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.

コードクローン保守支援を目的とした変更履歴可視化システム

本田　紘貴† 　徳井　翔梧† 　横井　一輝† 　崔　　恩瀞†† 　吉田　則裕†††
井上　克郎†

† 大阪大学大学院情報科学研究科　〒565–0871 大阪府吹田市山田丘 1-5
†† 奈良先端科学技術大学院大学先端科学技術研究科　〒630–0192 奈良県生駒市高山町 8916-5
††† 名古屋大学大学院情報学研究科　〒464–8601 愛知県名古屋市千種区不老町
E-mail: †{h-honda,s-tokui,k-yokoi,inoue}@ist.osaka-u.ac.jp, ††choi@is.naist.jp, †††yoshida@ertl.jp

あらまし　コードクローン変更管理システム Clone Notifier は，2 バージョン間で行われたコードクローンの追加，編集，削除といった変更履歴情報を開発者に提供する．しかし，Clone Notifier は 2 バージョン間の変更履歴情報のみを提供するため，過去から現在にわたるコードクローンの変更履歴全体を分析することは困難である．本研究では，任意の隣接するバージョン間のクローンセットに関する統計データを集計し，変更履歴全体を可視化するコードクローン変更履歴可視化システムの開発を行った．本システムにより，開発者のコードクローンの分析と保守を支援する．
キーワード　コードクローン，ソフトウェア保守

A system for visualizing clone evolution to support clone maintenance

Hirotaka HONDA†, Shogo TOKUI†, Kazuki YOKOI†, Eunjong CHOI††, Norihiro YOSHIDA†††,
and Katsuro INOUE†

† Graduate School of Information Science and Technology, Osaka University　1-5, Yamadaoka, Suita, Osaka, 565-0871, Japan
†† Graduate School of Science and Technology，Nara Institute of Science and Technology　8916–5 Takayama, Ikoma, Nara, 630–0192, Japan
††† Graduate School of Informatics, Nagoya University　Furo-cho, Chikusa-ku, Nagoya, Aichi 464–8601, Japan
E-mail: †{h-honda,s-tokui,k-yokoi,inoue}@ist.osaka-u.ac.jp, ††choi@is.naist.jp, †††yoshida@ertl.jp

Abstract　Clone Notifier, a code clone change management system, notifies developers about the evolution of code clones such as adding, editing and deleting code clones between two versions. However, it is difficult to analyze the entire evolution of code clones in Clone Notifier because it only informs the evolution of code clones between two versions. To tackle this problem, in this study, we developed a system for visualizing the evolution of code clones to support code clone maintenance. This system aggregates statistical data about the clone sets between arbitrary adjacent versions and then visualizes the entire evolution of code clones from the past to the present.
Key words　code clone, software maintenance

1. はじめに

コードクローンとは，ソースコード中に含まれる互いに一致または類似した部分を持つコード片のことである [1]．一般的に，コードクローンの存在はソフトウェアの保守を困難にすると言われている．コードクローンに対する主な保守作業として，同時修正と集約が挙げられる．同時修正とは，クローンセット（互いにコードクローンとなっているコード片の集合）を一貫して編集することである．例えば，クローンセット中の 1 つのコード片に欠陥が存在して修正を行う場合，そのクローンセット中の他のコード片にも一貫した修正を行う必要が考えられる．また，集約とはクローンセットを 1 つのサブルーチンにまとめることである．集約を行うことによって，ソースコード中のコードクローンの数を削減することができる．

