
2018/2/18

Eclipse のローカルヒストリーを用いた持ち帰り課題の取り組み状況分析ツール

高橋圭一[†1]

概要：オープンソースソフトウェアの統合開発環境の 1 つである Eclipse には編集履歴を自動的に保存するローカルヒストリー機能が標準で搭載されている．本稿では，このローカルヒストリー機能を実現するために記録されるファイル群を収集し，統合した編集履歴情報を利用することで，演習時間外の受講生の課題の取り組み状況を分析するツールを提案する．

キーワード：プログラミング課題，類似レポート抽出，統合開発環境

A Tool for Analyzing the Process of Making Assignments outside Lectures using Local History of Eclipse

KEIICHI TAKAHASHI[†1]

Abstract: Eclipse, open-source integrated development environment (IDE), has a function called local history that Eclipse automatically saves the edited files in its workspace at every execution. In this paper, we propose a tool for analyzing the process of making assignments outside lectures by using local history of Eclipse.

Keywords: Programming Assignments, Finding Similar Reports, Integrated Development Environment

1. はじめに

高等教育機関のプログラミング演習科目では，受講生は講師が出題した課題に対応したプログラムを作成し，テストコードや実行結果を添えてレポートを提出することが求められる. 演習時間内に完了できなかった課題については，受講生が演習時間外に時間を確保してレポートを仕上げる必要がある．これまで，演習時間内において受講生の進捗状況を把握して支援につなげる研究が多数行われている[1-4]．一方で，演習時間内に完了できなかった課題の取り組み状況を把握し分析する研究は我々の知る限りない．また，演習時間外に着手して提出されたレポートの中には，類似するプログラムコードが含まれる場合があり，適切な評価が難しい場合がある．そこで本稿では，Eclipse を用いたプログラミング演習科目を対象として，受講生が演習時間外にコンピュータ室や自宅のパソコンでプログラミング課題に取り組む状況を把握し，特に類似するレポートを抽出することに着目して，プログラムコードの作成過程の分析作業を支援するツールを提案する．

2. 分析ツール

2.1 分析対象とするプログラミング言語と開発環境

多くの高等教育機関のプログラミング科目でも採用されていることからプログラミング言語は Java とする．分析ツールは，自宅等に持ち帰ってプログラミング課題に取り組むことを考えると，学内のコンピュータ室のパソコンはもちろん，それとは異なる環境でも動作する必要がある．文献[1-3]のように，ブラウザベースの開発環境を構築することは有効であるが，Web アプリケーション開発のように多言語を用いた開発やデータベースや他のサーバーとの連携の必要性を考えると，個々の科目に対応した開発環境を準備することは手間がかかり容易ではない．Java には様々な統合開発環境があるが，本研究では，インストールした後に特別な設定や操作を必要とせず，プログラムコードの編集履歴が記録される Eclipse を採用する．

2.2 Eclipse のローカルヒストリー機能

Eclipse にはローカルヒストリーと呼ばれる編集履歴を保存し管理する機能が標準で搭載されている．利用者がプログラムコードを編集し保存操作を行うたびに，ワークスペースのプロジェクト内にある各ファイルの編集履歴が.metadata/.plugins/org.eclipse.core.resources/.history 配下に自動的に蓄積される．Eclipse の初期状態では，編集履歴の保存期間は 7 日間であり，プログラミング演習科目が毎週開講されることを考えると，持ち帰り課題の提出期限を踏まえた保存期間として十分である．

2.3 LocalHistory2File

編集履歴を収集し必要な情報を統合するツールを LocalHistory2File と呼ぶ．LocalHistory2File が生成するデー

†1 近畿大学産業理工学部情報学科
Kindai University

タ構造を概略クラス図で示す（図 1）．Eclipse のワークスペース内の各プロジェクトには，開発するアプリケーションの構成要素である HTML ファイルや Java ファイルなどの複数のファイル（LocalHistory）がある．それぞれのファイルに編集履歴ファイル（ALocalHistory）があり，さらに課題に取り組むときに利用したパソコンのホスト名およびログインしたユーザ名を加えたデータ（LocalHistoryData）がある．LocalHistory2File はこれら全てのインスタンスをシリアライズ化した単一のファイルを生成する．このファイルを本稿では LH ファイルと呼ぶことにする．

