ビデオ会議システムにおけるリモートモブプログラミングのための役割ごと支援機能の検討

寺中 靖幸　高山 裕暉　濱田 優弥　井垣 宏

複数人でチームを組んでソフトウェアを開発する手法の一つとしてモブプログラミングというものが知られている．モブプログラミングとは 1 つの端末を使って複数人でソフトウェアを開発する手法である．開発者らは端末を操作するドライバと呼ばれる役割と開発画面を見ながら意見を出してドライバをサポートするナビゲータと呼ばれる役割に分かれ，役割を短時間で交代しながら開発を進めていく．昨今のテレワークの普及や感染症の影響による自粛要請等の世界情勢において，このモブプログラミングをリモート環境で行う事例が増加しつつある．著者らが実際にリモート環境でモブプログラミングを実施した結果，対面でのモブプログラミングと比較して，引き継ぎやドライバ・ナビゲータ間のコミュニケーションにおける課題が有ることが分かった．そこで本研究では，ドライバやナビゲータの役割支援を目的としたビデオ会議システムにどのような機能が求められるかを分析し，UI 等の設計を行った．

1 はじめに

1 台のパソコンの前に 2 人の開発者が座り，1 人が開発して 1 人が助言やコードレビューを行うペアプログラミングと呼ばれるソフトウェア開発手法がある [6]．大学などの教育機関におけるプログラミング教育手法の一つとしても採用されており，学生が独力で開発する場合と比較し，よりよい品質のコードを書くことができるようになったといった良い側面が数多く報告されている [5] [4]．モブプログラミングは 3 人以上の開発者が 1 台のパソコンの前に座って協力しながら問題を解決するペアプログラミングを発展させたチーム開発手法の一つである [9]．1 人が実際にパソコンを操作してプログラミングを行い，残りの開発者はそれを見ながら問題点を指摘したり，解決方法の提案を行ったりすることで，共同で開発を進めていく．ペアプログラミングと異なる点として，一つのチームが一つのパソコンに集まって作業を進めていくため，情報の共有やタスクの割当といったコミュニケーションに係るオーバヘッドが削減できることやあらゆる問題をチームのメンバ全員で即座に共有するため，解決が迅速に行えることなどが挙げられている [3]．

一般的なモブプログラミングでは，パソコンに向かって開発作業をする開発者（ドライバと呼ばれる）とドライバを支援するその他の開発者（ナビゲータと呼ばれる）が 10 分〜20 分程度ずつ（モビングインターバル，本稿ではインターバルと呼ぶ）交代して開発をすすめる．ドライバは時間が来るとパソコンの前から離れ，次のドライバと交代する．事前に決めた回数ドライバを交代した後（モビングセッション，本稿ではセッションと呼ぶ），セッションの内容を振り返る．最近ではモブプログラミングを取り入れた日本企業の報告も行われるようになっており [7]，他にも国際会議 [1] が開催されるなど，注目を集めている．

一方でモブプログラミングでは，3 名以上の開発者が 1 つのパソコンの周りに集合して開発を行うため，いわゆる 3 密なコミュニケーションを避けられない．そのため，最近ではリモート環境でモブプログラミングを行う事例なども報告されている [11]．著者らの所属する研究室でも，3 名程度の学生が 1 つのチームと

Consideration of a Video Conferencing System to Support Developers by Role in Remote Mob Programming

Yasuyuki Teranaka, Yuki Takayama, Hiroya Hamada, Hiroshi Igaki, 大阪工業大学 情報科学部, Faculty of Information Science and Technology, Osaka Institute of Technology.

して Zoom[†1] や Discord[†2] といったビデオ会議サービスを利用したリモートモブプログラミングを行っている．対面で実施する場合と異なり，インターネットを介して実施するリモートモブプログラミングでは，開発対象のソースコードの引き継ぎをしなければならず，ドライバ・ナビゲータ間のコミュニケーションが対面よりも難しいといった課題がある．そこで本稿では，我々が行っているリモートモブプログラミングの経験にもとづき，リモートモブプログラミングのためのビデオ会議システムを検討する．

2 リモートモブプログラミング

モブプログラミングをリモート環境で行う場合，1つの部屋に集まるのではなく，ビデオ会議サービスなどを利用して開発者それぞれの所有するコンピュータの画面を共有して開発を行うことになる．教育の手段としてペアプログラミングをリモート環境で実施する取り組みは従来より数多く行われており [8]，一人でプログラミングを行う場合と比較し，学生のモチベーションや開発されるソースコードの品質等複数の観点において良い効果が得られるとされている．

