ソフトウェア開発PBLにおける開発履歴データによるチーム活動評価手法

松原 克弥 [*1]　伊藤 恵 [*1]　木塚 あゆみ [*1]
[*1] 公立はこだて未来大学

A Method of Evaluating the Team Activities in Software Development PBL with Their Code Update History

Katsuya Matsubara[*1]　Kei Ito[*1]　Ayumi Kizuka[*1]
[*1] Future University Hakodate

There are many dimensions of student achievement that we need to evaluate in PBL; not only individual learning and final product, but also collaborative process in team activities. Unfortunately, it could be hard to have an enough grasp of the team activities for PBL teachers because the team members may be geographically dispersed, expecially in distributed PBL. This paper proposes a method to evaluate the team activity process in software development PBL using Git as a management tool of their outcomes such as program code and then visualizing updatehistory of files in the Git repository.

キーワード：PBL，プロセス評価，Git

1　はじめに

大学等の高等教育機関における実践的な教育の取り組みとして，情報教育の分野を中心に，PBL(Project-Based Learning) を用いた授業の導入が進んでいる [(5)]．また，2012 年度に 15 の大学が連携して開始された実践的情報教育ネットワーク enPiT[(12)] では，PBL 主催大学の学生に加えて，異なる複数の機関の学生との混成チームを構成して，テレビ会議システム等を通して遠隔から参加する PBL(以下，分散 PBL) が実施されている [(6)]．

新たな授業形式である PBL では，個々の学生の学びを評価するこれまでの学習評価に加えて，チーム毎のプロジェクトの最終成果物や成果発表，さらには，プロジェクトの運営や管理といったチーム活動過程を評価することが重要となる．しかし，分散 PBL のように，PBL 実施教員がメンバ全員の活動状況を目視等で常に確認することが容易ではないため，チームの活動過程に対する評価が難しいという課題がある．

本研究では，ソフトウェア開発を主体とした PBL における開発支援ツールの利用に着目し，ツールが管理するプログラムコードの更新履歴情報からチームの活動状況を把握することを試みる．従来行われてきたチームメンバへのアンケートやヒアリング，報告書や議事録等のドキュメントを介した状況把握では，個々の説明能力やドキュメント記述能力の違いの影響を受ける可能性がある．ツールの利用履歴を用いる本手法では，数値という客観性のあるデータに基づいてチーム活動推定を行うことにより，一貫性のある評価が可能となる．解析対象とするツールは，近年，分散ソフトウェア開発を支援するツールとして導入が進んでいる Git コード管理ツールを用いる．Git リポジトリの利用履歴を可視化することにより，分散したメンバで活動しているチーム状況の把握を容易にすることを目指す．

2　Git コード管理ツールを用いた開発プロセス

Git は，ソースコード・バージョン管理システム (以下，VCS) に分類されるツールである．Subversion のような，1 つのリポジトリを共有する集中型 VCS と異な

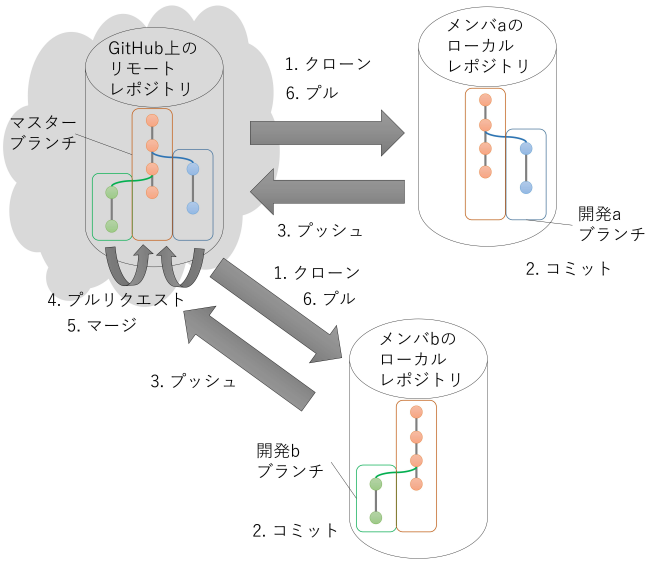

図 1: GitHub Flow に沿った Git リポジトリ運用プロセス

り，Git は，各開発者がソースコードのリポジトリを所有し，リポジトリへの変更内容のみをやりとりするという分散型の VCS となっている [7]．各開発者が独立したリポジトリを用いて開発を進められることや，ソースコードの変更の責任所在とその履歴管理が容易であるという特徴から，多くのオープンソース開発プロジェクトで Git が利用されている．

