特別研究報告

題目

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

指導教員
楠本真二教授

報告者
梅川 晃一

平成 24 年 2 月 16 日

大阪大学 基礎工学部 情報科学科

平成 23 年度 特別研究報告

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

梅川 晃一

内容梗概

近年，ソフトウェアの不具合を起因としたトラブルが増加し，深刻な社会問題となりつつある．その原因としてソフトウェアを構築するための人材が質，量ともに不足していることが指摘されており，高度なソフトウェア技術者や高度 ICT(Information and Communication Technology) 人材の育成が求められている．その中で高度 ICT 人材の育成を目指して，PBL(Project-based Learning，課題解決型学習) と呼ばれる教育・学習手法が様々な形態で行われている．しかし PBL は，グループ内でのタスクの分担比率や学生個人の学習経験を教員が把握し切れない，という問題を抱えている．

本論文ではソフトウェア技術者育成のための PBL における定量的な学生評価メトリクスを提案する．また，学生のグループ開発における学習内容とエフォートをオンラインストレージを用いて自動的に取得する手法と，学生評価メトリクスの可視化と運用の例も合わせて提案する．本論文で提案する作業時間と作業量の各メトリクスによって，各学生が成果物の開発に要した時間と，開発した量の定量的な評価が可能となる．

評価実験として，実際に複数の学生を対象として成果物ごとの開発量と開発に要した時間が計測可能であることを確認した．

主な用語

Project-based Learning(PBL)，オンラインストレージ，開発者メトリクス，グループ開発，ソフトウェア技術者教育

目 次

図 目 次

1　まえがき

近年，家電や携帯電話といった身近な製品から，交通や金融等の社会基盤に至る幅広いシステムまで，ソフトウェアの不具合を起因としたトラブルが増加し，深刻な社会問題となりつつある．その原因としてソフトウェアを構築するための人材が質，量ともに不足していることが指摘されており，高度なソフトウェア技術者や高度ICT(Information and Communication Technology) 人材の育成が求められている [1][2]．高度 ICT 人材の育成には，技術的能力に加え，マネジメント能力やコミュニケーション能力，問題発見・解決力など，総合的な能力についての教育が重要である [3].

このような総合的な能力は従来の講義形式の教育体系では十分な教育が難しいと言われている．その中で，情報系の大学では実際のソフトウェア開発に基づく PBL(Project-based Learning，課題解決型学習)[4] と呼ばれる教育・学習手法が様々な形態で行われている．これは高度 ICT 人材の育成を目指した教育体系であり，既に高い教育効果が確認されつつある [5]．PBL では，学生に少人数のグループで与えられた課題解決型タスクに基づいた実践的な成果物の開発を体験させ，主体的かつ自律的な学習機会を提供する．特に講義や実習では得ることの難しい学生間のコミュニケーションやプロジェクトマネジメント等の学習機会を与えるという観点を持つ学習手法である．

Batatia(2001) によると，PBL は以下の 2 つの側面を持っている [6].

D1 内容 (subject-matter knowledge) に関する学習

D2 プロジェクトマネジメントおよび総合的な能力の学習

教員および学生のアクティビティはこれら 2 つの側面を教員と学生のどちらがどの程度制御するかによって大きく異なっており，D1，D2 双方を学生に委ねるタイプの PBL は実施が最も困難であるとされている．

PBL のうちソフトウェア開発を扱うものをソフトウェア開発 PBL と呼び，以下ではソフトウェア開発 PBL に関して議論を行う．教員側のアクティビティのうち，D2 における学生の役割分担やスケジュールマネージメント等のプロジェクトマネジメントについての指導・補助が難しいと考えられている．その一因として，教員にとってグループの開発状況を常時監視するのは難しく，それ故にグループ内でのタスクの分担比率や学生個人の学習経験を把握し切れない，という点がある．また，教員がグループの開発状況を把握しようとすると，グループメンバーのアクティビティについてのデータ収集作業を行う必要があり，その結果学生と教員双方の負担が増してしまうという問題が新たに発生する

以上の問題を解決するため，本研究の目的を学生毎に作業の種類・量を人的コストを極力かけずに行うことを目標とした．アプローチとして，オンラインストレージを用いて学生の作業に関するデータを収集する．また，収集したデータを用いて作業の種類・量の定量化のために学生評価メトリクスを計測する．そして計測したメトリクスをグループ内で比較することで，作業分担に関する情報を可視化する．

本報告では，ソフトウェア開発 PBL における各学生が行った具体的な作業の種類と量 (エフォート) をオンラインストレージによって自動的に収集し，開発者メトリクスとして分析・可視化する手法を提案する．学生毎のエフォートの可視化と比較を自動化することで，コストを掛けずに学生個人の評価を行うことが容易となる．

以降，2 章で準備としてソフトウェア開発 PBL や学生の評価に関する既存研究について述べる．3 章では採用するキーアイディアと用いる開発者メトリクスの定義，可視化方法について述べる．4 章において作成したアプリケーションの概要と利用例について述べ，5 章では作成したアプリケーションを用いて行った実験の結果とその考察について述べる．最後に 6 章で本研究のまとめと今後の課題について述べる．

2　準備

2.1　PBL

2.1.1　一般的な PBL

PBL は学生に対し自発的な学習を促し，従来の授業では得られない問題解決能力を獲得してもらうという目的の下，分野を問わず活発に行われている．PBL として捉えられる学習は幅広く，それぞれの研究で独自の定義がなされており，統一された見解はない．いくつもの PBL の定義を包括する形で，一般的な PBL の特徴は以下のようにまとめられている [7].

- PBL は実社会における本質的な問いや問題 (driving question) からスタートする
- PBL ではプロジェクトがカリキュラムの中心である
- PBL には授業の時間枠を超えた様々な活動が含まれ，それらは本質的な問いや問題への答えを探究するためにデザインされている
- PBL では学生や教員，グループ・メンバー間での協調活動が要求される
- PBL では学生は主体的，自主的に活動に取り組むことが求められ，教員はそのための補助役として行動する
- PBL では学生の協調活動や研究，分析の能力を伸ばすためにテクノロジーを利用する
- PBL ではしばしプロジェクト成果を実社会に向けて発表するよう求められる．

PBL の学習効果として，出席数の増加，自立心の成長，学習態度の改善 [8] や，高度な思考力，問題解決能力，共同作業，コミュニケーションなどの複雑なスキルを伸ばす可能性が示唆されている [9].

一方で，同じ PBL という略称を持つ教育手法に Problem-based Learning というものが存在する．Problem-based Learning においては，教員が用意した現実に即する問題を持ったシナリオを対象としてグループ内で司会者，記録者，発表者といった役を決めて議論を行う．同時に自己学習やグループ学習を行い，シナリオに関する知識を収集する．

どちらの PBL においても，グループワークを行う中で自身の学習を管理し，協調的に学習に取り組むことを目標とする一方で，以下のような違いを持つ [10].

Project-based Learning　プロジェクトにおける社会的な生産活動を目的とするため，知識の適用により主眼が置かれる．

Problem-based Learning　問題の発見と解決のための思考と学習のプロセスに大きな比重が置かれるため，新しい知識の獲得により主眼が置かれる．

同じ略称を持つ2つの教育手法のうち，本論文ではPBLという略称をProject-based Learningに限定して用いる．

2.1.2　ソフトウェア開発 PBL

PBLの中でも題材としてソフトウェア開発を扱うものをソフトウェア開発PBLと言う．以降，本論文ではソフトウェア開発PBLをsPBLと呼称する．sPBLは一般的なPBLの特徴に加えて，プロジェクトを構成する学生のグループに対し，以下のような課題が与えられることがある [11].

T1 仕様を満たしたプロダクトが期間内に計画的に実装できているか

T2 グループ内でタスクが適切に分担されているか

T3 必要なタスクを十分に実施できているか

sPBLは，開発をより円滑に進めるためのプロジェクトマネジメントやコミュニケーションの方法を学生が自ら工夫し習得することを目標とした教育手法である．そのため，学生のグループはT1を目標として，プロダクトの実装に関してマネジメントを行い，開発計画を立てる必要がある．そしてT2としてグループ内でコミュニケーションを積極的に行い，メンバーでタスクを平等に分担して作業を行う．タスクの分担に関しては，円滑な開発を目標とするだけでなく，各学生がグループワークにおける学習の機会を平等に与えるという目的を含んでいる．また，sPBLでは開発において特定の手法を利用することを求められる場合がある．その場合T3として，開発だけでなくそれ以外の活動に関するタスクの実施も求められる．

これは，sPBLの目的が純粋な開発演習だけではなく，学習の過程を重要視する教育という側面を持つためである．

ここで，既存のsPBLの実施事例をいくつか挙げる．

ITSpiral[12][13]　ITSpiralに関する説明として，公式で紹介されている文面を引用する [14].