昨今ではリモート環境でモブプログラミングを実施する取り組みも行われており [2]，どこにいても開発環境があれば開発が行える. また, 不要な作業に対する時間の削減を行うことができる. 一人で開発を行うよりも, モブプログラミングを行うことで孤独感が軽減される. などと報告されている．著者らの所属する研究室では，Discord や Zoom といったビデオ会議サービスを利用し，3 名の開発者がオンラインで集まってモブプログラミングを実際に行っている．以降では著者らの実施したリモートモブプログラミング環境とそのときの振り返りから得られた課題について述べる．

2.1 リモートモブプログラミング環境

リモートモブプログラミングの実施に際して，ビデオ会議サービスとして Zoom 及び Discord を利用している．両サービスともに画面共有機能が備わっているが，Zoom では画面共有時に共有者の画面に注釈を付ける機能があるため，ナビゲータがドライバの画面のどの部分についてコメントをしているか伝えやすいという特徴がある．Discord には注釈機能は存在しないが，Zoom などと違い，オンラインでの会議を開始する際に会議にアクセスするための URL を生成して周知するといった手続きが必要ない．そのため，Discord サーバにサインインするだけで音声での会議や画面共有を開始できるといった特徴がある．

今回実施したリモートモブプログラミングでは，Web アプリケーションを主に開発対象としており，開発環境としては Visual Studio Code やブラウザ（Chrome），Windows 用の bash ターミナルを利用している．通常のモブプログラミングと同様にドライバを実施する順番を決定し，インターバルを 20 分〜30 分，3 人が一度ずつドライバを担当したらセッションが終了するものとしている．次節では，実際に著者らが複数回実施したリモートモブプログラミングの振り返りにおいて得られた知見について詳述する．

2.2 対面でのモブプログラミングとリモートモブプログラミングの差異

ビデオ会議サービスを利用して開発者が自分の PC 画面を共有して進めるリモートモブプログラミングでは，対面でのモブプログラミングでは不要であった引き継ぎ作業が必要となった．著者らはソースコードの共有に Github[†3] を利用している．最初のドライバが自身の開発環境上にリポジトリを作成し，開発ブランチを作成して必要に応じてコミットを行う．ドライバ交代時にはすべてのコミットを push する．次のドライバは push された開発ブランチの内容を pull によって取得し，続きを開発する．引き継ぎにおいては他にも DB サーバの共有等，バックエンドサービスの共通化や開発環境の共通化といった作業も必要であることが分かった．Git の操作なども含めて，モブプログラミングに不慣れな学生にとっては，引き継ぎ作業そのものが難しく感じるということが分かった．

他にも対面の場合と比較して，ドライバ・ナビゲータ間のコミュニケーションが希薄になるといった感想

†1 https://zoom.us/jp-jp/meetings.html
†2 https://discord.com
†3 https://github.com/

が得られた．例えばドライバはナビゲータが開発に集中してくれているか分からず不安になり，ナビゲータはドライバが黙って開発を進めているため，集中力が続かないといったコメントがあった．ビデオ会議サービスによっては全メンバの顔を映しながら画面共有を行うことが可能であったが，顔画像が表示されているからといってコミュニケーションの難しさが大きく改善されるといった感想は得られなかった．他にもホワイトボードが使いにくいというものもあった．

一方で対面の場合と比較してリモートモブプログラミングが良かった点についての意見としては，実際に集まる必要がないため予定の調整さえすればいつでもモブプログラミングを実施できるというものがあった．特に Discord を利用して実施している場合，Discord サーバに繋ぐだけでモブを開始できるため，対面で実施する場合と比較して非常に容易に実施できるというコメントが得られた．

以上より，リモートモブプログラミングにおいては，役割ごとの引き継ぎ作業やドライバ・ナビゲータ間のコミュニケーションについて課題があるという知見が得られた．そこで本研究では，これらの課題の改善を目指して，初学者が含まれるリモートモブプログラミングの支援を目的としたビデオ会議システムの機能について検討する．