Git では，コミットと呼ばれる作業でソースコードをリポジトリへ登録する．コミットで登録される情報には，ソースコードの内容に加えて，コミットを行ったユーザの名前，コミット日時，コミット内容を示す文章が含まれる．各開発者により登録されたコミットは，プロジェクトのリポジトリへコミットを組み入れることにより，分散開発されたコードがひとつのソフトウェアとして統合される．リポジトリに登録されたコミットは，その登録順に整列したリスト[1]として管理される．一連のコミット列の途中から異なるコミット列を適用するバージョンを作ることができる．この分岐操作をブランチと呼ぶ．リポジトリ作成時に存在するブランチをマスターブランチと呼び，マスターブランチ以外のブランチは，主に開発中の一時的な用途として作成される．ブランチに登録されたコミットは，マージと呼ばれる操作により最終的にマスターブランチへ組み入れる運用が一般的である．

現在では，クラウド上で Git リポジトリを提供するサービスが登場しており，GiuHub[2]は，Git リポジトリのホスティングサービスとして広く採用されている．GitHub を用いた分散開発では，GitHub Flow と呼ばれる Git 運用プロセスが推奨されている [1, 4]．クラウド上のリポジトリ (以下，リモートリポジトリ) を開発者間で共有し，各開発者の環境にリモートリポジトリの複製 (以下，ローカルリポジトリ) を作成する (図 1 参照)．ローカルリポジトリに作成した開発者別のブランチへ開発成果のコミットを登録していき，適宜，リモートリポジトリへ反映 (プッシュ) する．ビルドや動作可能な状態まで各開発が進んだ適切なタイミングで，プルリクエストと呼ぶ方法によりリモートリポジトリにプッシュした開発者別ブランチからマスターブランチへのマージを要求する．プルリクエストされた開発者別ブランチ上のコミットは，共同開発者間での適切なレビューを経て，マスターブランチへマージされる．マージによりリモートリポジトリのマスターブランチの内容が更新されると，各開発者のローカルリポジトリのマスターブランチへリモートリポジトリの更新を反映 (プル) する．各開発者が開発を継続する場合は，この更新されたマスターブランチの内容を開発者別ブランチへも反映することで，次回のプルリクエスト時のマージが円滑に行われるよう備える．以上の Git 運用プロセスに沿った分散開発では，リモートリポジトリのマスターブランチに統合されたコードが最終成果物となる．

3 Git 利用履歴を用いたチーム活動過程の評価メトリクス

本研究では，PBL におけるチーム活動過程を評価する方法として，成果物を管理する Git リポジトリに対するコミット履歴に注目する．プロジェクトの開発過程で行われるコミットの数や頻度，コミットの作成者によって，ソフトウェア開発におけるチーム活動の進捗や役割分担などの状態を評価することを試みる．

3.1 Git 利用履歴の可視化

本手法では，Git リポジトリのコミット履歴の可視化のためのツールとして，GitStats[3]を用いた．GitStats は，対象リポジトリのマスターブランチに登録されているコミットを多角的な視点で分析して，表やグラフ等を

[1] 変更したソースコードは，スタックのように，登録の古いものから順番に上に重ねて適用される．

[2] https://github.com/

[3] http://gitstats.sourceforge.net/

表 1: チーム活動過程の評価メトリクスとその推定方法

評価メトリクス	評価内容	メトリクスデータ	推定方法
情報共有	チームメンバ間での開発成果や進捗の共有頻度が適切か	日別コミット数	開発期間中のマスターブランチのコミット数の増加タイミングから，マージが行われた回数や頻度により推定する
スケジュール管理	期間中の開発ペースに偏りがないか	日別コミット数	開発期間中の日別の総コミット数を比較することで，開発のペースを推定する
	開発作業が適切な時間帯に行われているか	時間帯別コミット数	コミット時間から，作業を行っている時間帯を推定する
開発分担とタスク量	特定の開発者にタスクが集中していないか．各メンバが適切な作業量を分担しているか	著者別コミット数	各著者のコミット数を比較して，特定の著者のコミットが多くないか，コミット数の増加割合がメンバ間で差が大きくないかで推定する

用いた可視化を行うツールである．集計結果を HTML ページとして出力するため，Web ブラウザにより容易に確認することができる．

3.2 GitStats を用いたチーム活動過程の評価メトリクス

本節では，GitStats による Git リポジトリ利用履歴の可視化結果からチーム活動過程を評価するメトリクスについて定義する．表 1 は，本評価手法におけるチーム活動過程の評価メトリクスとその評価のためのデータ，および，そのデータから活動推定方法をまとめている．以下に，Git 利用履歴からの各メトリクスの評価アプローチについて述べる．