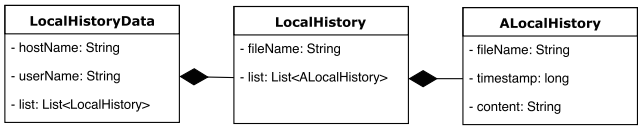

図 1　LH ファイルのデータ構造

次に LocalHistory2File の外観を示す（図 2）．本ツールを起動後，ワークスペースが存在するパス名を入力すると，ワークスペースに含まれるプロジェクト名の一覧が表示されるので，その中から演習時間内に作成したプロジェクト名を選択し，ファイル作成ボタンを押下すると LH ファイルが生成される．本ツールは Java 実行環境（JRE）が導入済みであれば自宅のパソコンでも使用できる．この LocalHistory2File を受講生に配布し，受講生がレポートを提出するときに本ツールを実行し，生成された LH ファイルとレポートを合わせて講師に提出する．

図 2 LocalHistory2File の外観

2.4 LocalHistoryAnalyzer

LH ファイルを分析するツールを LocalHistoryAnalyzer と呼ぶ．課題の取り組み状況を分析し，特に類似するプログラムコードを発見する作業を支援するため，LocalHistoryAnalyzer で出力する分析項目を次の 4 つとする．

プログラムコード類似度（以降，類似度と略す）：類似するプログラムコードを機械的に抽出するため類似度を計算する．計算手順としては，提出されたファイル群から 2 つの LH ファイルを選択し，共通するファイル名が含まれていたら，その編集履歴から最終版を取り出し類似度を計算する．この手順を繰り返して全てのファイル間の類似度を求める．類似度にはレーベンシュタイン距離を用いる．

編集履歴タイムライン：2 人の受講生が一緒に，あるいは別々に課題に取り組んだのか把握するために，2 人の編集履歴の発生時刻を上下段にわけて縦線で表す．編集履歴の発生時刻は ALocalHisotry クラス（図 1）の timestamp から得る．タイムラインの範囲は当該回の演習授業の開始日時を左端とし，そこから 1 週間経過した日を右端とする．上下段の縦線間の距離が近ければ 2 人が一緒に作業したと考えられ，離れていれば着手した時刻の前後関係がわかる．

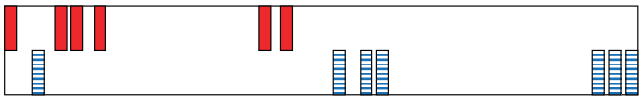

図 3 編集履歴タイムラインの表示例

編集回数：受講生がプログラムコードを編集し保存操作を行うたびに編集履歴が生成されるため，編集履歴の個数は編集回数と考えられる．また，プログラム実行前に修正したプログラムコードを保存する必要があるため，編集履歴の個数はデバッグ実行の試行回数とも考えられる．編集回数は, 図 1 の LocalHisotry クラスの list の要素数から得る．

座席距離：学内のコンピュータ室のパソコンを利用した場合，2 人の座席が近ければ一緒に課題に着手した可能性が高い．LocalHisotryData クラスの hostName は OS の環境変数であるホスト名であるため，コンピュータ室内の座席を特定できる．受講生間の相談のしやすさを考慮して，図 4 のように 2 次元平面座標に論理的に座席配置を決定し，座席間のユークリッド距離を座席距離とする．共同作業がしやすい横（例：PC3 と PC4）や背中合わせ（例：PC3 と PC5）に隣接する座席間の距離を 1 とし，向い合わせ（例：PC3 と PC1）の距離は 2 とする．ホスト名がコンピュータ室のパソコン名に一致しない場合は，座席距離として最大値（1000）を与える．

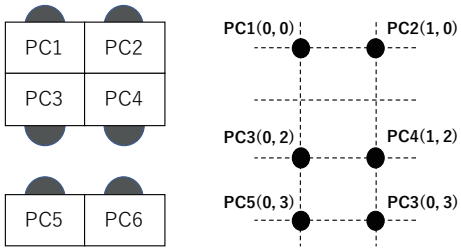

図 4 演習室の座席配置とその座標

2.5 LocalHistoryAnalyzer が生成するビュー

LocalHistoryAnalyzer を実行すると，2.4 に示した 4 つの分析項目を計算した後，類似受講生関係ビュー（図 5）および類似プログラムコード比較ビュー（図 6）を生成する．LocalHistoryAnalyzer は，全ての受講生間の類似度を計算するため，全ての組み合わせの情報をビューに表示することも可能であるが，表示件数が膨大となると分析作業が困難になる．そこで，類似するレポートを発見することを考慮し, 類似度が 0.95 以上のペアのみ表示する. この閾値（0.95）は，一見して見分けのつかないレポートを筆者が抽出して決定した．