3 リモートモブプログラミングのための役割ごと支援機能の検討

前節で述べたとおり，従来のビデオ会議システムを活用してリモートモブプログラミングを実施する場合，一定時間ごとに引き継ぎ作業が必要となることやドライバ・ナビゲータ間のコミュニケーションの希薄さが課題として挙がっている．本稿では，不慣れな開発者同士でもリモートモブプログラミングを容易に実行できるようになることを目的とし，ドライバ・ナビゲータ双方を支援するビデオ会議システムの新しい機能について検討する．現在検討中のリモートモブプログラミングのためのビデオ会議システムはドライバ・ナビゲータそれぞれに異なった UI を提供する．役割に応じた UI や機能を開発者に提示することで，引き継ぎやドライバ・ナビゲータ間コミュニケーションの支援を目指している．以降ではドライバ，ナビゲータそれぞれを対象としたビデオ会議システムの画面案を提示し，支援機能について検討する．

3.1 ドライバを対象としたビデオ会議画面の検討

図 1 は我々が開発中のリモートモブプログラミング実施時のドライバの画面である．画面右側がドライバのパソコン画面，左端にビデオ会議システムが提供するドライバ画面が表示されている．ドライバはパソコン画面で開発を行い，開発中のエディタなどがナビゲータに対して共有される．ドライバ画面には上部に残り時間やモブ中断・退室・引き継ぎなどを選択できるドロップダウンリスト，ドライバが共有している画面のサムネイルが表示されている．さらにその下にはモブに参加している全参加者の一覧とチャットエリアが表示されている．

残り時間は設定されたインターバル時間に従って設定されており，ドライバを交代する 5 分前と交代時刻ちょうどに引き継ぎに関するアナウンスがテキストチャットエリアに表示される．アナウンスでは，ドライバが操作するボタンや実行すべきコマンド情報が提示される．1 台のパソコンを 3 人以上のドライバやナビゲータが使い回す対面型のモブプログラミングと異なり，リモートモブプログラミングでは各自のパソコンで開発を行う．そのため，ドライバを別のメンバに引き継ぐ際には開発したソースコード等の成果物も引き継ぐ必要がある．そこで本稿で想定するリモートモブプログラミング環境では，成果物の引き継ぎを版管理システム Git とリポジトリ共有サービスの Github を利用する．Git 及び Github を利用してチーム開発を行う場合，まず開発対象のリポジトリの master ブランチから開発ブランチを開発者や機能に応じて作成する．次に対象の開発ブランチでの実装が完了した時点で，他の開発者によるレビューを行い，開発ブランチの内容を master ブランチにマージする [10]．モブプログラミングではチームの全開発者がモブに参加して開発を進めていくため，1 つのブランチをドライバが作成し，そのブランチを全ドライバが共有して開発を進めていく．そのため，引き継ぎ時には前ドライバが開発ブランチに実装した内容を push