3.2.1 情報共有

チーム活動では，各メンバの状況の共有が重要となる．各メンバがリポジトリを所有する分散開発では，各メンバの開発成果を頻繁にマスターブランチへマージするほうが，他メンバの状況把握につながる．また，マスターブランチへのマージ頻度が低いと，マージの際にメンバ間の変更が衝突する可能性が高まり，マージのオーバヘッドが増加する．GitHub Flow に沿った開発では，通常，マージ以外にマスターブランチへのコミット登録が行われない（図 1 参照）ことを利用して，マスターブランチにおける著者別のコミット数が増加しているときに，マージが行われている可能性が高いと評価する．マージと推測できるコミット数増加のタイミングと回数を観察することで，チームメンバ間での開発成果や進捗の共有頻度を評価することができる．

3.2.2 スケジュール管理

PBL におけるチーム活動では，プロジェクト遂行能力として，スケジュールの管理が重要である．特に，開講期間の限られている PBL では，授業時間外での活動状況が成果に影響を与える．コミットが行われた時間帯を集計し，休日や深夜にコミット数が多いプロジェクトは，スケジュール管理に問題があるという評価を下すことができる．

3.2.3 開発分担とタスク量

PBL によるチーム開発では，全メンバへ役割を割り当て，学習機会を等しく与えることが重要となる．特定のメンバのみで開発の負担が偏っていないことは，PBL におけるチーム活動評価の重要な指標となりうる．コミットを著者別にカウントし，メンバ間で偏りがないかを見ることで，適切な役割分担と平等な貢献度合いを評価することができる．

4 分散 PBL に対する試行

本チーム活動過程評価手法の有効性を評価することを目的として，2016 年 8 月に実施された分散 PBL に対して試行した結果について述べる．対象とした分散

[5] https://slack.com

表 2: チーム内のメンバ構成

	X 大学	Y 大学	Z 大学
A チーム	3 名	2 名	
C チーム	3 名		2 名
F チーム	2 名		2 名

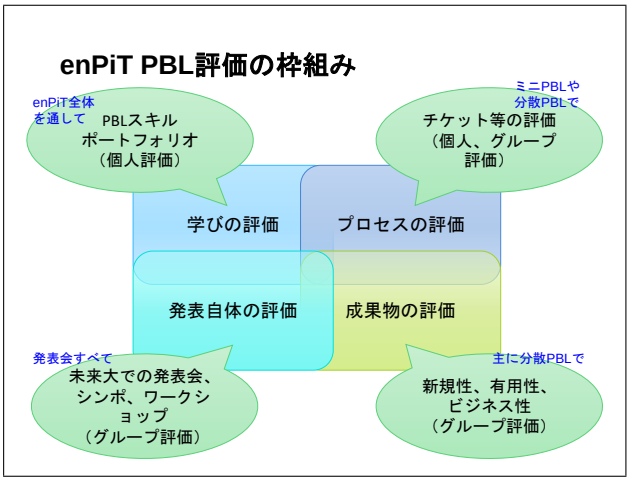

図 2: ガイダンスで使用した PBL 評価に関する資料

PBL は，地理的に分散した 4 つの大学の大学院学生が参加し，共通の課題に対してシステム提案とプロトタイプ実装を行う内容で，5 日間（月曜日から金曜日）の集中形式で行われた．初日（月曜日）に行った 30 分程度のブリーフィングを除いて，月曜日から木曜日までは 9 時から 17 時半頃迄，最終日の金曜日は 9 時から 12 時迄を各チームが共同作業できる時間とした．各日の最後には，チーム毎の進捗報告を行い，PBL 全体への情報共有を図った．最終日（金曜日）午後には，成果に関するプレゼンテーションをチーム毎に行い，PBL 参加学生と教員による相互評価を行った．実在する保育園に対して担当教員が事前にヒアリングした結果を基にして，現在手作業で行っている卒園児向け情報発信の IT 化による支援をテーマとした．卒園児家族向けのハガキ作成と郵送による情報発信作業に対して，IT 技術で支援が可能な部分の検討からその解決法までを各チームで検討し，ソフトウェアとして実装することを課題として取り組んだ．表 2 に示すように，複数の大学の学生が混在する 4-5 名のチームを 6 つ編成した．各チーム活動におけるコミュニケーションは，開発成果の管理と共有を行うための Git リポジトリのクラウドサービス GitHub，文字によるチャットやファイル交換のためのコミュニケーションアプリ Slack[5]，Polycom