> 情報通信技術、特にソフトウェアの高度な技術者育成を目標とし，ソフトウェア工学分野で教育・修得すべき内容をより豊富にかつ体系的・実践的に教育課程に取り込むため，関西圏の情報系9大学院に分散している該当分野の卓越した

専門家群を結集し，融合連携型専攻の構築を進めています．特に重要視する実践的教育については，参画企業と協働して，教科書的例題ではなく現実の開発プロジェクトそのものを教材として開発し，適用していきます．

ITSpiral では，実際の開発プロジェクトに準じた演習を通じ，1 章で説明した D1 の側面である開発対象ソフトウェアの決定は学生に行わせず，D2 を受講生が主体的かつ自律的に学習することを目的としている．また，ITSpiral では開発プロセスとして要求分析，実装，テストまでの一連の開発プロセスを経験させる．

芝浦工業大学工学部情報工学科 [15]　この学科で行われている「高度情報演習 1B」では，学生に文書作成・プログラム開発・検査・プロジェクト作業の苦労と達成感を実感してもらうことと，これからの情報技術者に求められる創造性と問題発見能力・問題解決能力の育成を目的として sPBL が行われている．この学科でも ITSpiral と同様に D1 に関する決定は学生に行わせず，D2 に関して学生に求める形でグループワークが進められる．経験させる開発プロセスも ITSpiral と同様に，要求分析，実装，テストの工程を経験させる．

東洋大学総合情報学部総合情報学科 [16]　この学科で行われている「総合情報プラクティス VI」において，学生は 26 テーマから好きなものを一つ選択し演習に臨む．その 26 テーマの一つとして実施される「情報科学系・ソフトウェア工学」において sPBL が行われている．この授業も D2 に関してのみ学生に求める形の sPBL である．ただしこの授業では，学生はプログラミング能力やソフトウェア工学に関する技術についてカバー出来ているという前提を置いている．そのため，sPBL で行う演習はモジュールの統合やソフトウェアテスト，プロジェクト管理の実践に重点が置かれている．

2.2　sPBL 実施環境

sPBL の実施環境は教育現場毎に異なる特色を持つ．多くの sPBL において設定されている環境を以下に示す．

- プロジェクトのプロダクトを構築する際に学生が利用する開発環境
- 学生の開発プロセスを支援する開発支援環境
- 教員による学生の開発プロセス解析を支援する教育支援環境

上記 3 つを総括して sPBL 実施環境と呼称する．開発環境は，Java SE などの開発・実行環境と，一般的なテキストエディタや統合開発環境である Eclipse 等を合わせた実際にプロ

グラム開発を行うための環境を指す．開発支援環境は，ファイル管理やタスク管理などによって円滑な開発作業を支援するための環境を指す．また，グループメンバー間の連絡手段に関しても開発支援環境に含める．教育支援環境は，教員によるグループの開発状況の確認や，最終的な学生・グループの評価や sPBL 自体の分析を行うためのソフトウェアを合わせたものである．

まず，実際に行われている sPBL ではどのような sPBL 実施環境が用いられているかを紹介する．

2.2.1 sPBL 実施環境の詳細

既存の sPBL で用いられることの多い開発支援環境および教育支援環境について説明する．

バージョン管理システム バージョン管理システムは，アプリケーションの開発中に開発者が作成する様々な種類のファイルを，全てのバージョンにわたって保管する場所を提供するシステムである [17]．バージョン管理システムはファイルのバージョンを保存し，ファイル内容や更新者などの情報を用いたファイルの管理，問題が発生した場合などに過去のファイル内容を取得する機能などを提供する．また，複数人の管理者が一つのファイルに対して，管理された方法で同時に作業できるようになる．

sPBL においては，学生側に対してはプロジェクトメンバー間で成果物を共有するための開発支援環境，教員側に対しては成果物や差分を外部から確認することを可能にする教育支援環境として機能する．

チケット管理システム チケット管理システムとは，チケット駆動開発と呼ばれるプログラム開発手法を行うための環境を提供するシステムの総称である．チケット駆動開発 (TiDD)[18] とは，プロジェクトの中で開発者が行うタスクをチケットと呼ばれる単位に分けて開発を行うソフトウェア開発手法である．ここでタスクは設計・実装といった各開発工程において，一人の開発者が短時間で行える作業レベルまで分割されたものである．チケットには

- チケットの概要
- 分割されたチケットに相当するタスクの内容
- タスクの対象となる成果物
- 開発者
- 見積り工数

- 開発工数

等の情報が保持される．分割されたタスクをチケットに割り当てることで，タスクの内容確認や分配が行いやすくなり，結果としてプロジェクトの進行と開発者のタスク管理を円滑に行うことができる．

ITSpiral で用いられた環境でのチケット駆動開発におけるタスクの割り当てから完了までの流れの一例を図 1 に示す．

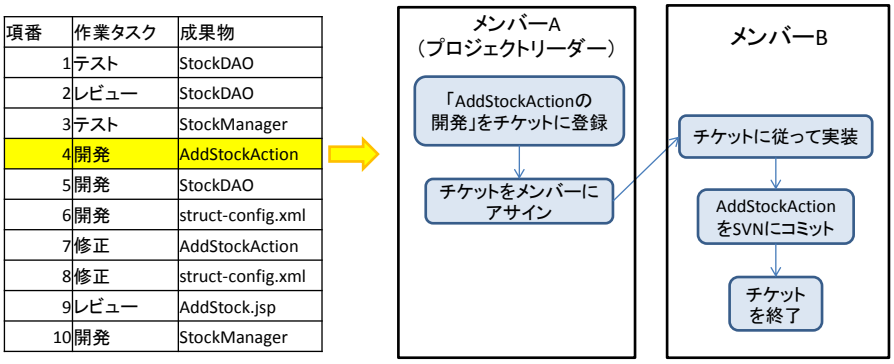

図 1: チケット駆動開発におけるタスク完了までの流れ

図 1 では，プロジェクトに対しタスクの抽出・整理を行った結果，AddStockAction を実装する必要があると判明した．この時，プロジェクトリーダーを担当するメンバーは「AddStockAction の開発」というタスクをチケットに登録する．登録されたタスクは他のメンバーにアサインされ，そのメンバーによってタスクが消化される．

sPBL においてチケット管理システムは学生側に対してプロジェクトにおけるタスクの管理とプロジェクトメンバー間のタスク分担をスムーズにする開発支援環境，教員側に対してはチケットに保持された情報を収集することによるプロジェクトの遅延や学生個人単位での進捗確認等のグループの開発プロセスに関する情報収集を支援する教育支援環境として機能する．

2.2.2 sPBL の実施環境例

ITSpiral ITSpiral では実務経験の無い教員でも以下の活動を行う事ができるような sPBL 実施環境の構築を行った．

1. タスク単位での進捗管理

2. 1. に伴うプロダクト，プロセスメトリクスの収集および教員によるモニタリング

3. 2. で行うモニタリング結果のグループ間比較

具体的には，sPBL 実施環境として表 1 に提示したシステムを学生に利用するよう指示し，sPBL を行った．

開発・実行環境	Java SE Development Kit
統合開発環境	Eclipse
バージョン管理システム	SVN(Subversion)
チケット管理システム	Trac

表 1: ITSpiral で提供された sPBL 演習環境

加えて，グループ内での開発状況把握や連絡を行うためのツールとして，ML(Mailing List) を学生に提供した．これらのうち，Java SE と Eclipse は開発環境，ML は開発支援環境に含まれる．SVN と Trac に関しては，開発支援環境と教育支援環境のどちらとしても機能する．

芝浦工業大学工学部情報工学科　芝浦工業大学では，開発環境と開発支援環境，教育支援環境を組み合わせた独自のツールとして EtUDE を開発した．EtUDE は「sPBL のグループワークにおいて必要なすべての作業が，時間と場所の制約を受けずに実施できるようなツール」を目指して，複数のソフトウェアプロダクトを組み合わせて開発されている．その一部を表 2 に提示する

開発・実行環境	Java2 Platform Standard Edition
統合開発環境	Eclipse
バージョン管理システム	CVS(Concurrent Version System)
メール送信ライブラリ	ozacc-mail library

表 2: EtUDE の構成ソフトウェア (一部)

表 2 の構成ソフトウェアにより，EtUDE では

- 中間成果物のバージョン管理を容易にするとともに，プロジェクトメンバーが中間成果物を互いに自由に閲覧できるようにする機能

- プロジェクトメンバー同士でのコミュニケーションを支援する機能
- 活動ログの自動取得機能

を実現している．

2.3 sPBL における評価

2.3.1 既存の評価事例

sPBL では，プロジェクトを構成するグループ単位での評価と，グループ内の学生を評価する個人評価が行われる．一方で，個人評価は定量的に行われている事例がほとんどなく，アンケート等を用いた定性的なものがほとんどである．実際に行われている個人評価に関する事例を以下に示す．[19].