これらのコードクローンに対する保守作業を効率よく行うためのシステムとして，コードクローン変更管理システム Clone

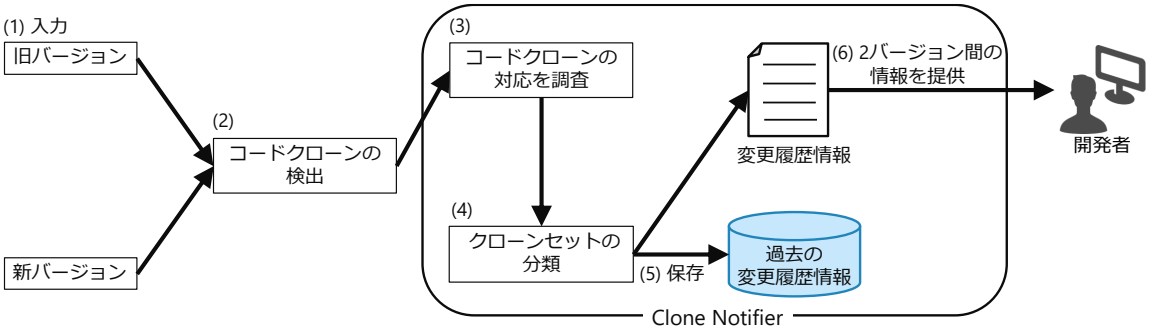


図 1: Clone Notifier の概要図

Notifier が提案されている [2]．Clone Notifier は，2 バージョン間で行われたコードクローンの追加，編集，削除といった変更履歴に基づいてクローンセットを分類し，その変更履歴情報を開発者に提供する．コードクローンに対する保守作業をより効率的に行うためには，過去から現在にわたるクローンセットの変更履歴全体を分析する必要がある．しかし，Clone Notifier は 2 バージョン間の変更履歴情報のみを提供するため，過去から現在にわたるクローンセットの変更履歴全体を分析することは困難である．

そこで本研究では，任意の隣接するバージョン間のクローンセットの変更履歴情報に関する統計データを集計し，過去から現在にわたるクローンセットの変更履歴全体を積み上げ棒グラフによって可視化するコードクローン変更履歴可視化システムの開発を行った．本システムを用いてプロジェクトの過去から現在にわたるクローンセットの変更履歴全体を可視化することで，開発者は各クローンセットの大小，比率，および推移を直観的に知ることができる．また，過去から現在にわたる各バージョン間におけるクローンセットの変更履歴情報の比較が容易となる．このようにプロジェクトの過去から現在にわたるクローンセットの変更履歴全体を可視化することで，本システムは，開発者のコードクローンの分析と保守を支援する．

また，本システムの分析事例を紹介し，本システムがクローンセットの集約の保守作業において有効であることを示す．具体的には，本システムで集約の対象となる可能性があるクローンセットの情報を開発者に提供する．開発者はそのクローンセットが集約可能であるか判断し，集約可能なクローンセットであれば，実際にそのクローンセットを集約する．その後，そのクローンセットが集約されてたことを本システムで確認するまでの事例を紹介する．

以降，2 章では，コードクローン，コードクローン検出ツール，およびコードクローン変更管理システム Clone Notifier について述べる．3 章では，コードクローン変更履歴可視化システムについて述べる．4 章では，本システムの分析事例について述べる．最後に 5 章でまとめについて述べる．

2. 関 連 研 究

2.1 コードクローン

コードクローンとは，ソースコード中に存在する互いに一致または類似した部分を持つコード片のことである．また，互いにコードクローンとなっているコード片の集合をクローンセットと呼ぶ．

コードクローンに対する保守作業には以下の項目が挙げられる．

同時修正：クローンセット中のコード片を一貫して編集すること．

集約：クローンセット中のコード片と同様の処理を実装するサブルーチンを作り，各コード片をそのサブルーチンの呼び出し文に置き換えること．

例えば，あるクローンセット中の 1 つのコード片に欠陥が発見された場合，そのクローンセット中の他のコード片についても同時修正を検討する必要がある．また，このようなコードクローンに対する修正コストを削減するために，新たに発生したコードクローンは集約を検討する必要がある．

しかし，一般的にコードクローンの検出量は膨大となりやすい．検出されたコードクローンの量が膨大となる場合，その中から，同時修正が行われていないクローンセットや新たに発生したコードクローンなどの変更履歴情報を人手で確認することは困難である．

2.2 コードクローン検出ツール

ソースコード中の中から，自動で高速にコードクローンを検出するためのツールがいくつか提案されている．現在までに提案された代表的なコードクローン検出ツールには以下の 4 つがある．