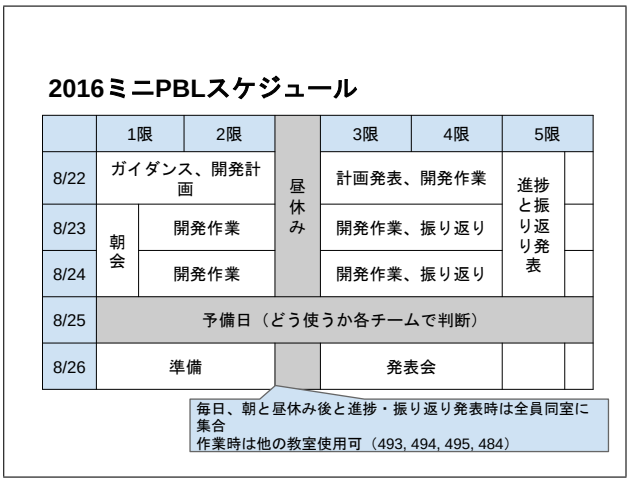

図 3: ガイダンスで使用したスケジュールに関する資料

テレビ会議システムを用いた．

実装に際して，GitHub 上にチーム毎に 1 つのリモートリポジトリを作成し，各自の PC 上のストレージへリモートリポジトリをクローンしたローカルリポジトリを作成した (図 1 参照)．各メンバは，開発したプログラム等の成果物をローカルリポジトリに作成したブランチ上でコミットとして登録していき，適宜，GitHub 上のリモートリポジトリの対応ブランチへプッシュすることでメンバ間での開発内容と進捗の共有を行った．プッシュされたコミットは，GitHub 上でマスターブランチへマージした後，他メンバのローカルリポジトリからプルすることで反映を行う．

本 PBL における評価に際して，初日のガイダンス時，図 2 に示す資料を用いて，PBL の評価にプロセス（チーム活動）の評価があることを説明した．加えて，最終日に実施する相互評価の評価項目（表 3 参照）をあらかじめ示したうえで，成果プレゼンテーションを準備するよう指示した．また，Git および GitHub Flow を説明した書籍 [4] とアジャイル開発に関する書籍 [3] を複数冊用意して，事前に各大学拠点へ貸与することで，PBL 開始前にチーム開発に関する自習を行うことを促した．加えて，初日ガイダンス時にも，チーム開発におけるタスク管理や成果物のバージョン管理の重要性を指摘した．開発スケジュールに関しては，初日のガイダンス時に図 3 の資料を用いて，PBL 期間内の開発作業時間帯を示した．PBL 期間中は，定期的に更新した GitStats の統計情報と各チームリポジトリの総コミット数を示すグラフを公開することで，Git リポジトリを観察していることを明示した．

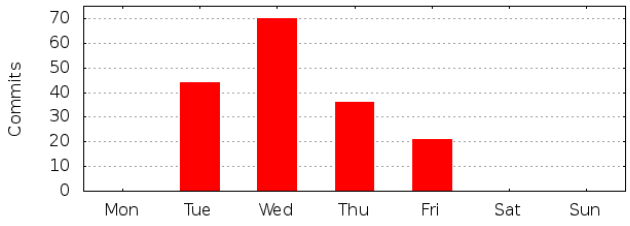

図 4: A チームのマスターブランチにおける日別コミット数

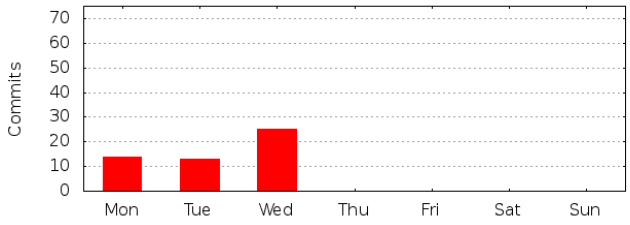

図 5: C チームのマスターブランチにおける日別コミット数

以降では，それぞれのチームについて，コード開発履歴の可視化結果と前章で述べた方式に基づいて推定したチーム活動について述べる．

4.1 マスターブランチのコミット数推移

チーム別のマスターブランチのコミット数の曜日別に集計したグラフを図 4，図 5，図 6 に示す．各グラフの横軸は PBL を実施した期間の各曜日，縦軸が各日におけるコミットの数である．