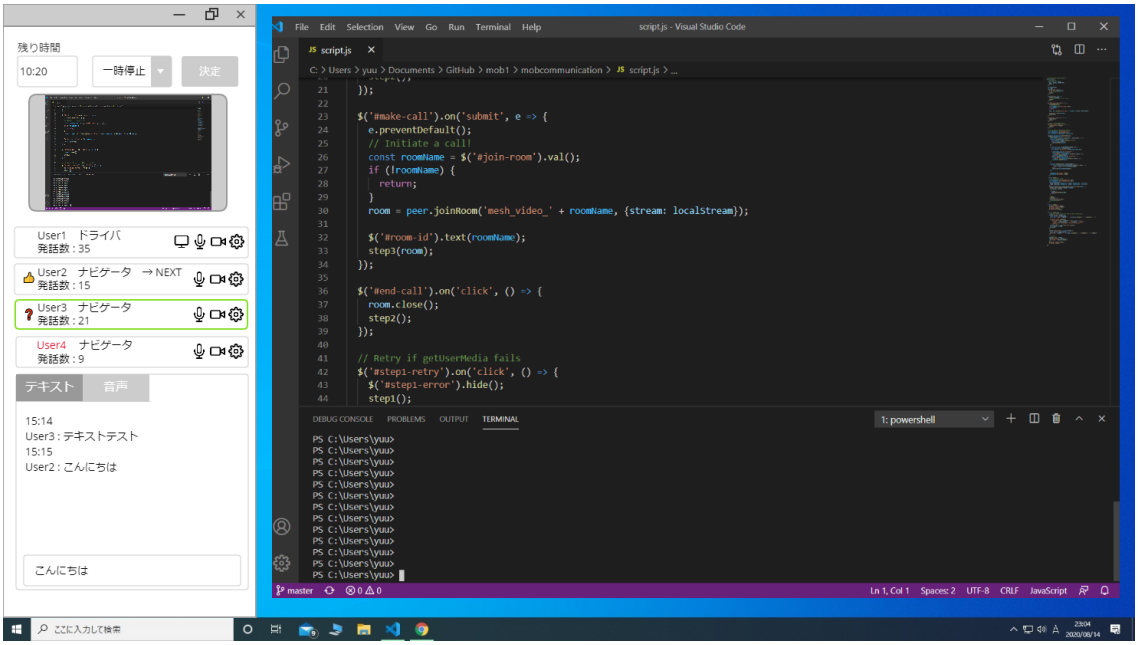

図 1　ドライバの画面

し，次のドライバがその内容を pull して取得して開発を継続していくことになる．ドライバ画面と次節で述べるナビゲータ画面では，これら一連のコマンドに関する情報を開発者が引き継ぎを行うタイミングで提示することで，容易に引き継ぎが行えるよう支援することを想定している．

次に全参加者一覧には，ドライバがナビゲータの状況を把握しやすくするような可視化を行う．まず誰が現在発話しているかを示すため，発話者の枠を緑色で表示する．さらに，ドライバやナビゲータの発話回数（沈黙が継続しないよう各自が意識しやすくすることを目指している），ナビゲータがドライバの画面以外を見ていないか（別の画面を見ているとユーザ名が赤色になる）などをドライバは把握することができる．こういった可視化により，ドライバから見てナビゲータが集中して画面を見ているか，どの程度積極的に発話してくれているかといった情報を得ることができる．

ドライバ画面最下部のチャットエリアには各参加者がテキストを入力して行うテキストチャットエリアとマイクを利用して音声で発話した内容をテキストに変換して提示してくれる音声チャットエリアがある．マイクが利用できない場合やメモにはテキストチャットエリアを利用し，音声で助言やコメントなどを行った場合に，あとで振り返ったりする場合に音声チャットを確認すると言った使い分けが行われることを想定している．

3.2　ナビゲータ支援を想定したビデオ会議画面の検討

図 2 は我々が開発中のリモートモブプログラミング実施時のナビゲータの画面である．ナビゲータ画面には，ドライバが共有している画面が表示されている．さらにその下にはモブに参加している全参加者のカメラ画像，その右には残り時間や退室・交代を選択できるドロップダウンリストと全参加者リスト，チャットエリア，他の参加者に感情を表現するリアクションが表示されている．

顔画像では，ドライバ及び他のナビゲータの状況を可視化するために全参加者を映す．全参加者一覧にも搭載されていた発話者の表示とドライバの画面を見ているかに加え，顔画像も可視化することで，より自