関数クローン検出ツール [3]：情報検索技術を用いることで，意味的に処理が類似した関数単位のコードクローンを検出するツール．

ブロッククローン検出ツール [4]：情報検索技術を用いることで，意味的に処理が類似したブロック単位（関数単位より小さい粒度）のコードクローンを検出するツール．

CCFinderX(注1)：字句解析を用いることで，構文的に一致した字句単位のコードクローンを検出するツール．

SourcererCC [5]：大規模なソフトウェアに対して，意味的に処理が類似したコードクローンを高速に検出するツール．

(注1)：http://www.ccfinder.net/ccfinderx-j.html

2.3 コードクローン変更管理システム Clone Notifier

2.3.1 Clone Notifier の概要

2.1 節で示したコードクローンに対する保守作業を効率よく行うためのシステムとして，コードクローン変更管理システム Clone Notifier が提案されている．

Clone Notifier は，2 バージョン間で行われたコードクローンの追加，編集，削除といったコードクローンの変更履歴に基づき，クローンセットを分類し，その変更履歴情報を開発者に提供する．Clone Notifier を用いてコードクローンの変更履歴を管理をすることで，保守作業の対象となるコードクローンの確認コストを削減することができる．

Clone Notifier の概要を図 1 に示し，処理流れを以下に示す．

（1） 旧バージョンと新バージョンのプロジェクトを入力

（2） コードクローン検出ツールを用いて，入力された 2 つのバージョンのコードクローンを検出

（3） 同ファイル，同位置にあるコードクローンの対応を調査

（4） コード片の追加，編集，削除といった変更履歴に基づきクローンセットを分類

（5） 変更履歴情報を過去の変更履歴情報として保存

（6） 2 バージョン間におけるクローンセットの変更履歴情報を開発者に提供

開発者は，処理 (6) で提供される変更履歴情報をウェブユーザインタフェースを用いて確認することができる．そして，開発者は提供された変更履歴情報から，コードクローンに対する保守作業の必要性を判断する．

2.3.2 クローンセットの分類

本項では，2.3.1 項で説明した Clone Notifier の処理 (4) で行うクローンセットの分類について説明する．Clone Notifier は，変更履歴に基づいて，2 バージョン間のソースコードに含まれるすべてのクローンセットを以下の 4 つのクローンセットに分類する．

Stable クローンセット：2 つのバージョンにわたって存在し，変更がなかったクローンセットを指す．

Deleted クローンセット：旧バージョンのみに存在するクローンセットを指す．新バージョンで集約などによって削除されたクローンセットを意味する．したがって，開発者は Deleted クローンセットに分類されたクローンセットを確認することによって，集約が行われたことを確認することが可能である．

Changed クローンセット：2 つのバージョンにわたって存在し，変更されたクローンセットを指す．一貫した修正がされていない Changed クローンセットは同時修正を検討する必要がある．

New クローンセット：新バージョンのみに存在するクローンセットを指す．新バージョンで新たなコードクローンが追加されたことを示す．コードクローンの数を削減するために，New クローンセットは集約を検討する必要がある．

2.3.3 Clone Notifier の問題

Clone Notifier には以下の 2 つの問題がある．

（問題 1）各クローンセット数の大小や比率を直観的に知ることが困難

クローンセット分類情報	
総クローンセット数	510
STABLEクローンセット数	457
CHANGEDクローンセット数	11
NEWクローンセット数	7
DELETEDクローンセット数	35

図 2: Clone Notifier の出力結果

Clone Notifier が出力する結果を図 2 に示す．図 2 より，Clone Notifier は，各クローンセット数の表を開発者に提供する．しかし，表での結果の提供では各クローンセット数の大小や比率を直観的に知ることができない．

（問題 2）過去から現在にわたる複数の変更履歴情報の比較が困難

現状の Clone Notifier で複数の変更履歴情報を比較する例を図 3 に示す．図 3 では，V_{t-2} と V_{t-1} バージョン間の変更履歴情報を過去の変更履歴情報，V_{t-1} と V_t バージョン間の変更履歴情報を現在の変更履歴情報としている．これら 2 つの変更履歴情報を比較する際に，開発者は 2 つの変更履歴情報を逐次参照する必要がある．また，複数の変更履歴情報の比較に表を用いると，過去から現在にわたって，各クローンセット数がどのくらい増減したのかといったクローンセット数の推移に関する情報の取得が困難である．このため，複数の変更履歴情報の比較において，表での比較は適切でない．