図 4 のチーム A の結果では，初日にコミットはないものの，2 日目から 40 を超えるコミット数があり，最終日の午前まで積極的なコミットが行われていることが読み取れる．チーム A のマスターブランチにおける総コミット数は 171 で，全チームのなかで最も多い．3 日目のコミット数が最も多く，その翌日からコミット数が減っている．Git リポジトリのコミットメッセージを確認してみると，3 日目から修正に関するコミットの割合が増えていっている．このことから，3 日目から実装から動作確認やデバッグへ作業のフェーズが移行していっていることが読み取れる．

図 5 のチーム C は，初日からコミットがあり，3 日目に最大数のコミットが行われていることからも，早期に実装を始めて，3 日目で開発を完了させようとしている意図が読み取れる．

図 6 のチーム F は，他チームと比べて，最終日のコミット数が最も多く，成果プレゼンテーション直前まで開発を行っていたことが推測できる．

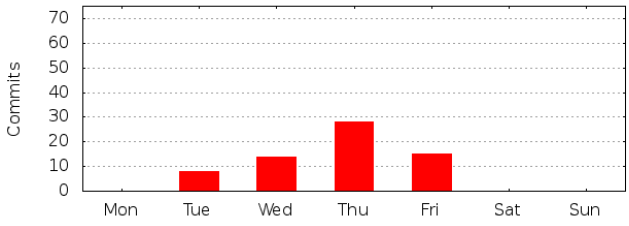

図 6: F チームのマスターブランチにおける日別コミット数

Weekday	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Mon																								
Tue										1	7	4	2	6	11	9	3		1					
Wed										10	10	10	9	12	1	8	8	1	1					
Thu											2	14	8	3	1	6	2							
Fri										7	8	6												
Sat																								
Sun																								

図 7: A チームの時間帯別コミット数

4.2 時間帯別のコミット数

図 7，図 8，図 9，は，マスターブランチに登録されたコミットをその登録時間帯に沿って集計した結果である．各マトリックスにおける行は PBL 実施期間の曜日，列がコミットを登録した時間帯を示している．コミットの曜日および時間帯は，各メンバのリポジトリに登録したときのもので，主リポジトリのマスターブランチへマージする際も変更されない．マス目の数字は，対応する行の曜日と列の時間帯に登録されたコミット数である．空欄のマス目は，コミットの登録がひとつもなかったことを示している．

図 7 の結果から，A チームのコミット時間が 2 日目 (火曜日) から最終日 (金曜日) 午前に渡って平均的に分布していることがわかる．また，すべてのコミットが 9 時から 18 時の間に行われており，与えられた授業時間内のみの活動で計画的に開発を進めていたことが推測できる．

一方，図 8 の C チームでは，4.1 節のチーム C の活動推測で述べた通り，3 日目までで開発を完了させようと計画しているために，初日から 3 日目まで 18 時以降の授業時間外も開発を継続していたことが読み取れる．

図 9 では，4.1 節で述べた F チームの活動推測を裏付けるように，4 日目の授業時間後から最終日の朝までの

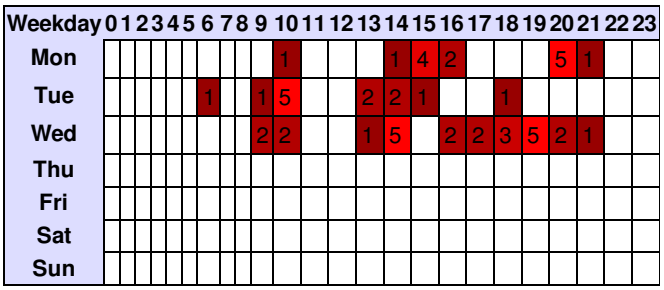

図 8: C チームの時間帯別コミット数

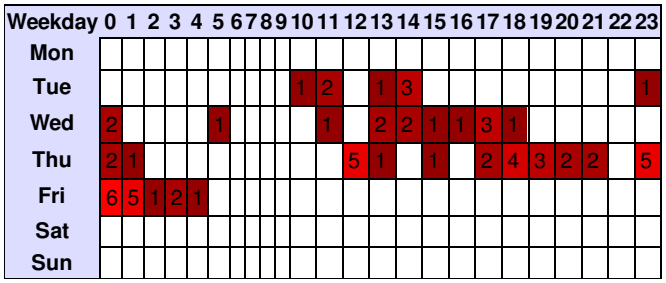

図 9: F チーム時間帯別コミット数

コミット数が一番多い．この時間別コミット数の統計から，成果プレゼンテーションに間に合わせるために，最終日前日に徹夜で開発を進めたことが読み取れる．

また，どのチームも，金曜日の 13 時以降，土曜日および日曜日にコミットがない．金曜日の午後に行った成果プレゼンテーションでは，全チームが開発したソフトウェアのデモンストレーションを実施している．このことから，すべてのチームが予定通りの開発を期限までに完了しており，プレゼンテーション開始以降に実装の改善や残実装の着手など行われていないことがわかる．