- 学生の評価をチーム内で統一する
  個人の学習を評価するのは難しいと判断して，評価をチーム単位に限定し個人評価を行わないパターンである．この場合，グループ内で負担が大きかった学生にとっては不平等な評価となってしまう．また別パターンとして，PBL に最後まで参加した学生には全て”優”評価を与えている学校も存在する．

- 教員の主観による評価を行う
  担当の教員目線で，授業の理解度やグループワークへの貢献度等を評価するパターンである．このパターンでは，評価基準について事前のコンセンサスを取っていても最終的には教員の主観になってしまうため，グループワーク全体で共通した評価を行うことが難しい．また，教員がグループメンバーそれぞれを注視しなければならないため，教員に負担がかかることになる．例に挙げた東洋大学は教員によるグループ評価と個人評価を 7:3 の割合で合計したものを最終評価としている．

- 筆記試験，レポートを課す
  PBL 終了後，筆記試験やレポートを用いて学生の授業に対する理解度を問うパターンである．このパターンでは，学生の知識として得たものは評価できる．しかし，この後で提案する手法でも判断が出来ない内容とはいえ，この方法では PBL を行う際に求められるコミュニケーション能力や，問題発見・解決力についての経験を評価できるとは限らない．

- 学生の開発内容を記録するシステムを用いた評価を行う
  ソフトウェア開発において，開発内容を記録できるツールを開発支援環境，教育支援

環境として用いて学生評価基準の一環としている．ITSpiral では Trac と Subversion，芝浦工業大学では独自の sPBL 実施環境 EtUDE を用いて学生の開発内容を記録し評価を行った．ただし，どちらの大学も定量的な評価基準は設けておらず，評価の目安として用いられているのみである．

2.3.2 開発プロセス評価に関する既存研究

開発者の作業に関するプロセスの定量化を行った既存研究を以下に提示する．

PSP(PersonalSoftwareProcess)[20] PSP とは個人のソフトウェア開発へのアプローチを訓練し，構築するための開発プロセス改善法である．開発者個人の開発時間，編集行数，欠陥数などを測定し自己評価，自己改善を進めていくための構想とメソッドを提供する．厳格に規定された開発プロセス遂行のための手続きと，開発プロセス記録・分析のための表現形式が用意されている．これらを用いたプログラム開発を通して，開発プロセスの改善を行う，開発者個々のソフトウェア開発能力向上を目的とした手法である．

この手法を補助するツールとして Hackystat が存在する．Hackystat は sensor と呼ばれる計測用プラグインをエディタなどの各種開発ツールに導入することで，

- ファイルを開いた
- ファイルの名前を A から B に変更した
- プログラムを実行した

等，ツールを使用する際の行動を計測することができる [21]．

PSP の補助ツールである Hackystat は開発プロセスに関する収集，解析，可視化，解釈，注釈など多くの機能を提供する．ただし，開発ツールに導入するプラグインという形を取るためどのような環境でも用いることができるわけではない，という問題がある．

松浦佐江子氏の提案する評価モデル [22] PBL における評価モデルとして，ソフトウェア開発プロジェクトを

- プロダクト (成果物)
- プロセス (実行の仕方)
- グループメンバー (実行する人) のコラボレーション

の 3 つの要素で構成されると捉える評価モデルである．それぞれの項目に設定された達成目標に対して，グループおよび個人の観点から評価を定義し，総合的にプロジェクトの評価を定義している．そのうちプロセスに関する評価モデルの構成要素を表 3 に示す．

プロセス	
定義	プロダクトを作成するための作業
達成目標	機能性[1]・使用性[2]・保守性[3] の高いプロダクトを 作成するための作業を理解，実践できる
評価データ	作業報告のレビュー，作業ログのレビュー，最終レポートの評価
評価基準	理解度・実践による達成度

表 3: プロセスに関する評価モデル

この評価モデルはグループ開発で得ることが望まれる学習についての達成目標を設定している．ただし，評価はレビューによって行われるために定量的な評価には繋がらない，という問題がある．

2.4 sPBL が抱える課題

sPBL の主体となるグループ開発では個人の開発内容やかけた労力が分かりにくくなる傾向がある．そのため，教員には sPBL 実施中から学生の日頃の取り組みに対する十分な把握が求められる．しかし，限られた人数の教員で学生全員の取り組みを確認し続けるのは困難である．これに対して，ITSpiral ではチケット管理システムとバージョン管理システムを用いることによりタスク単位での学生の負荷を評価することを可能とした．ただしこの方法を用いる場合には，学生側に作業報告のための人的コストを課すことになる．加えて，学生が実際の開発時間ではなくタスクの開始時から終了時までの時間を誤って入力してしまうなど，手作業のデータ入力では必ずしも正確な情報が得られない可能性があることが報告されている [12]．学生の負荷を評価することに関しても，現状の sPBL では定量的で具体的な評価基準が設けられることは稀である．そのため，学生個人の開発内容やエフォートを定量的に評価することが出来なかった．結果として，グループが T2 を実現出来ているかの評価も定量的な評価が出来ていなかった．

これまで紹介してきた実例から sPBL が抱える問題を以下の 3 つとして提示する．

[1]ユーザーの明示的および暗示的な必要性を満たす一連の機能を実現しているかを示すメトリクス
[2]ユーザーがソフトウェアを使用する際の労力，使い勝手，仕様結果に関するメトリクス
[3]ソフトウェアに対して明示された改定を行うために必要な労力

P1 評価のためのデータ収集は学生，教員に人的コストをかける

P2 学生個人のグループワークにおける学習内容の評価が困難である

P3 グループ内でタスク分担の割合が適切かどうかの評価が難しい

これらの課題を踏まえて，本論文では学生，教員に人的コストを課すことなく学生の開発内容・エフォートを定量的に評価することを目的とした手法を提案する．

3　オンラインストレージを用いた開発者メトリクスの収集・可視化

本章では sPBL の抱える問題に対して，オンラインストレージと開発者メトリクスを用いた手法を提案する．

3.1　キーアイディア

sPBL における学生評価に関する問題点の解決のために，本論文では以下の 3 つのキーアイディアを採用する．また，以降では学生が sPBL において担当した核開発作業に要した時間や作成した成果物の量をその作業におけるエフォートと呼ぶ..

K1 オンラインストレージの導入による開発ログの自動収集

K2 開発ログを利用した開発作業におけるエフォートの定量化を目的とした開発者メトリクスの計測

K3 K2 で定量化した学生評価メトリクスのグループ内比較による貢献度の定量化

これらのキーアイディアを実現するために，我々はではオンラインストレージを用いた人的コストのかからない学生の開発ログ収集手法および，学生の評価を目的とした開発者メトリクスを導入し，学生毎のエフォート分析，及びグループ内における負荷の偏りを定量化する手法を提案する．

3.2　システムの全体像

図 2 にキーアイディアの実現を行ったシステムの全体像を示す．

以降で，K1 を実現するためのオンラインストレージを用いた開発ログの自動収集手法と，K2，K3 を実現するために設定した学生評価メトリクスに関する説明を行う．

3.3　オンラインストレージ

オンラインストレージとはインターネット上でファイル保管用のディスクスペースを貸し出すサービスである．オンラインストレージサービスには様々な種類のものが存在するが，sPBL での利用を想定し，本論文では下記の条件を満たすものを採用した．