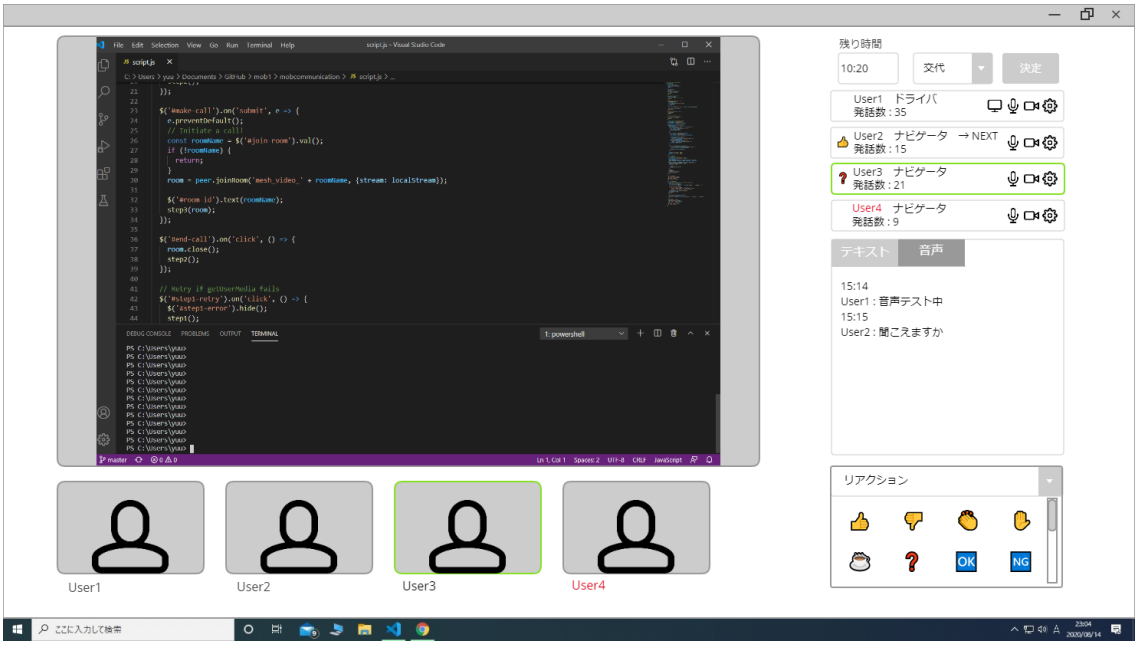

図 2　ナビゲータ画面

然にコミュニケーションをとることができるようになることを目標としている．

画面右側の参加者一覧にはドライバ画面と同じ情報が表示されており，カメラやマイクの設定の他にリアクション内容，発話回数など，モブにどの程度積極的に参加しているかを示す情報が提示される予定である．

4　想定するシナリオ

この節では，開発中のビデオ会議システムを利用する際のリモートモブプログラミングの流れを具体的なシナリオとして示す．

準備

リモートモブプログラミングを開始する際には，利用する開発環境が必要となる．特に引き継ぎを繰り返して開発を進めていく場合，Github のようなリポジトリ管理サービスの利用が前提となる．そのため，準備としてはビデオ会議システムにおける設定と合わせて以下のような作業が行われることを想定している．

- 開発環境のセットアップ．プログラミングを行うにあたって必要なコンパイラやエディタ，git コマンドの設定
- リポジトリの作成．Git を利用してリポジトリを作成し，Github に登録する．
- ビデオ会議システムにおけるモブルームの作成

モブに参加するすべての開発者がコンパイラなどの言語環境や Git コマンドなどを含む開発環境をセットアップする．次に，最初のドライバがプロジェクトの Git リポジトリを作成し，Github に登録する．さらに，ビデオ会議システムにアクセスし，モブプログラミングを行う会議室（モブルームと呼ぶ）を作成する．図 3 にモブルーム作成画面の例を示す．リモートモブプログラミングを開始する際は，1 つのインターバルの時間，対象リポジトリの URL，モブルーム名とパスワードを登録し，作成ボタンをクリックする．モブルームが作成されると，図 1 に示すドライバ画面が表示される．最初のドライバはモブルーム名とパスワードを他の開発者に連絡する．他の参加者はドライバから聞いたモブルーム名とパスワードを入力して，モブルームに参加する．ドライバを分担する順序はモブルームに参加した順番で決定されるが，

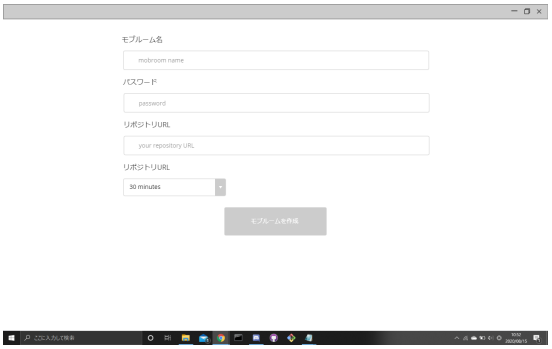

図 3　ルーム作成画面画面

ドライバはドライバ画面の参加者一覧において，ドライバ担当順序を変更する機能を検討している．

リモートモブプログラミングの開始

ドライバはルーム入室時にナビゲータに対して共有する画面の設定を行う．すべてのナビゲータがモブルームに参加したことを確認後，開発用ブランチをリポジトリに作成し，モブプログラミングとしての作業を開始する．開発中は，ドライバおよびナビゲータともに，テキストチャットを利用したい場合はテキストチャットを，音声でコメントする際はマイクを ON にしてコメントを行う．リアクション機能を用いることによって，自分のユーザ情報の欄に「いいね！」等のリアクションを表示することも可能である．

