6A-5

プログラム理解支援を目的とした 分散ペアプログラミングのコミュニケーションログの活用

秀毛嶺維馬† 奥野拓‡

公立はこだて未来大学大学院† 公立はこだて未来大学‡

1 はじめに

離れた二者が画面共有機能などを用いて共にコーディングする,分散ペアプログラミングという手法がある[1]．この手法では作業者らの対話記録が容易であるが，多くの場合この記録は活用されない．しかし，コーディング時のコミュニケーションには意思決定が含まれている．そのため，このログは何らかの活用ができると考えられる．

分散ペアプログラミングのような分散開発においては，開発メンバ間の密接な情報交換が難しい．システムの拡張や保守のためにはソースコードの細かな理解が必要であるが，実装担当者からの説明やドキュメントなどの手がかりが無い場合，プログラムの理解には非常に時間が掛かる．

この問題に対して，分散ペアプログラミングのコミュニケーションログに着目し，作業者単独で行えるプログラムの理解支援を行う．特に手書きの注釈によるコミュニケーションに焦点をあて，本稿ではその有効性について述べていく．

2 プログラム理解の課題と既存の支援手法

2.1 編集履歴の量

統合開発環境では，ファイル内容の変更差分が編集履歴として残される．この編集履歴から，ソースコードの変遷を追うことができる．しかし，ステップ数の多いソースコードや頻繁に編集されたソースコードは編集履歴の量が膨大となるため，任意の点を参照するのに時間がかかる．

大森らは編集履歴を時系列に可視化することで，この課題に対応している[2]．

2.2 編集理由の類推

編集履歴には編集意図は記録されない．版管理システムではコミットメッセージを残せるが，これは一つの大きな変更に対して残される物である．そのため，一つ一つの変更に対する編集意図を知るには不十分である．西川らはチャットの記録を編集意図の類推に利用する手法を提案している[3]．しかしチャットのログには編集意図の類推には無関係なノイズが多数含まれる．そのため，コミュニケーションログの利用にはこのノイズが課題となる．

Understanding Support of Program by Communication Log of Distributed Pair Programming

†Reima SHUKE, Graduate School of Future University Hakodate

‡Taku OKUNO, Future University Hakodate

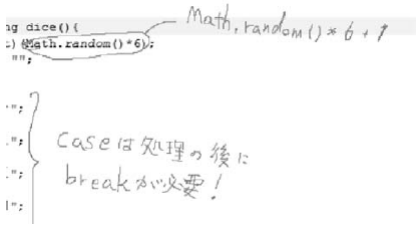

図１．手書き注釈の例

表１．対話手法毎の主な利用状況

対話手法	利用状況	頻度
テキストチャット	Web サイトの文章の引用	低
音声通話	ソースコードの説明 疑問点への問いかけや対応	高
手書き注釈	図表による音声通話の補助 記号による注目点の指示 気づいたことのメモ	中

3 プログラム理解支援手法の提案.

3.1 手書き注釈ログの利用

本稿では 2.2 節で課題として挙げたノイズの削減に，手書き注釈のコミュニケーションログを用いる．ここでの手書き注釈とは，図 1 のようにソースコード上に付記された文字や図形を指す.

表 1 は 3 種類のコミュニケーション手法が分散ペアプログラミングのどの場面で主に利用されるかを示したものである[1]．ソースコードの説明が含まれる音声ログをユーザに提供することで，ドキュメントには残らないソースコードの情報が得ることができる．しかし，そのままでは 2.2 節で挙げたコミュニケーションログのノイズが問題になると推測できる．そこで，ソースコード編集に無関係な対話は少ないと考えられる手書き注釈を利用する．手書き注釈により補助された音声ログを抽出することで，ノイズの少ないコミュニケーションログをユーザに提供できると仮説を立てた.

3.2 編集履歴と手書き注釈ログの可視化

3.1 節より，ソースコードの編集履歴と手書き注釈の記録を可視化する手法を提案する．システムの全体図を図 2 に示す．このシステムは，コーディングエディタの変遷を記録した動画と，編集操作・注釈付記を時系列に並べた作業タイムラインからなる．作業タイムラインの特定箇所を選択すると，その時点のソースコードと作業者等の対話音声が提示される．ユーザは作業タイムラインを手がかりに調査したい変更点を見付け，そこでの

対話を聞くことで編集意図の類推及びプログラム理解を進めていく.

4 手書き注釈と対話の結びつきの検証

3.1 節で述べた仮説の検証を行った.以下にその方法と結果について述べる.

4.1 検証方法

分散ペアプログラミングの試行を記録し，作業中の対話を観察した．注釈が付記される際，編集理由となるような対話の有無を調査した.