1. ローカルとオンラインストレージ乗でのファイル同期が自動
2. バージョン管理が行われる
3. ローカルの特定フォルダを他者と共有できる

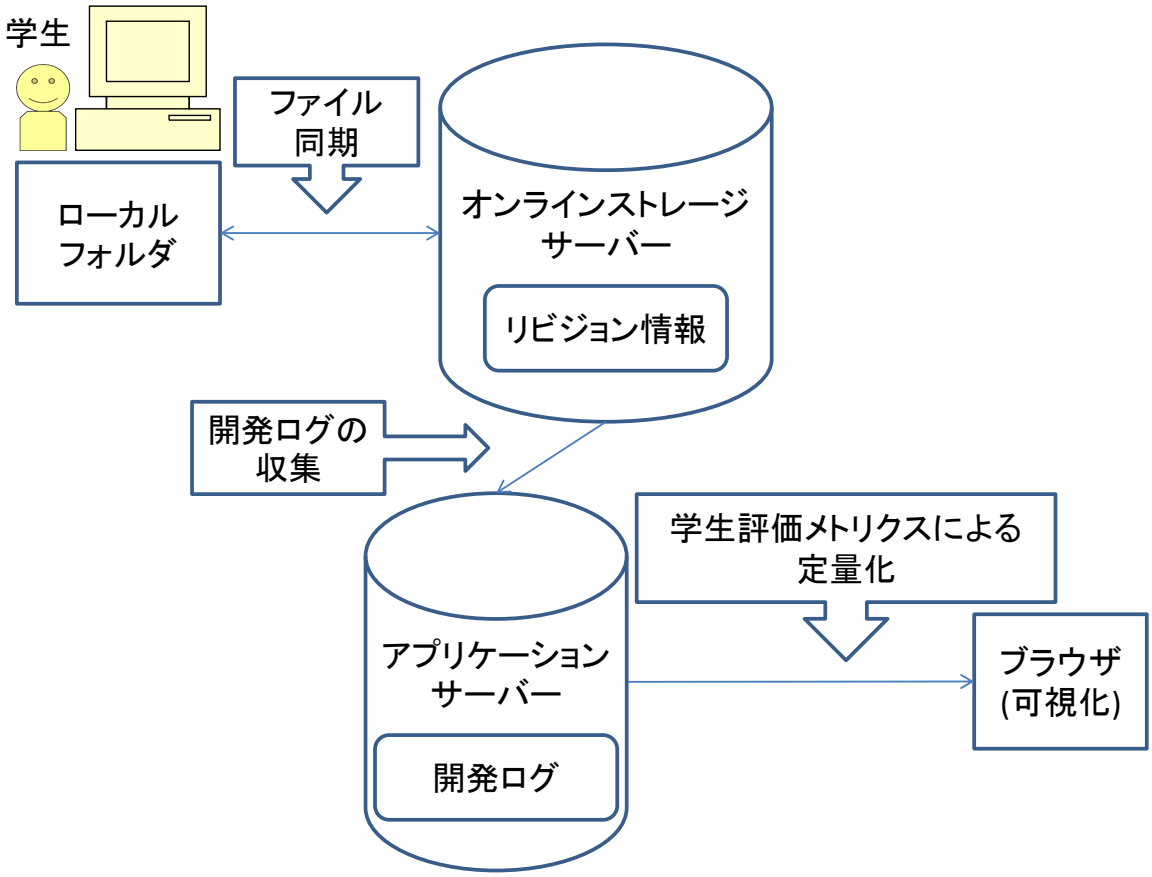
学生
ファイル
同期
ローカル
フォルダ
オンラインストレージ
サーバー
リビジョン情報
開発ログの
収集
学生評価メトリクスによる
定量化
アプリケーション
サーバー
開発ログ
ブラウザ
(可視化)

図 2: システムの全体像

4. オンラインストレージ内のデータにアクセスできる API を持つ

1 から 5 の条件を満たすサービスとして Dropbox と SugarSync の 2 つが存在する．その上で，学生全員にアカウントを用意してもらうことを考えて，より普及している Dropbox を用いた．

Dropbox は，ローカルに作成した Dropbox フォルダ内にファイルを入れるだけでサーバーにアップロードされる機能を持っている．また，作業中の端末がネットに接続されている状態で, かつ作業が Dropbox 上で行われている限りファイルの更新に関しても自動でアップロードされる．また，その更新は常にリビジョンとして Dropbox 側に保存される．

Dropbox に保存されているリビジョン情報は JSON 形式で取得可能である．取得可能な JSON の例を図 3，JSON の各オブジェクトの内容を表 4 に示す．

size	ファイルのサイズ
bytes	バイト単位でのファイルサイズ
is_dir	エントリがディレクトリかどうか
is_deleted	削除されたエントリかどうか
rev	ファイルのリビジョンを表す文字列 更新の検知やバージョン衝突の回避に用いる
hash	フォルダ内の更新を感知するための文字列
tumb_exist	ファイルが画像であり, サムネイルに変換できるかどうかを示す項目
icon	Dropbox 内でファイル形式を表示するための アイコン画像の名前
modified	Dropbox 内でファイルが更新された時刻
root	アプリケーションがアクセスするルートフォルダ
revision	リビジョン番号

表 4: リビジョン情報 (JSON) のオブジェクト内容

ファイルの特定リビジョンにおけるファイル内容は，ファイルの path とリビジョンを表す文字列である rev 用いて Dropbox にリクエストを送ることで取得可能である．さらにファイルを共有することにより，そのファイルのリビジョン情報も共有される．そのため，学生の Dropbox 内の作業用フォルダを教員用 Dropbox アカウントと共有することにより，教員用のアカウント一つで複数の学生の作業内容を取得することが可能になる．

```
[
{
    "is_deleted": true,
    "revision": 4,
    "rev": "40000000d",
    "thumb_exists": false,
    "bytes": 0,
    "modified": "Wed, 20 Jul 2011 22:41:09 +0000",
    "path": "/hi2",
    "is_dir": false,
    "icon": "page_white",
    "root": "app_folder",
    "mime_type": "application/octet-stream",
    "size": "0 bytes"
},
{
    "revision": 1,
    "rev": "10000000d",
    "thumb_exists": false,
    "bytes": 3,
    "modified": "Wed, 20 Jul 2011 22:40:43 +0000",
    "path": "/hi2",
    "is_dir": false,
    "icon": "page_white",
    "root": "app_folder",
    "mime_type": "application/octet-stream",
    "size": "3 bytes"
}
]
```


図 3: リビジョン情報の JSON 例

3.4　学生評価メトリクス

3.4.1　作業量

開発途中のエフォートを，コードに対して行った追加・編集行数基準で定量化するメトリクスとして作業量を定義する．このメトリクスによって，開発途中に行った試行錯誤などを含めた編集作業などのエフォートを取得可能となり，K2 のエフォートの定量化を実現することが可能となる．図 4 に作業量の可視化例を示す．

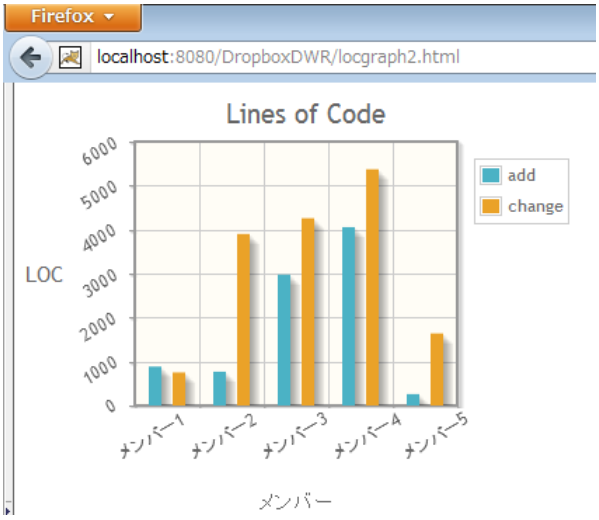

図 4: 作業量の可視化例

まず，ファイル単体における作業量を，学生がファイルに対して記述した行数の総和と定義する．特定のファイルの連続するリビジョン間でファイル内容の差分を diff[23] の定義に従って「追加」,「削除」,「変更」の 3 種に分類する．そのうち，ファイルに 1 行「追加」した例とファイルを 1 行「変更」した例を図 5，6 に示す．

「変更」は，該当箇所を「削除」した後に変更内容を「追加」したと見なす．以下では追加行数を「追加」されたコードの行数，変更 LOC を「変更」において「追加」されたコードの行数と定義する．

ファイル F に関する作業量を取得するに当たり，ファイル F の i 番目のリビジョンを Rev_i と定義する．ファイル F のリビジョンが Rev_1 から Rev_x まで存在していて，Rev_i においてファイル F に対して行った編集の追加行数を $\mathrm{Add}(i-1,i)$，変更行数を $\mathrm{Chg}(i-1,i)$ とすると，ファイル F に対する作業量 $\mathrm{L}_{file}(F)$ を

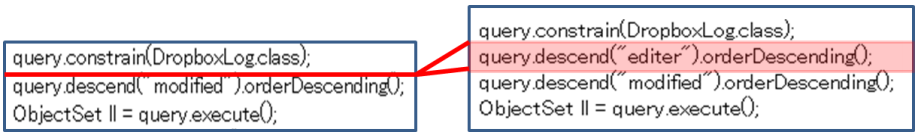

図 5: 「1 行追加」の例

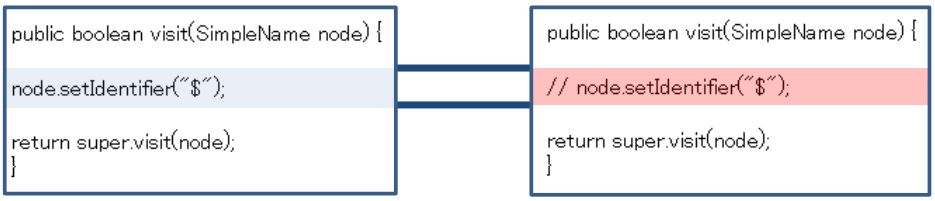

図 6: 「1 行変更」の例

$$\mathrm{L}_{file}(F) = \sum_{i=1}^{x}(\mathrm{Add}(i-1,i) + \mathrm{Chg}(i-1,i)) \tag{1}$$