リモートモブプログラミングの引き継ぎ

インターバル終了時刻になると，システムから作業終了通知がテキストチャット欄に自動的に表示される．ドライバはそこに表示される内容に従ってそれまでの開発内容をリポジトリに commit し，次のドライバが変更内容を取得できるよう Github に push する．その後，ビデオ会議システムの引き継ぎボタンをクリックして次のドライバへと引き継ぎ提案を行う．次のドライバに割り当てられているナビゲータはビデオ会議システムの交代ボタンを押し，次のドライバとなる．さらに git リポジトリを pull し，開発対象のブランチを checkout してから開発を開始する．入力すべきコマンドについてはビデオ会議システムが示してくれるものをそのまま入力するだけでも開発を開始できる．

セッションの終了

作業を終了する場合には，引き継ぎと同様にまずは開発内容を commit し，push する．それから引き継ぎではなくセッション終了ボタンを選択し，決定ボタンをクリックする．その後，これまでのコメント内容が記録されたチャットエリアの情報などを利用して，セッションの振り返りを行う．

5　考察

2.2 節において，引き継ぎに関する課題とドライバ・ナビゲータ間コミュニケーションについての課題について述べた．ドライバ交代時の引き継ぎ処理においては，引き継ぎのタイミングや誰から誰に引き継ぎを行うか，引き継ぎ時の git コマンドの支援等を提案システムにより行うことで，不慣れな開発者でも容易にドライバ交代を行うことができるようになる．ドライバ・ナビゲータ間コミュニケーションについては，ナビゲータが画面を見てくれているかをドライバや他のナビゲータから確認する手段として，画面が Active であるかどうかを表示したり，インターバル中の発言回数を表示したりするといった対応を検討している．今後，セッション終了時の振り返りにおいて，参加者がどの程度モブに貢献できたかを数値化するといったメトリクスについての検討も進めていきたい．

6　おわりに

リモート環境でのモブプログラミング支援を目的としたビデオ会議システムに求められる機能の検討を行った．With コロナの状況下において求められるチームでのリモートワーク支援のために，今後実際にシステム開発を行い，継続して評価を行っていきたい．

謝辞

本研究は JSPS 科研費 JP17K00500 の助成を受けたものである．

参 考 文 献

[1] Atsushi Nagata: ここがすごい！モブプログラミング, https://confengine.com/scrum-fest-sapporo-2020/proposal/14013.

[2] Dana Prey: Remote Mob Programming: Unleash your team's potential with remote, collaborative work, https://smartbear.com/blog/collaborate/remote-mob-programming/.

[3] Harald Reingruber: Remote Mob-Programming, https://dev.to/harald3dcv/remote-mob-programming-hunter-industries-18ac.

[4] Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., and Balik, S.: Improving the CS1 Experience with Pair Programming, *SIGCSE Bull.*, Vol. 35, No. 1(2003), pp. 359–362.

[5] Umapathy, K. and Ritzhaupt, A. D.: A Meta-Analysis of Pair-Programming in Computer Programming Courses: Implications for Educational Practice, *ACM Trans. Comput. Educ.*, Vol. 17, No. 4(2017).

[6] Williams, L. and Kessler, R.: *Pair Programming Illuminated*, Addison-Wesley Professional, 2002.

[7] Yasunobu Kawaguchi: 大企業だけど新規ビジネス開発をモブプログラミングでやってみた, https://nihonbuson.hatenadiary.jp/entry/2017/09/24/090000.

[8] Zacharis, N.: Evaluating the Effects of Virtual Pair Programming on Students ' Achievement and Satisfaction, *International Journal of Emerging Technologies in Learning (iJET)*, No. eISSN: 1863-0383(2017).

[9] マーク・パール, 及部敬雄: モブプログラミング・ベストプラクティス, 日経 BP, 2019.

[10] 井上拓海, 小島遥一郎, 藤原賢二, 井垣宏: 版管理システム利用時のソフトウェア開発フロー遵守状況可視化手法の検討, 電子情報通信学会技術研究報告= *IEICE technical report:* 信学技報, Vol. 117, No. 380(2018), pp. 121–126.

[11] 長谷部/渋谷/佐藤/榎本: モブプログラミングが当たり前になるまで〜導入からフルリモート化までの課題と解決〜, hhttps://techblog.yahoo.co.jp/entry/2020052730002064/.