類似受講生関係ビュー：類似度の高い受講生同士を結び，線上に類似度を表した図である．図の生成には Graphviz を

用いる[5]．図 5 の例では，受講生 A と受講生 C，受講生 B と受講生 C のプログラムコードの類似度がそれぞれ高い（≥ 0.95）ことを表している．ここで受講生 A と受講生 B の類似度は高くない（0.95 未満）であることに注意されたい．また，受講生 Xa と受講生 Xb の類似度が 1.0 であるため，両者のプログラムコードの内容が同一であることを表している．

図 5 類似受講生関係ビューの例

類似プログラムコード比較ビュー：編集履歴タイムラインを最上段に表示し，類似度，2 人の受講生の編集回数，座席距離およびプログラムコードを表示する（図 6）．図 6 を用いて見方を説明する．編集履歴タイムラインより，受講生 B が先に着手して完成させ（下段），しばらくして受講生 A が着手し完成させていることがわかる（上段）．座席距離が 1.0 であるため両者が横隣のパソコンを利用しているが，着手時刻に差があるため別々に作業した可能性がある．一方，類似度は 1.0 とプログラムコードが同一であることから受講生 A が受講生 B のプログラムコードを参照した可能性が考えられる．編集回数は同程度であるため，この情報のみで両者のプログラムコードの参照関係を判断することは困難である．

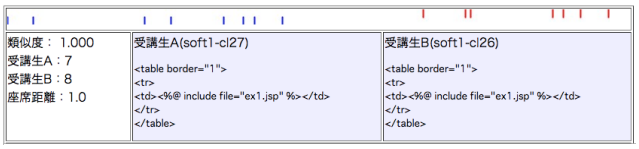

図 6 類似プログラムコード比較ビューの例

3. 実験

本稿の主題である「演習時間外の課題の取り組み状況の把握」と「類似するプログラムコードの作成過程の分析」に関して，提案手法の有効性を評価するため以下の実験を行った．

3.1 実験方法

本学科の 3 年次向けの開講科目に Java を用いた Web アプリケーション開発の演習科目がある．本科目の第 7 回（2017 年 5 月 26 日実施）の演習において，受講生に LocalHistory2File を配布し，レポート提出直前に LocalHistory2File を実行して生成した LH ファイルをレポートと共に指定した Moodle サーバに提出させた．課題提出の猶予期間は 1 週間である．なお，本科目は筆者が単独で担当する科目である．第 7 回の演習課題は 3 問あり，それぞれにおいて HTML と JSP（Java Server Pages）を 1 つずつ作成し，Web ページに入力されたデータをファイルに格納し，そのデータを一覧表示する Web アプリケーションを完成する必要がある．作成するプログラムコードのファイル名は問題文で指定している．

3.2 実験結果

受講生が提出した LH ファイルをもとに分析した結果を以下に示す．

3.2.1 課題提出状況

期限内にレポート課題を提出した受講生は 49 名であった．また，提出された LH ファイルの総数は 54 であった．これは学内のパソコンの他に，自宅のパソコンで取り組んだ LH ファイルも合わせて提出したためである．したがって，5 名（= 54 - 49）が自宅で課題に取り組んだことがわかる．LH ファイルに含まれるプログラムコード別の提出人数を表 1 に示す．演習 1 はほぼ全員が提出したが，演習 3 は約 10 名が着手さえできなかった．また，演習課題の問題文でファイル名が指定されているにも関わらず，異なるファイル名があったため，その人数を「その他」とした．その他の提出情報は以降の分析では除外する．

表 1 プログラムコード別の課題提出人数

	ファイル名	提出人数
演習 1	ex1.html	48
	ex1.jsp	47
演習 2	ex2.html	44
	ex2.jsp	43
演習 3	ex3.html	39
	ex3.jsp	39
その他		20

3.2.2 課題に着手した日時

受講生がいつ課題に着手したのかを調べるため，LH ファイルの timestamp（図 1 参照）を第 7 回の演習開始日時を原点として，編集回数の分布を図 7 に示す（N=4840）．