3. コードクローン変更履歴可視化システム

3.1 本システムの概要

本研究では，2.3.3 項で示した Clone Notifier の問題を解決することで，開発者のコードクローンに対する保守作業を支援するコードクローン変更履歴可視化システムの開発を行った．

本システムの概要を図 4 に示し，処理流れを以下に示す．

(A) 旧バージョンと新バージョンのプロジェクトを入力

(B) コードクローン検出ツールを用いて，入力された 2 つのバージョンのコードクローンを検出

(C) 同ファイル，同位置にあるコードクローンの対応を調査

(D) コード片の追加，編集，削除といった変更履歴に基づきクローンセットを分類

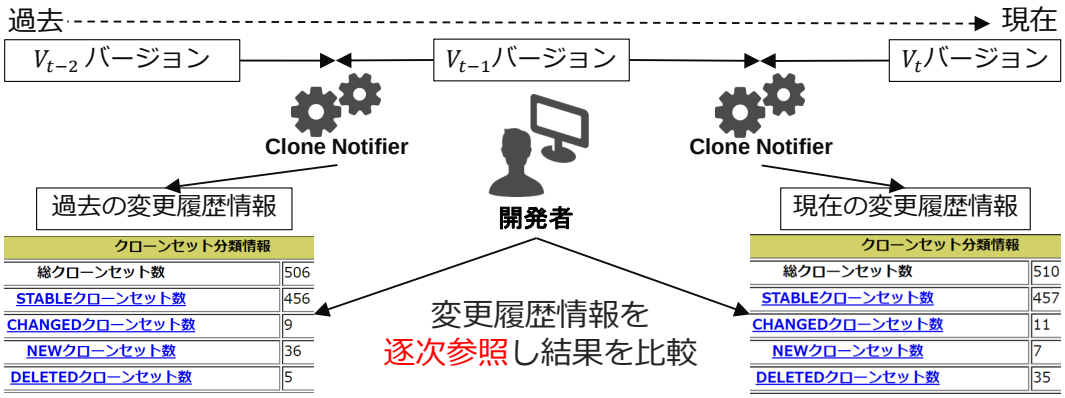
(E) 変更履歴情報を過去の変更履歴情報として保存

(F) 過去の変更履歴情報から任意の隣接するバージョン間のクローンセットの変更履歴情報を集計

(G) 変更履歴全体を含めたクローンセットの変更履歴情報を可視化して開発者に提供

開発者は，処理 (G) で変更履歴情報を可視化したものをウェブユーザインタフェースを用いて確認することができる．そして，開発者は提供された変更履歴情報から，コードクローンに対する保守作業の必要性を判断する．

本システムでは，2.3.3 項で示した Clone Notifier の問題について，処理（F）と（G）で以下の対処をしている．



クローンセット分類情報	
総クローンセット数	506
STABLEクローンセット数	456
CHANGEDクローンセット数	9
NEWクローンセット数	36
DELETEDクローンセット数	5

クローンセット分類情報	
総クローンセット数	510
STABLEクローンセット数	457
CHANGEDクローンセット数	11
NEWクローンセット数	7
DELETEDクローンセット数	35