4.3 著者別コミット数の推移

図 10，図 11，図 12 は，マスターブランチに登録されたコミットをその著者別に集計し，その推移を可視化したグラフである．各グラフの横軸が PBL 実施期間の各日，縦軸が累積コミット数を示している．各ラインがコミットの著者を示しており，GitStats が出力する結果に含まれる著者名の記載を削除している．なお，図 10 のグラフの基データでは，開発中に Git の著者設定の誤りを訂正したことにより，チームメンバ数よりもコミット著者数が多くなっているが，グラフ生成前に基データに対して手動で名寄せを行っている．

図 10 の A チームでは，メンバ全員のコミット数が毎日平均的に増えており，均一的な開発分担により，チームメンバ全員が開発に貢献できていることが読み取れる．また，コミット数が増加するポイントが毎日存在し，各メンバのコミットがマスターブランチに統合され，メンバ間での共有が行われていることが推測できる．

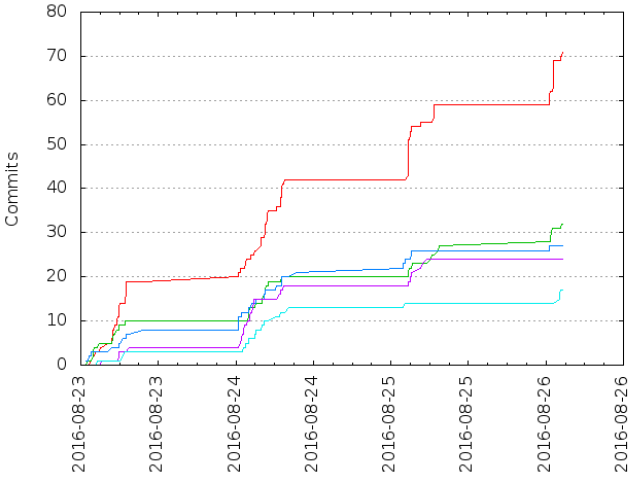

図 10: A チームの著者別コミット数の推移

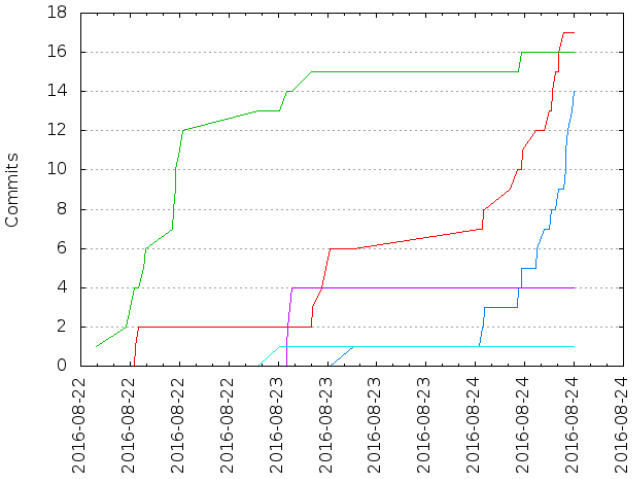

図 11: C チームの著者別コミット数の推移

図 11 の C チームでは，期間前半のコミット数が特定のメンバに対してのみ増加しており，前半の開発を主導しているメンバが存在していることが推測できる．期間後半は，他のメンバのコミットが増えていっているが，コミット数が増加しないメンバも 2 名存在することから，開発に関する分担に均一性がないことが示されている．

F チームの図 12 からは，4.2 節で述べた 4 日目授業時間後からの翌日朝にかけて，4 名中 3 名のコミットが大きく増加している．このことから，4 日目夜からの徹夜での開発は，主に 3 名で行われていたことが推測できる．