試行の観察のため，Java プログラミングを習得している 2 名を選んだ．被験者 1 は Java でのアプリケーション実装経験があり，被験者 2 は日常で Java を利用していない.

課題はプログラミングの講義で利用されたものを用いた．被験者にはプログラムの仕様と参考資料，一部のメソッドが未実装のクラスファイル，テストケースを提供した．また，実装には統合開発環境の Eclipse を利用し，コミュニケーションには音声通話，テキストチャット，手書き注釈を利用した．注釈付記にはペンタブレットを用いた.

試行にあたっては実装役・補助役を決め，20 分毎に役割の交代を行った.

4.2 検証結果

検証の様子を撮影し，動画を元にソースコードの編集・手書き注釈が付記された箇所を図示した(図 3)．試行中はお互いに注釈を付記する際やソースコードを編集する直前に，自分の考えを発話していた．しかし，各被験者の注釈数には大きな差が見られた．被験者 1 が 284 回注釈付与したのに対し，被験者 2 の注釈回数は 15 回だった．ただし，注釈付記のためにペンタブレットが接地してから離れるまでを 1 回と数える.

本稿の提案手法は，作業者等の対話の如何で効果が左右される．特に手書き注釈が付記されなければ，対となる音声通話の場所の当たりを付けられない．提案手法が有効である状況を明らかにするため，手書き注釈が付記されにくい状況について，次に考察する.

4.3 手書き注釈が付記されにくい状況

手書きの注釈が付記されにくい状況があることが実験から分かった．一つは被験者らのコミュニケーションに問題がある場合，もう一つは被験者らが独立して作業している場合である．以下に詳細を述べる.

4.2 節から，被験者 1 に比べて被験者 2 の注釈数に開きがあった．これは作業者の熟練度の差に起因すると考えられる[4]．熟練度に差がある場合対話数が減少する場合があり，注釈の数も対話数に伴って減少する．この場合において発言機会を減少させないためには，互いに作業進捗の確認するよう注意することが必要となる.

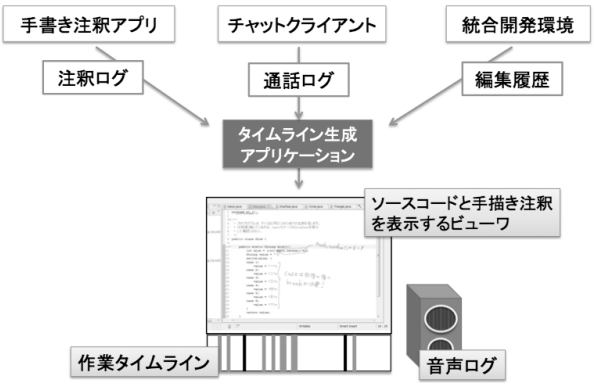

図 2. 提案手法の全体図

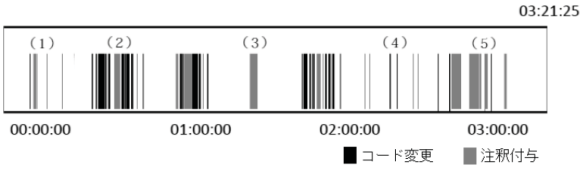

図 3. 試行中の被験者の行動

また図 3 の(4)では，注釈による補助が行われなかった．ここでは作業者らは実装に試行錯誤していた．実装役が思うままにソースコードを編集する間，補助役が参考資料の調査をしていた．このように実装の方針が定まっていない場合には，対話が少なく，注釈による意図伝達も行われない.

6 おわりに

本稿では，分散ペアプログラミング中に蓄積されるコミュニケーションのログを利用して，プログラム理解を支援する手法を提案した.

検証では，仮説が成り立つ状況を確認した．しかし被験者らのコミュニケーションが少ない場合や，独立して作業している場合には注釈が付記されないことがわかった.

また，提案手法を取り入れたログ視聴アプリケーションを実装した．今後，このアプリケーションを利用して，ソフトウェアの保守作業時を想定した提案手法の検証を行っていく.

参考文献

[1]秀毛嶺維馬，奥野拓：分散ペアプログラミングにおける手書き注釈を用いたコラボレーション機能の提案，情報処理学会北海道シンポジウム講演論文集，pp.175-180,(2012)

[2]大森隆行，丸山勝久：プログラム開発履歴調査のための編集操作再生器，ソフトウェア工学の基礎XVII，pp.45-54,(2010)

[3]西川穂高，酒井三四郎：分散ペアプログラミングにおけるコミュニケーションログとコード変更箇所の対応付けによる理解支援，情報科学技術フォーラム一般講演論文集, pp.103-104, (2005)

[4]Laurie Wiliams and Robert Kessler: Pair Programming Illuminated, Addison-Wesley Professional, (2002)