図 3: 複数の変更履歴情報を比較する例

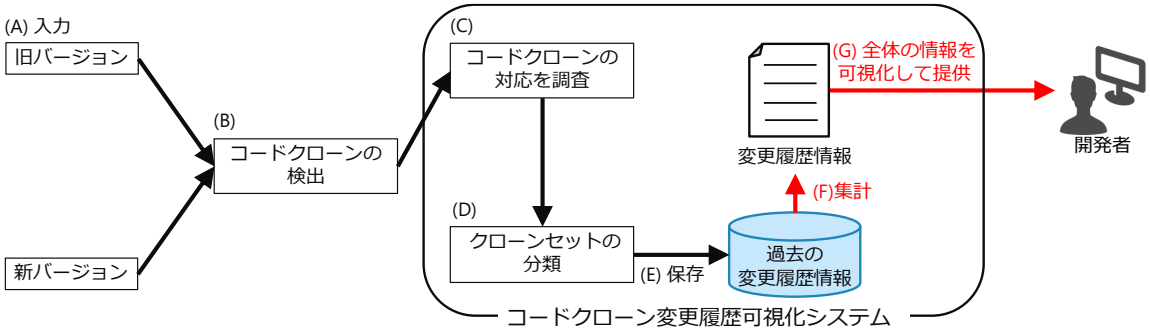


図 4: コードクローン変更履歴可視化システムの概要図

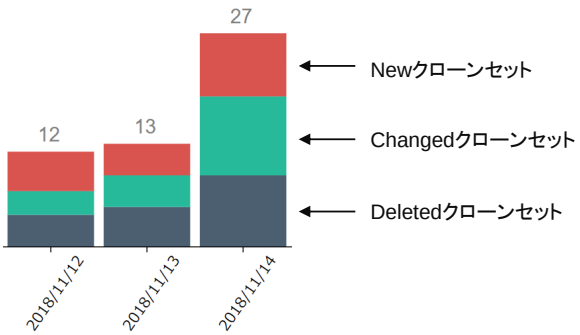


図 5: 本システムが出力する積み上げ棒グラフ

（問題 1 の対処）積み上げ棒グラフで各クローンセット数の大小や比率を可視化することで対処

（問題 2 の対処）過去から現在にわたる変更履歴情報を含めた変更履歴全体のクローンセット数の積み上げ棒グラフを時系列に並べて可視化することで対処

問題 1 と問題 2 に対する対処を行った本システムが出力する結果を図 5 に示す． 図 5 では，縦軸は各クローンセットの数を示し，下の各要素から順に Deleted（黒色の要素），Changed（緑色の要素），および New（赤色の要素）クローンセットの数を表している．横軸は，各分析日を示している．図 5 の場合，2018 年 11 月 11 日の 00 時 00 分を時点 t=0 として，1 日間隔で 2018 年 11 月 12 日から 2018 年 11 月 14 日までの間，t=1,2,3 の計 3 時点に対して本システムを用いて分析している．つまり，2018 年 11 月 11 日時点から 2018 年 11 月 12 日時点までの変更履歴情報を 2018 年 11 月 12 日の分析結果，2018 年 11 月 12 日時点から 2018 年 11 月 13 日時点までの変更履歴情報を 2018 年 11 月 13 日の分析結果，2018 年 11 月 13 日時点から 2018 年 11 月 14 日時点までの変更履歴情報を 2018 年 11 月 14 日の分析結果としている．

本システムを用いてプロジェクトの過去から現在にわたるクローンセットの変更履歴全体を可視化することで，開発者は積み上げ棒グラフの各要素から各クローンセットの大小，比率，および推移を直観的に知ることができる．また，過去から現在にわたる各バージョン間におけるクローンセットの変更履歴情報の比較が容易となる．このようにプロジェクトの過去から現在にわたるクローンセットの変更履歴全体を可視化することで，本システムは，開発者のコードクローンの分析と保守を支援する．

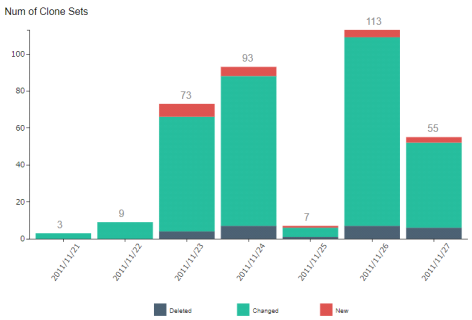
3.2 ウェブユーザインタフェースでの変更履歴情報の提供

ウェブユーザインタフェースの例として，PostgreSQL(注2) プロジェクトにある複数バージョン間において本システムを適用した結果を図 6 に示す．以下では，変更履歴全体を可視化した積み上げ棒グラフページ（図 6（a）），クローンセット一覧ページ（図 6（b）），およびソースファイルページ（図 6（c））について説明する．