と定義する．

学生個人の作業量は，学生が作成した各ファイルの作業量の合計と定義される．

3.4.2 進捗度合い

進捗度合いを，ファイルおよびプロジェクトの開発が完了するまでの完成度として定義する．このメトリクスの推移を可視化することにより，sPBL 終了後に学生の開発プロセスを効率や日頃計画的な開発が行われていたかなどの観点から評価することが可能になる．また，同じファイルを担当した学生達の進捗度合いを比較することで，学生にとって難易度の高かったタスクなどの割り出しが可能になる．

図 7 に学生の進捗度合いの可視化例を示す．

このメトリクスでは図 8 のように，作業終了時の成果物と各リビジョンにおける成果物の比較を行い，共通部分の割合を計測し進捗度合いとする．

まず，ファイル単体を対象とした進捗度合いを定義する．ファイル F の完成前のリビジョン Rev_{prog} と完成後のリビジョン Rev_{fin} の diff を取ったとき，$\mathrm{Add}(prog, fin)$ と $\mathrm{Chg}(prog, fin)$ の合計は完成前と完成後の差分コードの行数となる．完成後のファイルの行数から差分コードの行数を引けば，完成前と完成後において一致しているコードの行数が取得できる．完成前と完成後で一致しているコードの行数を E(prog, fin)，完成後のファイルの行数を LOC(fin) とするとファイル F の Rev_{prog} における進捗度合い $\mathrm{P}_{file}(prog)$ は

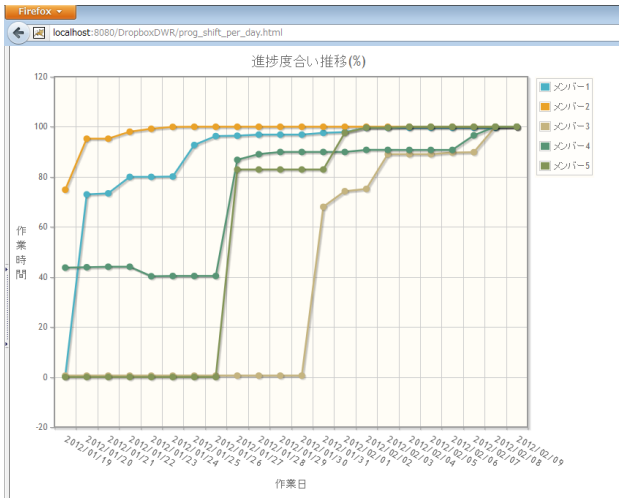

図 7: 進捗度合いの可視化例

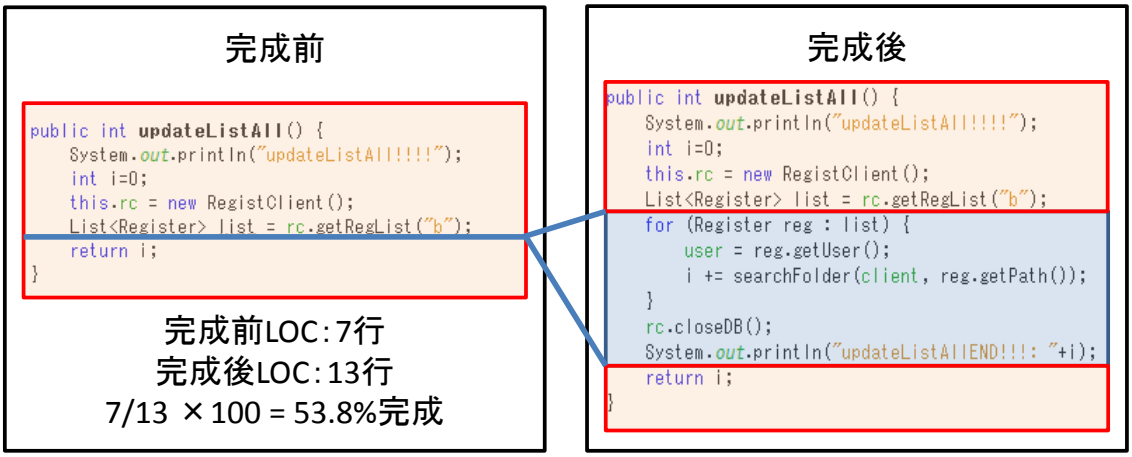

図 8: 進捗度合い

$$\mathrm{P}_{file}(prog) = \frac{\mathrm{E}(prog, fin)}{\mathrm{Loc}(fin)} \times 100\% \tag{2}$$

と定義される．

続いて，メンバー個人を対象とした特定時期における進捗度合いを定義する．メンバー M が担当した全成果物の完成時における総行数を $\mathrm{SUM}_{fin}(M)$ と定義する．また，特定時期におけるメンバー M が担当した成果物それぞれの，完成した成果物との共通したコードの行数の合計を $\mathrm{SUM}_{prog}(M)$ と定義する．ある時期におけるメンバー M の進捗度合い $\mathrm{P}(M)$ は

$$\mathrm{P}(M) = \frac{\mathrm{SUM}_{prog}(M)}{\mathrm{SUM}_{fin}(M)} \times 100\% \tag{3}$$

と定義される．

3.4.3 作業時間

作業時間とは，学生が成果物の開発に要した時間を表すメトリクスである．

図 9 に作業時間の可視化例を示す．

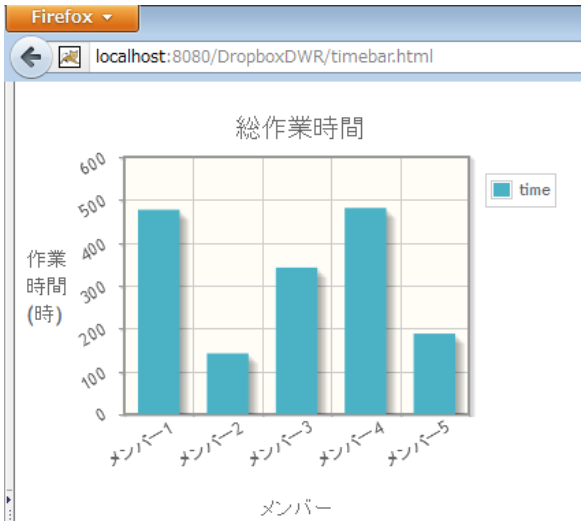

図 9: 作業時間の可視化例

作業時間を定量化することによって，学生がプロジェクトにかけた時間をエフォートとして定量化することが可能になり，作業量と同様に K2 を実現する．加えて，作業時間が少ない学生に対して作業を促すための指標となる．

ある学生が更新した全ファイルのリビジョンを更新時刻順に取得する．隣り合ったリビジョンにおける経過時間が一定時間より短ければ，その 2 つのリビジョン間は作業中であると判断する．作業中であると判断したリビジョン間の経過時間の合計をその学生の作業時間と定義する．学生 M が更新した全ファイルのリビジョンを更新時刻順に並べたもののうち i 番目のリビジョンを RevM_i とする．ファイルのリビジョンが RevM_1 から RevM_z までであり，RevM_{i-1} と RevM_i の経過時間を t(i) とする．作業時間外と判断する経過時間の閾値を TO とすると，作業時間は Algorithm1 で取得した sum となる．

Algorithm 1 作業時間計測アルゴリズム

```
sum ⇐ 0
for i = 1 to z do
  if t(i) < TO then
    sum ⇐ sum + t(i)
  end if
end for
```

3.4.4 貢献度

作業量，作業時間それぞれについてグループにおける各メンバーにかかった負荷の割合を貢献度として定義する．貢献度を定量化することによって，グループにおけるメンバー間の開発内容やエフォートの偏りを定量化することが可能になり，グループが T2 を実現できているかの評価を行うことができるようになる．また，貢献度の低い学生に対して作業を促すなどの指導が可能になる．

図 10 に作業時間に関する貢献度の可視化例を示す．作業量に関する貢献度も同様の可視化が可能である．

グループ全体での作業量を L(G) とすると，メンバー M の作業量に関する貢献度 $\mathrm{C_{LOC}}$ は

$$\mathrm{C}_{LOC} = \mathrm{L}(M)/\mathrm{L}(G) \tag{4}$$

と定義される．作業時間も同様に，メンバー M の作業時間を T(M)，グループ全体での作業時間を T(G) とおくと，メンバー M の作業時間に関する貢献度 C_{TIME} は

$$\mathrm{C}_{TIME} = \mathrm{T}(M)/\mathrm{T}(G) \tag{5}$$

と定義される．

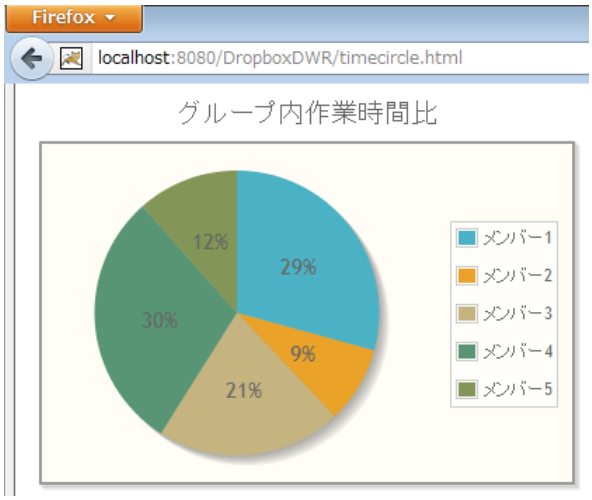

図 10: 貢献度の可視化例 (作業時間)

4　実装

本章では開発したアプリケーションの実装について述べる.