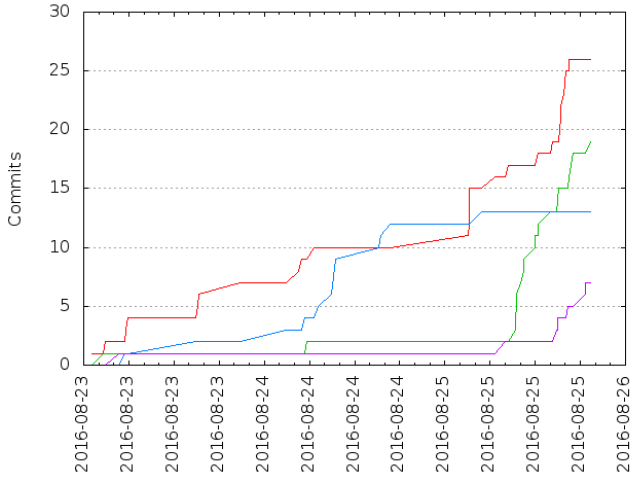

図 12: F チームの著者別コミット数の推移

4.4 最終成果物による相互評価結果との比較

本学におけるこれまでの分散 PBL では，最終プレゼンテーションによる相互評価結果を総合評価のための指標として参照してきた．本節では，本手法による Git 変更履歴可視化結果から推測したチーム活動評価と従来の相互評価結果との比較を行う．最終プレゼンテーションによる相互評価では，分散 PBL 最終日において，開発した成果物のデモンストレーションを含めたプレゼンテーションをチーム毎に行い，他のチームの学生や PBL 担当教員により複数の評価項目に関して各 5 段階[4]で採点を行った．なお，総合評価としては，評価項目毎の平均点を算出した結果を合計した得点が高いチームを優秀チームとして表彰した．

分散 PBL におけるチーム開発の進め方に関する相互評価結果を表 3 に示す．前節までのコード更新履歴による評価では，A チームの活動が優れているという結果が得られていたが，相互評価の結果においても同様に，高い評価点が得られている．また，C チームの相互評価結果が A チームと比べて低い．しかし，前節までに示したコミットの推移からのチーム活動の評価では，A チームと C チームの開発活動の傾向は大きく異なる．本相互評価は，チーム活動を観察していた担当教員も参加していたが，分散 PBL におけるこれらのチーム活動の違いが相互評価では十分に反映されない場合があることが読み取れる．F チームに対する相互評価結果は，かなり低い点となっている．前節までで示した GitStats の可視化統計情報に基づいたチーム活動推定においても，計画的な開発進捗や開発分担などの点で課題があることを指摘しており，相互評価においても同様の結果となった．

[4]最低点を 1，最高点を 5 とした．

表 3: チーム開発の進め方に関する相互評価結果

A チーム	C チーム	F チーム
4.41	4.04	3.43

5 関連研究

文献 [11] の研究は，能力度成熟度モデル統合 (CMMI) に基づいて，PBL におけるプロジェクトのプロセスを評価する手法を提案している．チーム活動のプロセスを評価する手法として関連しているが，本提案手法では，プロジェクトの作業過程で生成される定量的なメトリクスのみで評価を行うため，評価項目毎の議論や振り返りなどの追加作業を必要としない．

文献 [6] では，分散 PBL 等の異なる大学のカリキュラムを履修する受講生に対して，共通の評価指標によって客観的な評価を行う手法を提案している．開始前と終了後に実施する PROG テストが共通の評価指標となっているが，本提案手法では，評価のための作業を追加せず，プロジェクト内の開発で使用するリポジトリを共通のメトリクスとして利用する点が異なる．

本提案手法と類似して，ソフトウェア開発型の PBL における Subversion リポジトリやプロジェクト管理ツールに着目して，チーム進捗を把握する研究も行われている [2, 7, 8, 9, 13]．PBL の運用支援や管理情報の記録・可視化などを目的とするこれらの研究と比較して，本研究では，特に，Git リポジトリが持つ著者別のコミット日時やその登録順，および，マスターブランチの運用に着目して，教員が直接把握することが難しい分散 PBL のチーム開発活動状況を把握することを目的としている．

文献 [10] は，PBL の評価に関する研究ではないが，ソフトウェアの品質を評価するレビューの効率化を目的として，プロジェクトやプロダクトの状態を間接的に表すメトリクスを計測することで，欠陥の偏在箇所や種類を予測する手法を提案している．用いるメトリクスとして，ファイル更新時間や設計書の更新回数のようなソフトウェアに直接関連するものから，飲み会の頻度やディスプレイの大きさなど，ソフトウェア開発

プロジェクトの環境に関するものまで幅広い範囲の定量的な値からの予測を行っている．ソフトウェア等の成果物を直接評価するのではなく，間接的で定量的なメトリクスを用いてプロジェクトの状態を推定して評価する点で，本提案手法と関連が深い．