(a) 変更履歴全体を可視化した積み上げ棒グラフページ

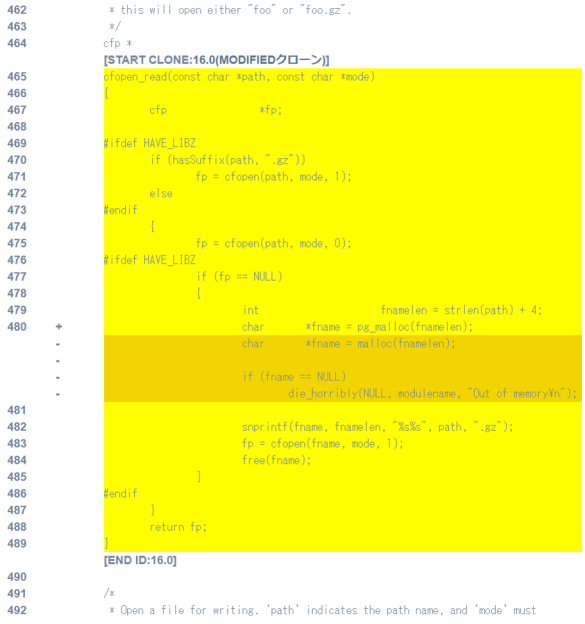
クローンセットの変更履歴全体を含めた変更履歴情報を可視化した積み上げ棒グラフが表示される．積み上げ棒グラフは，上の各要素から順に，New（赤色の要素），Changed（緑色の要素），および Deleted（黒色の要素）クローンセットの順で表示される．また，積み上げ棒グラフの上の数字は，New，

(注2)：https://www.postgresql.org/



(a) 変更履歴全体を可視化した積み上げ棒グラフページ

(b) クローンセット一覧ページ



(c) ソースファイルページ

図 6: ウェブユーザインタフェースの例

Changed，Deleted クローンセットの合計の数を示す．積み上げ棒グラフの要素をクリックすると，クリックした要素の分析日のクローンセット一覧ページに移動する．

本システムでは，積み上げ棒グラフの要素に，Stable クローンセットの情報を含めていない．これには 2 つの理由がある．

1 つ目の理由は，Stable クローンセットの必要性が低いためである．本システムは，同時修正や集約の保守作業の支援を目的としているため，これらの保守作業にあまり関係がない Stable クローンセットの情報は必要性が低い．

2 つ目の理由は，短期間の分析では，クローンセットのほとんどが Stable クローンセットに分類されるためである．本システムは，1 日や 1 週間単位と比較的短い期間での分析を目的

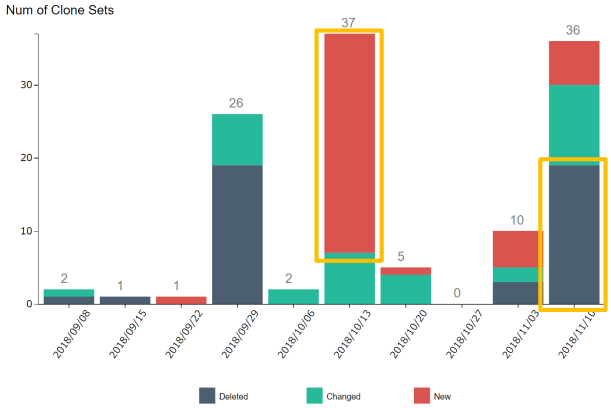


図 7: 集約の分析事例

としており，ほとんどのコードクローンは短期間の開発では変更されない [6]．このため，Stable クローンセットを積み上げ棒グラフの要素に含めた場合，積み上げ棒グラフの大半の要素を Stable クローンセットが占めてしまう可能性が高くなる．これは，New，Changed，および Deleted クローンセットといったコードクローンの保守作業に必要な情報を開発者は取得しづらいくしてしまう．これらの理由で，積み上げ棒グラフに Stable クローンセットの情報を含めていない．