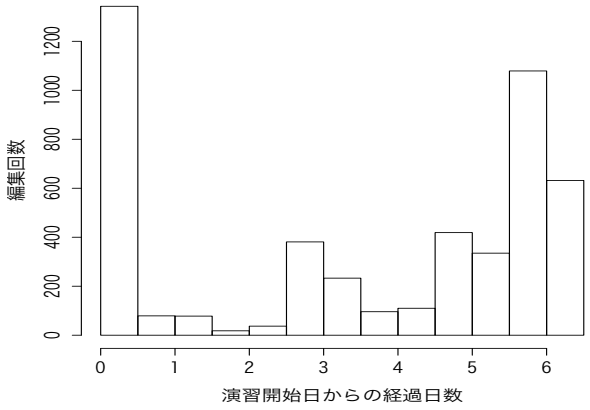

図 7 提出期間内の編集回数の分布（N=4840）

単位時間当たりの編集履歴が最も多いのは最左の階級であり，つまり演習時間内の編集履歴であった．また，演習開始 3 日後と提出期限前日に多くの受講生が課題に着手し

ている．演習時間内を表す最左の階級は 1343 であることから，演習時間外の編集履歴は全体の 72%（= 1 – 1343/4840）を占めることがわかった．

受講生が演習時間外に課題に着手した日内の時刻を調べるため，図 7 のデータから演習時間内（最左の階級）を除外したデータをもとに着手時刻の分布を調べた（図 8）．15 時から 18 時の時間帯に多くの受講生が課題に着手している．

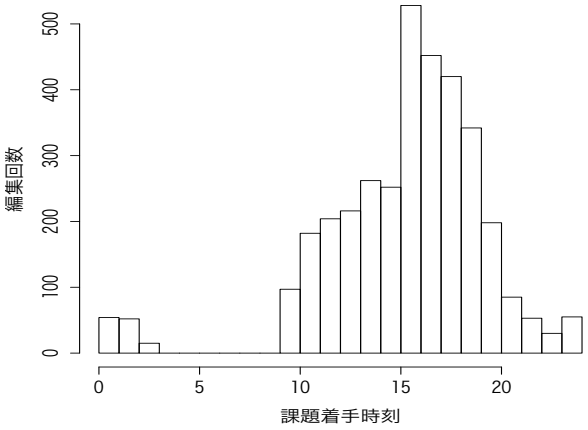

図 8 提出期間内の日内の編集回数の分布（N=3497）

3.2.3 プログラムコードの類似度

プログラムコードの最終版同士の類似度の分布を図 9 に示す（N=4544）．2.5 で述べたように，一見して見分けのつかないレポートの類似度を 0.95 とすると，図 9 の最右の階級が該当し，その編集回数は 37 であった．つまり，プログラムコードの全てのペアのうち 0.8%（= 37/4544）が，編集履歴に含まれる情報を用いて，さらに分析する必要があることがわかった．この 0.8%のプログラムコードのペアを対象として，分析ツール LocalHistoryAnalyzer の利用例を次章に示す．

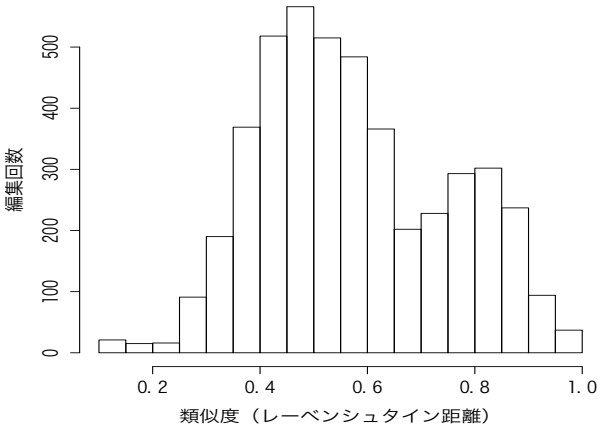

図 9 プログラムコードの最終版同士の類似度の分布（N=4544）

4. LocalHistoryAnalyzer の利用例

3 章の実験で得られた LH ファイルを対象として，LocalHistoryAnalyzer を利用して類似するプログラムコードを発見し，詳細を分析する事例を紹介する．