6 おわりに

本論文では，分散 PBL におけるチーム活動状況の把握を目的として，Git リポジトリに対するコミット履歴の可視化を提案した．Git は，ソフトウェアの分散開発で広く用いられているコード管理ツールであり，その関連ツールも数多く存在している．本手法では，オープンソースの Git リポジトリ状況可視化ツール GitStats を用いて，コミットの登録日時や数、コミットを行ったメンバに関する統計情報を可視化する．その可視化結果から，PBL におけるチーム活動の状況や経過の把握を試みる方法について提案した．さらに，実際の分散 PBL において，Git リポジトリへのコミットを可視化し，その特徴からチーム活動過程を推定する試みの結果について述べた．

今回の試行では，チーム構成メンバの Git 習熟度の違いから発生する影響について考慮できていない．Slack を使った学習の振り返り記録では，A チームの 2 名のコメントに Git の習得や活用に苦労した旨の内容が含まれていた．本手法を用いてチーム間やメンバ間の比較評価を行うためには，事前講習会や自主学習を課すことで，Git 習熟度の影響を小さくした状態で PBL を実施する必要がある．また，今回の試みでは，チーム活動過程が個々の学びに与える影響について評価できていない．円滑なチーム活動が個々の学びに対しても良い影響を与えることが期待されるが，本アプローチにより把握したチーム活動過程と個人の学びとの関係性について評価を行いたい．さらに，本手法が解析対象とした Git リポジトリのコミット履歴では，開発以外のチーム活動過程を把握することが難しい．今後，Slack や GitHub の issue 機能などの履歴を組み合わせて，設計や最終方向などの開発以外の活動についての評価方法を検討する．

参 考 文 献

(1) GitHub.com. Understanding the github flow. https://guides.github.com/introduction/flow/, last accessed on Jan. 15, 2018.

(2) U. Kohichi, H. Igaki, Y. Higo, and S. Kusumoto. A study of student experience metrics for software development pbl. In *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 465–469, Aug 2012.

(3) 西村直人, 永瀬美穂, 吉羽龍太郎. SCRUM BOOT CAMP THE BOOK — スクラムチームではじめるアジャイル開発. 翔泳社, 2013.

(4) 大塚弘記. GitHub 実践入門 — Pull Request による開発の変革. WEB+DB PRESS plus. 技術評論社, 2014.

(5) 美馬のゆり. 大学における新しい学習観に基づいたプロジェクト学習のデザイン. 工学教育, Vol. 57, No. 1, pp. 45–50, 2009.

(6) 山本雅基, 小林隆志, 宮地充子, 奥野拓, 粂野文洋, 櫻井浩子, 海上智昭, 春名修介, 井上克郎. enpit における教育効果測定の実践と評価. コンピュータ ソフトウェア, Vol. 32, No. 1, pp. 213–219, 2015.

(7) 伊藤恵, 木塚あゆみ, 奥野拓. 過去の PBL の開発履歴を活用した PBL 運用支援. 日本ソフトウェア科学会大会論文集, Vol. 31, pp. 249–260, 2014.

(8) 井垣宏, 柿元健, 佐伯幸郎, 福安直樹, 川口真司, 早瀬康裕, 崎山直洋, 井上克郎. 実践的ソフトウェア開発演習支援のためのグループ間比較にもとづくプロセスモニタリング環境. 日本教育工学会論文誌, Vol. 34, No. 3, pp. 289–298, 2010.

(9) 梅川晃一. ソフトウェア開発 pbl におけるオンラインストレージを用いた学生評価メトリクスの提案. 学位論文, 大阪大学基礎工学部情報科学科, 2012.

(10) 細川宣啓, 永田敦, 森崎修司, 中谷一樹, 諏訪博紀, 田邊哉好, 森崎一邦, 末次努, 小田部健, 山本浩之, 牧野将治, 小原美帆, 奥山剛. 間接的メトリクスを用いて欠陥予測を行うレビュー方法の提案. ソフトウェア品質シンポジウム 2011 発表報文集, pp. 1–8, 2011.

(11) 日戸直紘, 伊藤恵, 大場みち子. 能力成熟度モデル統合に基づいた pbl における定量的学習評価手法の提案. 日本ソフトウェア科学会大会講演論文集, Vol. 34, pp. 1–7, 2017.

(12) 文部科学省. 情報技術人材育成のための実践教育ネットワーク形成事業：分野・地域を越えた実践的情報教育ネットワーク, 2012. http://www.enpit.jp.

(13) 矢ケ崎隆磨, 井垣宏, 田胡和哉. チケット駆動開発を適用したグループ並行型 pbl のための開発履歴可視化・分析システム. 情報処理学会第 73 回全国大会講演論文集, Vol. 2011, No. 1, pp. 539–540, mar 2011.