(b) クローンセット一覧ページ

指定した分析日における 2 バージョン間に含まれるクローンセットの一覧が New，Changed，Deleted，および Stable クローンセットの順で表示される．また，各クローンセットに関して，属するコードクローンの一覧が表示される．各コードクローンは，ID, 分類，コードクローンが含まれるソースファイル名，およびソースファイル中のコードクローンの位置が表示される．ソースコードのアイコンをクリックすることで，そのコードクローンが含まれるソースファイルページに移動する．

(c) ソースファイルページ

コードクローンが含まれるソースファイルが表示される．コードクローンとなっているコード片は黄色の背景色が付いている．"+"は新バージョンで追加された行，"-"は新バージョンで削除された行を示す．

4. 分析事例

本システムを用いて行った分析事例を紹介する．今回，紹介する事例の分析観点を以下に示す．

（1） 本システムは，集約の対象となる可能性のある New クローンセットの情報を開発者に提供できるか

（2） 集約されたことを最新の変更履歴情報で確認できるか

また，本システムで行った分析内容を以下に示す．

- 対象ソフトウェア：Apache Tomcat(注3)
- プログラミング言語：Java
- コードクローン検出ツール：SourcererCC
- 分析期間：2018 年 9 月 1 日の 00 時 00 分を時点 t=0 として，1 週間間隔で 2018 年 9 月 8 日から 2018 年 11 月 10 日までの間，t=1,2,…,10 の計 10 時点