ステップ 1：まず，演習課題ごとに生成された類似受講生関係ビューを参照することで，類似するプログラムコードの有無を確認する．ex2.jsp の類似受講生関係ビューを図 10 に示す．まず，類似度が 1.0 である受講生 A と受講生 X の状況を類似プログラムコード比較ビューで確認する．

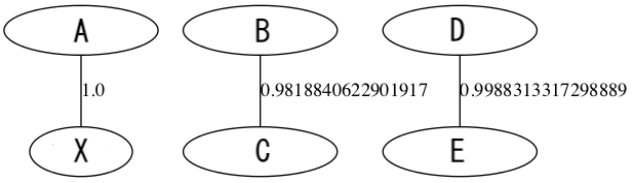

図 10 ex2.jsp の類似受講生関係ビュー

ステップ 2：受講生 A と受講生 X の類似プログラムコード比較ビューを図 11 に示す．類似度が 1.0 であるため 2 つのプログラムコードは同一である．受講生 X のユーザ名から両者が同一人物であると推察できた．受講生 A が演習時間内に作成したプログラムコードを持ち帰り，自宅のパソコンに受講生 X としてログインして作業を継続したため，類似度やその他の属性が同一になったと考えられる．

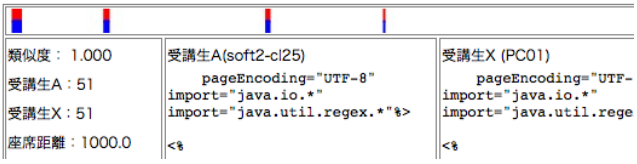

図 11 ex2.jsp の受講生 A と受講生 X の類似プログラムコード比較ビュー

ステップ 3：図 10 において，2 番目に類似度が高い受講生 D と受講生 E の類似プログラムコード比較ビューを調べる（図 12）．編集履歴タイムラインより受講生 D が先に課題に着手し（上段），時間をおいて受講生 E が課題に着手している（下段）．受講生 D の編集回数は受講生 E と比べてかなり多い．両者の ex2.jsp は約 100 行あるが，両者が独立して作業した場合，この規模のプログラムコードの内容がほぼ同一になる確率は非常に低く，編集回数にこれだけの差がつくとは考えにくい．

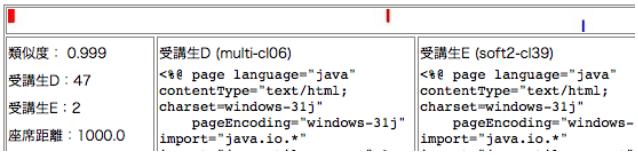

図 12 ex2.jsp の受講生 D と受講生 E の類似プログラムコード比較ビュー

ステップ 4：受講生 D と受講生 E の関係をさらに確認するため，他のプログラムコードを確認すると，ex3.jsp の類似プログラムコード比較ビューに両者のユーザ名を発見した

（図 13）．課題に着手した前後関係および編集回数の差異は ex2.jsp と同様の傾向であった．複数のプログラムコードにおいて，類似度や課題着手の前後関係などが同様の結果になる確率は非常に低いことから，受講生 E が受講生 D のプログラムコードを参照して課題を提出した可能性が高い．

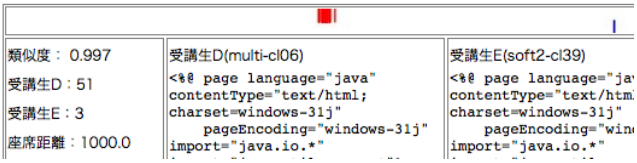

図 13 ex3.jsp の受講生 D と受講生 E の類似プログラムコード比較ビュー

5. 考察

(1) 持ち帰り課題の取り組み状況

図 7 に示したように，演習授業日の 3 日後と提出期限前日に多くの学生が課題に取り組んでいた．演習授業日は金曜日であったため，その 3 日後は月曜日になる．また，図 8 より課題に着手した時刻は 15 時から 18 時の間が最も多かった．本学科の学生を対象として入学時に行ったアンケートによると 74.6%の学生がパソコンを所有していた．この状況を考えると，週中の帰宅後や休日に自宅で課題に取り組む学生がある程度はいると想像したが，実験結果を見る限り，授業の合間に学内のパソコンを利用して課題に取り組む受講生が多いことがわかった．