4.1　概要

学生の開発内容とエフォートに関する情報を収集，保存，定量化，可視化することを目的として，オンラインストレージを組み合わせた Web アプリケーションを実装した.

4.2　開発環境

開発言語として Java SE Development Kit1.6，JavaScript を用い，Eclipse INDIGO および NetBeans IDE 7.0.1 上で開発を行った．また，オンラインストレージとして Dropbox を利用した.

4.2.1　ライブラリ

本アプリケーションを開発するためにライブラリとして以下の 4 つを用いた.

Dropbox4j　Dropbox に対する OAuth 認証およびサーバーへのアクセス，Dropbox からの送られるデータの処理に必要なメソッドを提供する API[24]

db4o　オブジェクトデータベースを提供するライブラリ [25]

DWR　Java と JavaScript を連動するための Ajax アプリケーション開発用ライブラリ [26]

jqplot　JavaScript 上で動作するグラフ描画プラグイン [27]

4.3　アプリケーションの構成

以下で，Java を用いた処理部分のクラス図と本アプリケーションのユーザーインタフェースを構成する html の構成を示す.

4.3.1　処理部分

図 11 でアプリケーションの処理部分の構成をクラス図で示す.

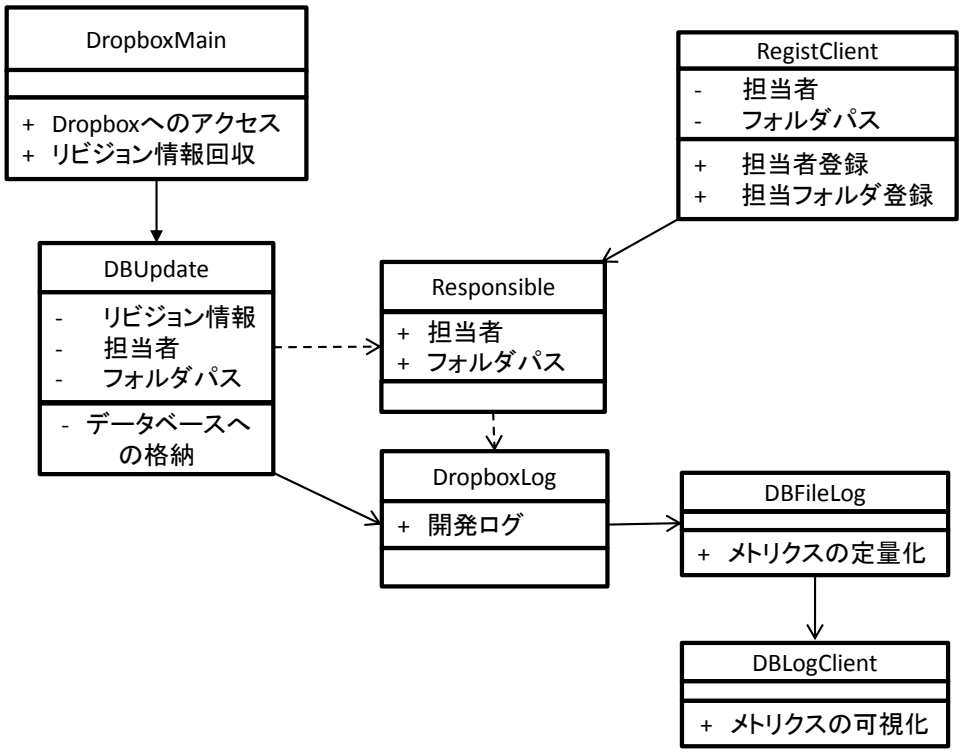

図 11: クラス図

本アプリケーションの処理は，以下のクラスを軸に行われる.

DropboxMain　Dropbox への OAuth 認証，アクセス，リビジョン情報の回収を行うクラス

DBUpdate　リビジョン情報とフォルダパス、フォルダ担当者の情報を開発ログとして DropboxLog クラスへ格納するクラス

RegistClient　フォルダパスと担当者を登録し，Responsible クラスとして格納するクラス

Responsible　フォルダパスと担当者の情報を記録するクラス

DropboxLog　開発ログを記録するクラス

DBFileLog　開発ログを学生評価メトリクスに従って定量化するクラス

DBLogClient　定量化した情報を可視化するためのデータ形式に変換するクラス index でオンラインストレージへの認証と接続を行う．接続が終了したのちに main ページへ移動し，フォルダ担当者の登録，可視化グラフの表示が可能になる．またオンラインストレージからのリビジョン情報収集は，main ページに設置したボタンから行う．

4.4　html 構成

図 12 にユーザーインタフェースなる html 構成を示す．

4.5　処理の流れ

オンラインストレージの認証

オンラインストレージへの認証を行う流れを図 13 に示す．index ページにおいてオンラインストレージへの認証と接続を行う．接続が完了した後に main ページへ移動する．

フォルダ担当者の登録

情報の保存における処理の流れを図 14 に示す．教員は学生と共有したフォルダのフォルダパスと，そのフォルダの元の持ち主である学生 (担当者) の名前を入力し，データベースに登録する．