(注3)：http://tomcat.apache.org/

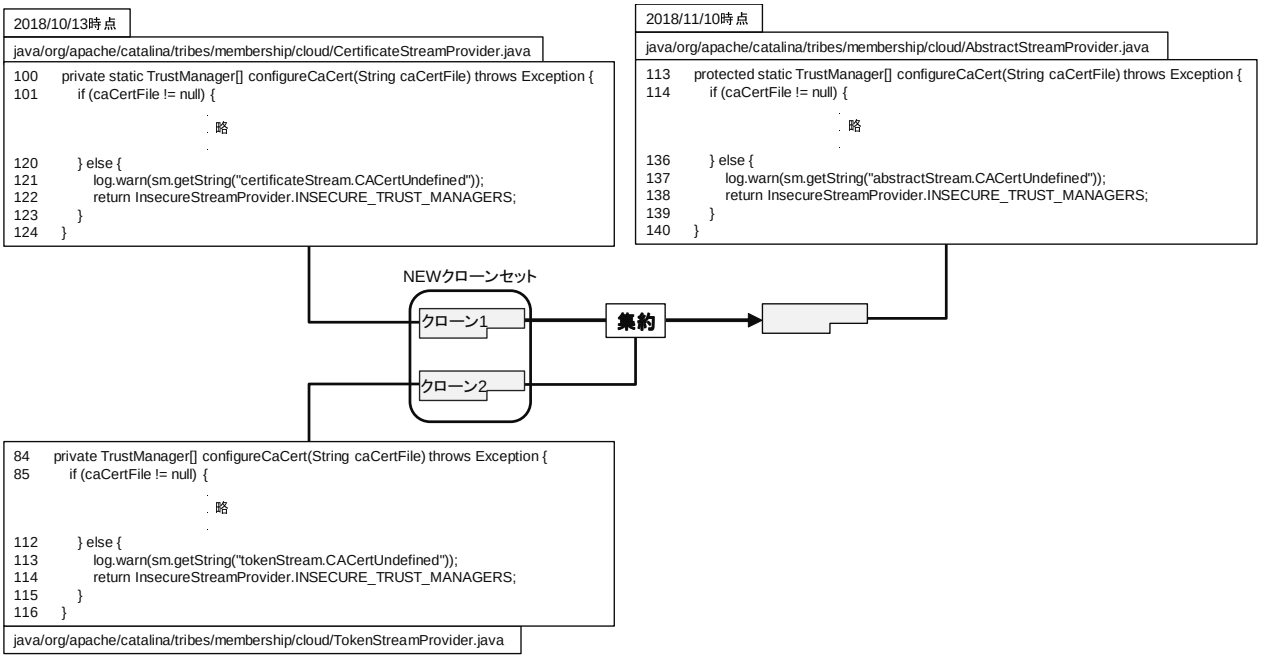


図 8: 本システムを用いて分析した集約のコード例

以上の分析内容で，本システムが出力する可視化結果を図 7 に示す．この図より，以下のことがわかる．

（1） 2018 年 10 月 13 日時点に積み上げ棒グラフの赤色の要素があるため，New クローンセットが発生していること

（2） 2018 年 11 月 10 日時点に積み上げ棒グラフの黒色の要素があるため Deleted クローンセットが発生していること

実際にソースコードを確認すると，2018 年 10 月 13 日時点発生していた New クローンセットが 2018 年 11 月 10 日時点で Deleted クローンセットとなり集約されていた．実際の 2018 年 10 月 13 日時点で発生した New クローンセットと集約後の 2018 年 11 月 10 日時点のソースコードを図 8 に示す．この図より，2018 年 10 月 13 日時点で CertificateStreamProvider.java と TokenStreamProvider.java の両方に存在していた configureCaCert メソッドが 2018 年 11 月 10 日時点で集約され，AbstractStreamProvider.java のみに存在するようになったことがわかる．

このように本システムでは，集約対象となる New クローンセットの情報を開発者に提供し，それを集約できたかどうかを Deleted クローンセットの情報で開発者に知らせることができる．本システムでは，新たに発生したコードクローンを New クローンセットとして，積み上げ棒グラフの赤色の要素で表示する．また，集約されたコードクローンは Deleted クローンセットとして，積み上げ棒グラフの黒色の要素で表示する．このため，集約に関する保守作業の情報を開発者に色で直観的に知らせることができる．さらに，過去の変更履歴情報を含めた変更履歴全体を可視化しているため，過去から現在の変更履歴情報の参照が容易となる．これにより，過去の変更履歴情報より過去に保守作業漏れとなった集約対象となる New クローンセットを見つけだし，現在の変更履歴情報で集約が成功しているか把握することができる．つまり，本ツールでは，過去の変更履歴情報より過去に保守作業漏れとなったクローンセットの情報を開発者は取得することができる．そして保守作業後，現在の変更履歴情報より，そのクローンセットに対して行った保守作業の内容を確認することができる．

5. ま と め

本研究では，任意の隣接するバージョン間のクローンセットの変更履歴情報に関する統計データを集計する．そして，過去から現在にわたるクローンセットの変更履歴全体を積み上げ棒グラフによって可視化するコードクローン変更履歴可視化システムの開発を行った．また，本システムの分析事例を紹介し，本システムがクローンセットの集約の保守作業において有効であることを示した．

謝辞 本研究に対して，様々なご協力をいただいた日本電気株式会社前田直人氏，渋谷健介氏に深く感謝する．本研究は JSPS 科研費 JP25220003, JP18H04094, JP16K16034 の助成を受けた．

文 献

[1] 肥後芳樹, 楠本真二, 井上克郎, “コードクローン検出とその関連技術,” 電子情報通信学会論文誌, vol.J91–D, no.6, pp.1465–1481, 2008.

[2] 山中裕樹，崔 恩瀞，吉田則裕，井上克郎，佐野建樹，“コードクローン変更管理システムの開発と実プロジェクトへの適用,” 情報処理学会論文誌，vol.54，no.2，pp.883–893，2013.

[3] 山中裕樹，崔 恩瀞，吉田則裕，井上克郎，“情報検索技術に基づく高速な関数クローン検出,” 情報処理学会論文誌，vol.55，no.10，pp.2245–2255，2014.

[4] 山中裕樹，崔 恩瀞，吉田則裕，井上克郎，“情報検索技術に基づく細粒度ブロッククローン検出,” コンピュータソフトウェア，vol.35，no.4，pp.16–36，2018.

[5] H. Sajnani, V. Saini, J. Svajlenko, C.K. Roy, and C.V. Lopes, “SourcererCC: Scaling code clone detection to big–code,” Proc. of ICSE, pp.1157–1168, 2016.

[6] J. Krinke, “Is Cloned Code more Stable than Non-cloned Code?,” Proc. SCAM, pp.57–66, 2008.