(2) プログラムコードの類似度

プログラミング課題の場合，問題文は全ての受講生で共通するため，解答となるプログラムコードは類似する傾向がある．実験結果より，プログラムコードの類似度は広く分布しており，一見して見分けがつかないほど似ているプログラムコードは全体の 0.8%と僅かであった（図 9）．課題の提出期間中に数人のグループで課題に取り組む様子をたびたび見かけた．受講生同士で教え合うことは双方にとって良い効果があるが，結果としてプログラムコードが似ることは止むを得ないことと考える．本稿では類似度が低いプログラムコードは分析していないが，実験結果の類似度のばらつきを見る限り，教え合いによる類似度への影響は少ないと考える．

(3) 類似するプログラムコードの作成過程

4 章で LocalHistoryAnalyzer を用いた類似するプログラムコードの分析過程の一例を示した．レポートに記載されるプログラムコードは最終版のみのため，一見して見分けがつかないほど似ているレポートの評価は困難であった．この問題に対して LocalHistoryAnalyzer を利用することで，類似するプログラムコードの編集日時や編集回数を知ることができ，分析に役立つことがわかった．個人差は多少あるとしても，同一規模のプログラムコードを完成するまでに要する編集回数に極端な差が出ることは考えにくい．そのため，類似するプログラムコードの編集日時の前後関係と編集回数の大小関係から，受講生間の参照関係をある程度判断できる．また，それらペアの他のプログラムコードが類似していれば，受講生間の参照関係の判断の信頼性をさらに向上できる．実験結果には受講生間の着手時刻や編集回数の関係が逆転するケースも存在したが，このように他の課題のプログラムコードにも同様の傾向があるか確認することで精度向上が期待できる．

(4) ツールとしての扱い易さ

LocalHistory2File が収集する情報は Eclipse のローカルヒストリー機能を実現するために蓄積された編集履歴情報を利用する．この方式を採用することで，プラグインの追加や特別な設定を必要とせず，Eclipse が導入されていれば課題の取り組み状況を漏れなく取得できる．もし追加のソフトウェアや特別な設定を必要とすると，操作や設定忘れのため編集履歴の蓄積ができない場合が発生してしまう．特に学外のパソコンでのツールの使用を考えた場合，こうした前提条件が少ないことは，受講生・講師の双方にとって重要な要素と考える．本稿の実験ではレポートを提出した 49 名の受講生のうち 5 名が自宅のパソコンを利用して LH ファイルを提出したが障害報告は受けていない．

6. まとめ

本稿では，演習時間外に受講生の課題の取り組み状況を把握するため，Eclipse のローカルヒストリー機能の情報を利用した 2 つのツールを提案した．実験により，持ち帰り課題の着手日時や編集回数などを把握できることが示された．また，LocalHistoryAnalyzer を利用することで，一見して見分けがつかないほど似ているレポートのプログラムコードの生成過程を追跡し，編集日時や編集回数の関係から受講生間の参照関係を分析することができた．

現状の類似受講生関係ビューと類似プログラムコードは独立しているため，類似プログラムコードの発見や絞り込み作業が煩雑である．これらビューを統合することで分析作業の負担を軽減することを考えている．また，本稿では類似するプログラムコードの分析に注目したが，LocalHistory2File で収集する編集履歴を活用することで，持ち帰り課題の躓き箇所やその要因の特定にも役立てることができる．今後，このような支援システムの開発も進めていきたい．

参考文献

1) 井垣宏，齊藤俊，井上亮文，中村亮太，楠本真二．プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案．情報処理学会論文誌．2013, vol.

54, no. 1, p. 330-339.
2) 加藤利康, 石川孝. プログラミング演習のための授業支援システムにおける学習状況把握機能の実現. 情報処理学会論文誌, 2014, vol.55, no.8, p.1918-1930
3) 蜂巣吉成, 吉田敦, 阿草清滋. プログラミング演習におけるコーディング状況把握方法の考察. 研究報告コンピュータと教育（CE）. 2014, 2014-CE-125(3), p. 1-8.
4) J. Spacco, P. Denny, B. Richards, D. Babcock, D. Hovemeyer, J. Moscola, and R. Duvall. Analyzing student work patterns using programming exercise data. In Proc. SIGCSE '15. 2015, p. 18-23.
5) "Graphviz". https://graphviz.gitlab.io/, (参照 2018-1-5).