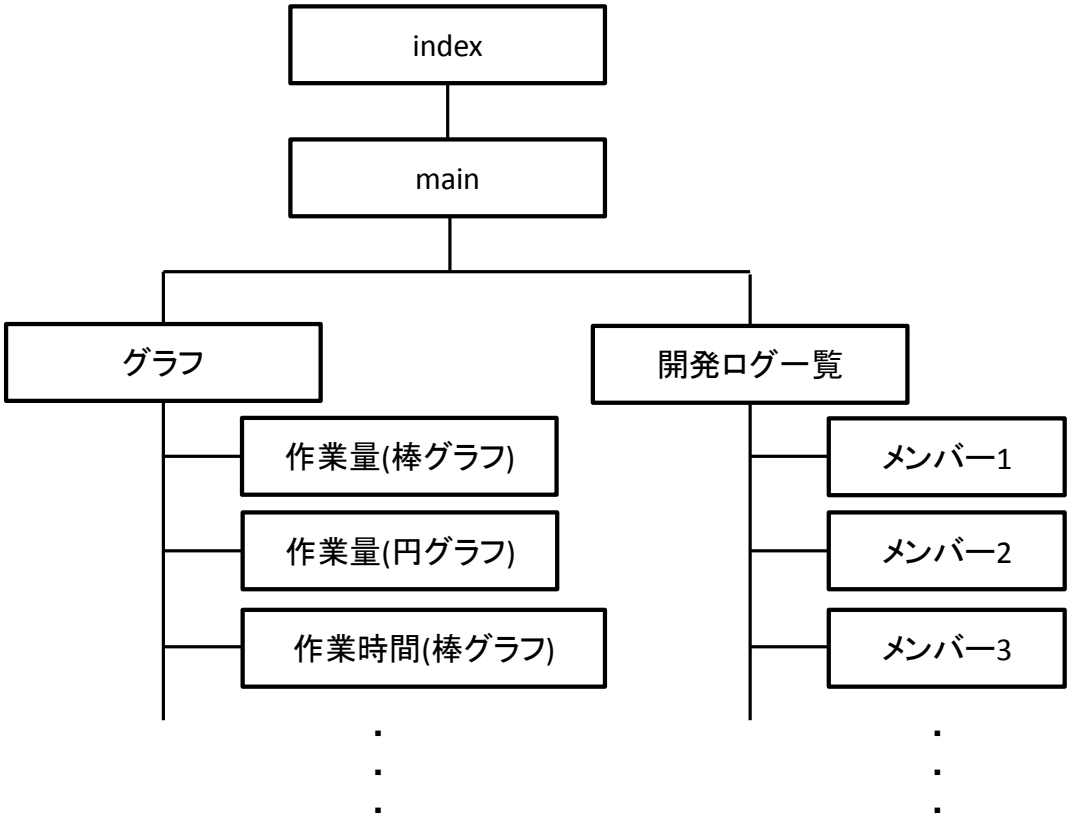

図 12: html 構成

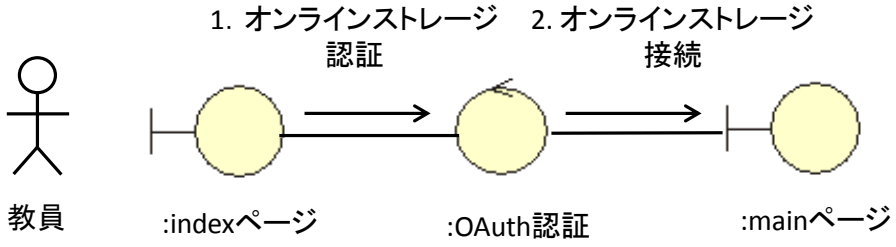

図 13: アプリケーション開始の流れ

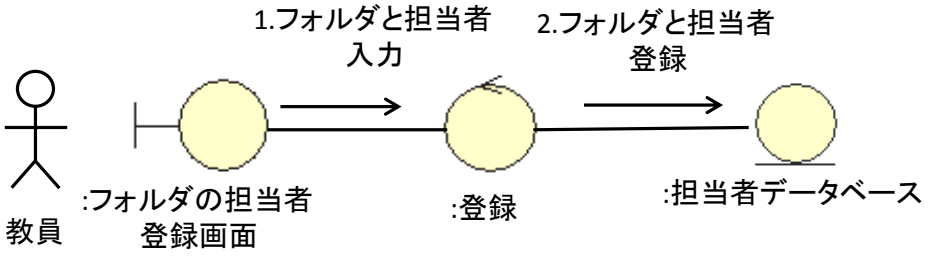

図 14: フォルダ担当者登録の流れ

オンラインストレージからのデータ収集

情報の保存における処理の流れを図 15 に示す．アプリケーションに開発ログの収集指示を出すと，アプリケーションは担当者データベースからフォルダの担当者と担当するフォルダのパスを取得する．取得したパスに従ってオンラインストレージからリビジョン情報を取得し，フォルダ担当者の名前と合わせて開発ログとして開発ログデータベースへ格納する．

メトリクスの可視化

情報の保存における処理の流れを図 16 に示す．アプリケーションがグラフ表示の指示を受けると，開発ログデータベースから必要な開発ログを受け取り，学生評価メトリクスに従って定量化を行う．定量化したデータを可視化するために変換を行い，ブラウザ上でグラフとして表示する．

4.6 利用方法

この手法を用いる際の利用例を学生側，教員側に分けて解説する．

4.6.1 準備

sPBL 開始のために，学生と教員共にいくつかの簡単な環境設定を行ってもらう．

学生側

1. 自分用のオンラインストレージアカウントを用意する
2. オンラインストレージ上に共有用フォルダを用意し，その中に開発用ワークスペースを配置する

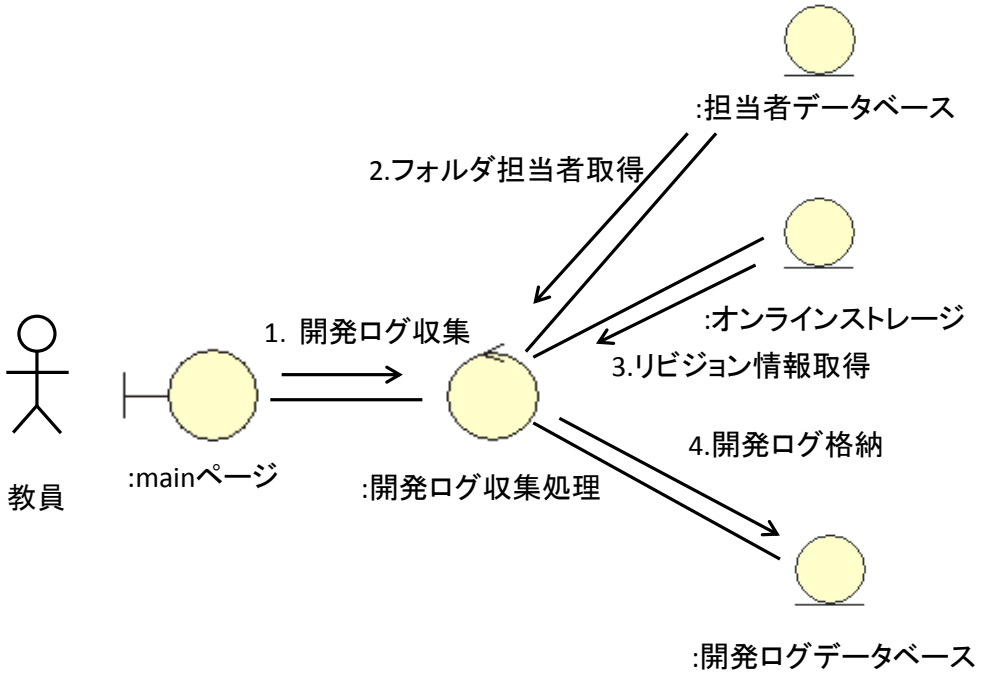

図 15: オンライストレージからのデータ収集の流れ

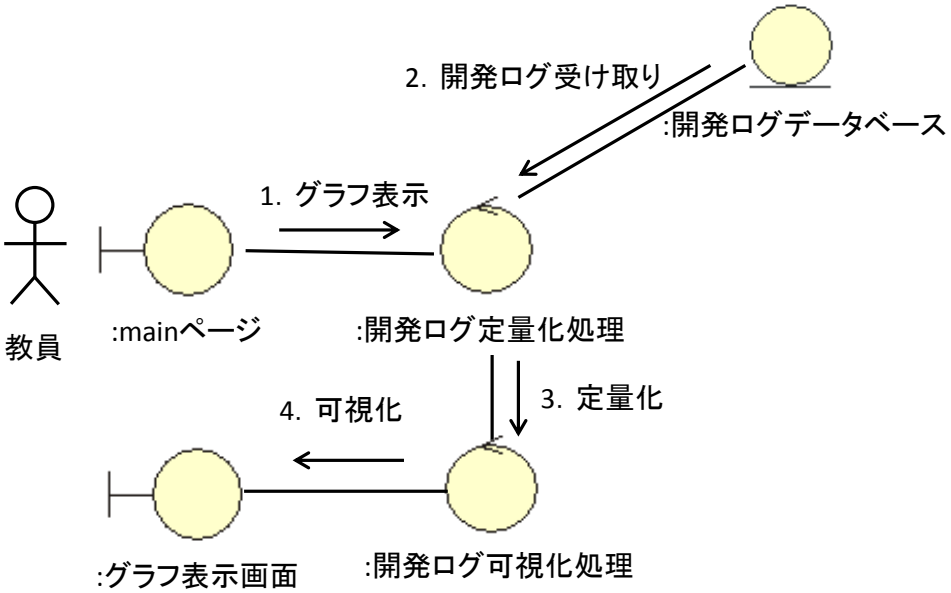

図 16: オンライストレージからのデータ収集の流れ

3. フォルダの共有申請を教員用のオンラインストレージアカウントに送る

教員側

1. 教員用 Dropbox アカウントを用意する
2. 教員用 Dropbox アカウントに本アプリケーションを登録する
3. 学生の Dropbox 上にある共有用フォルダとの同期を完了する
4. 共有したフォルダの持ち主である学生の名前と共有したフォルダの教員用 Dropbox フォルダ上でのパスを担当者データベースに登録する

4.6.2 開発中

開発ログの収集に際し，学生に少し制約をかける．

学生側 開発ログの収集のために，学生には PC をネットに接続した状態で，Dropbox 上で開発を進めてもらう．

教員側 随時，Dropbox から開発ログを収集し Web ブラウザ上で学生の開発状況を監視する．

4.6.3 開発終了後

収集した各学生の開発者メトリクスから学生個人の開発内容とエフォートの評価，グループメンバー間のメトリクス比較からグループにおける貢献度の偏りに関する評価を行う．

以下に可視化したグラフの活用例を示す．

作業量，作業時間の棒グラフ (図 4(P.17) ，図 9(P.20)) 学生の作業量や作業時間をプロジェクト全体およびタスク毎，開発言語毎などの分類を行った上で確認することが可能である．これらの情報は学生の評価基準となるだけではなく，学生の開発者としての経験の記録とすることができる．

貢献度の円グラフ (図 10 (P.22)，図 17) グループにおけるタスクの偏りを監視する．偏りが発生している場合，貢献度の低い学生にタスクを割り振るよう促す．また，グループの評価としてタスクの分担を適切に行うことができたかどうかの評価基準となる．図 17 においては，メンバー 1 とメンバー 5 の作業量が特に少ないことが分かる．そこでこの 2 人に対して作業に関する聞き取り等を行う等の指導が可能となる．

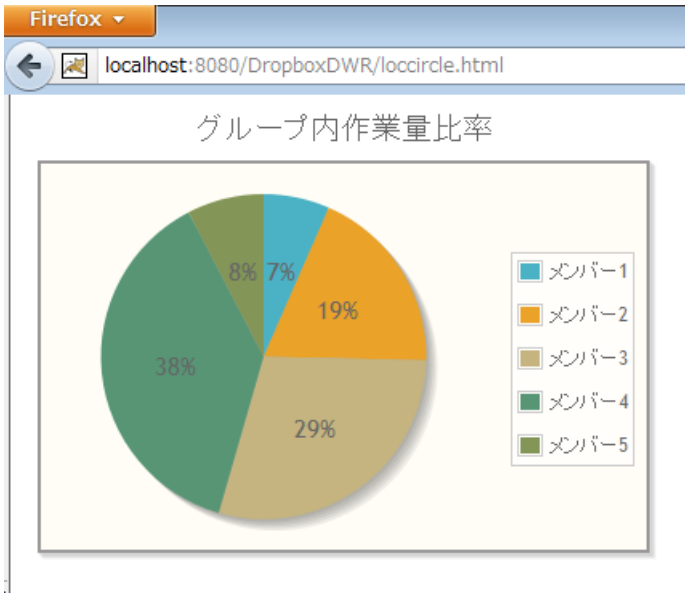

図 17: 貢献度の可視化例 (作業量)

作業量，作業時間推移の折れ線グラフ (図 18)　作業量，作業時間の増加傾向を可視化し，sPBL の授業時間外を含めたプロジェクトへの注力度に関する評価を行う．また，長期的に数値が増加していない学生を割り出す．作業時間が停滞している場合，作業を行っていないと判断することができる．また，作業時間は増加しているが作業量の増加率が下がった場合に，開発に手間取っている可能性があると考えられる．どちらの場合も，直接その学生に現状を尋ねる等の指導が可能となる．図 18 においては，メンバー 3 の作業量が 0 行から推移していないことが見て取れる．そこで教員が作業を促すことで，メンバー 3 に作業を進行させることに成功した，ということがグラフから判断できる．

進捗度合の折れ線グラフ (図 7(P.19))　学生の開発速度に関する評価の基準となる．また PBL 全体において，各ファイルの進捗度合いを分析することで学生が躓きやすい問題をあぶり出し次回以降の sPBL における指導方針を決定するための指針の一つとすることが可能となる．図 7 においては，メンバー 4 の進捗が停滞し，一時期に関しては進捗が後退している．このことからメンバー 4 の開発において何かしらの問題が発生していると判断でき，教員による聞き取りや指導を行うことが可能となる．その結果，問題を解決して作業が進行したということが確認できる．

5　評価と考察

5.1　評価の概要

本論文で設定した学生評価メトリクスのうち，コードの行数が基準となるメトリクスに関しては誤差なく取得可能である．しかし，作業時間のメトリクスに関してはいくつかの要因によって誤差が発生することが考えられる．そのため本研究では，本アプリケーションが定量化するメトリクスのうち作業時間についての誤差と妥当性を確認するための評価実験を行った．

5.2　作業時間誤差の発生要因

作業時間誤差は以下の 2 種類の要因で発生する．

E1 リビジョン間の経過時間を対象とした閾値の設定によって作業時間が変わる

E2 作業開始時刻から最初に成果物が保存された時刻までの経過時間が分からない

E1 は作業時間計測の際に設定される閾値に起因する誤差である．3.4.3 節で述べた通り，リビジョン間の経過時間を作業時間として計測するかどうかは，計測時に設定される閾値の大きさに依存する．そのため，閾値が過剰に大きければ作業していない時間が計測されてしまい，逆に閾値が小さすぎれば，作業をしていたにもかかわらず作業時間として計測されない．

E2 はオンラインストレージを用いて作業時間計測を行うために発生する誤差である．ある成果物が存在する時，開発者がその成果物を対象として開発を開始してから，最初にその成果が保存されるまでの時間はオンラインストレージ側では分からない．

本論文では，これらの誤差のうち E1 に相当する閾値の適切な値を評価するための実験を行った．

5.3　評価方法

作業時間についての誤差と妥当性を確認するため，学生 6 人を対象として作業を行った時間に関するデータを収集した．作業を行った時間に関して，以下のデータを同時に計測した．

- 実際に作業を行った時間 (申告値)
- 本アプリケーションが取得した作業時間 (測定値)

5.3.1 申告値

実際に作業を行った時間を「WorkTimer ／作業時間計測ツール」[28] を利用して極力学生に人的コストをかけないよう計測を行った．WorkTimer は随時起動しておき，作業開始と終了時にボタンを押すだけでその時刻と経過時間を取得できるツールである．

図 19 に WorkTimer を利用中の画面を示す．左端のウィンドウが WorkTimer である．

5.3.2 測定値

3.3 節で提示した本アプリケーションの手法を用いて開発ログの収集を行った．その上で，申告された時刻に行われた作業に対応する開発ログから作業時間を計算し，測定値として用いた..

またこの実験では，作業を行った環境や作業内容が計測にどのような影響を及ぼすかを調査するため，作業を以下の二つに分類した．

- コーディング作業
- 論文執筆作業

申告値と測定値のペアを，コーディング作業に関して 21 個，論文執筆作業に関して 32 個用意した．

5.4 誤差と妥当性の評価

用意したペアに対して，閾値を変化させた上で申告値と測定値の間に発生する誤差を計算し，妥当性があるかの評価を行った．

計測したデータは絶対値にばらつきが存在するため，誤差を相対誤差で表す．申告値を T，閾値が x 分のときの測定値を S(x) と置いたとき，閾値 x 分のときの相対誤差 E(x) を

$$\mathrm{E(x)} = (1 - \frac{\mathrm{S(x)}}{\mathrm{T}}) \times 100\% \tag{6}$$

とする．x を変化させたとき相対誤差 E(x) がコーディング作業，論文執筆作業のそれぞれに関してどのように推移したかを図 20 に示す．

図 20 で示すように，閾値を増加させる毎に誤差が減少している．コーディングに関しては閾値を 100 分以上にすると相対誤差の変化がなくなる．一方，論文執筆では閾値が 140 分以上で相対誤差は変化しなくなる．それぞれ相対誤差が変化しなくなった閾値において測定値が妥当性を持つかを評価する．

図 18: 作業量推移の可視化例

図 19: WorkTimer 利用中の画面

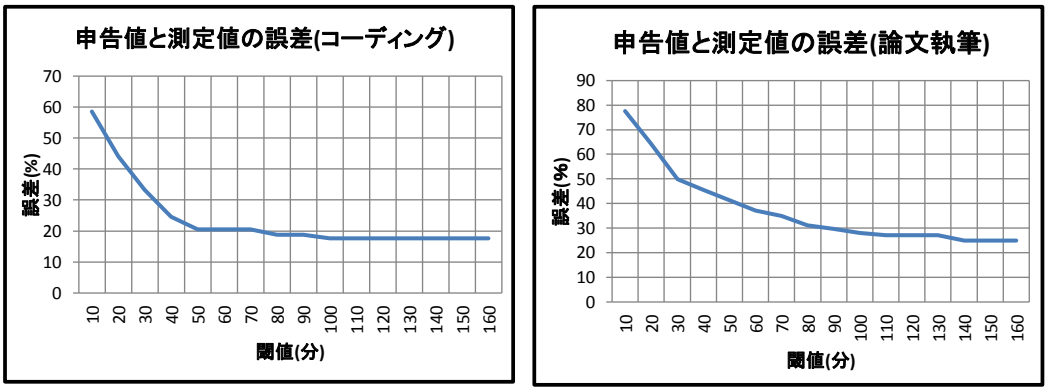

図 20: 作業時間の誤差推移

妥当性の評価基準として，申告値と測定値の散布図から二つの数値が相関性を持つかどうかを調べる．相関係数は以下の式で求められる．

$$R = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i)^2 \sum_{i=1}^{n}(y_i - \overline{y})}} \tag{7}$$

図 21 で申告値と測定値の散布図と回帰直線，相関係数の 2 乗である寄与率をコーディング作業，論文執筆作業のそれぞれについて示す．

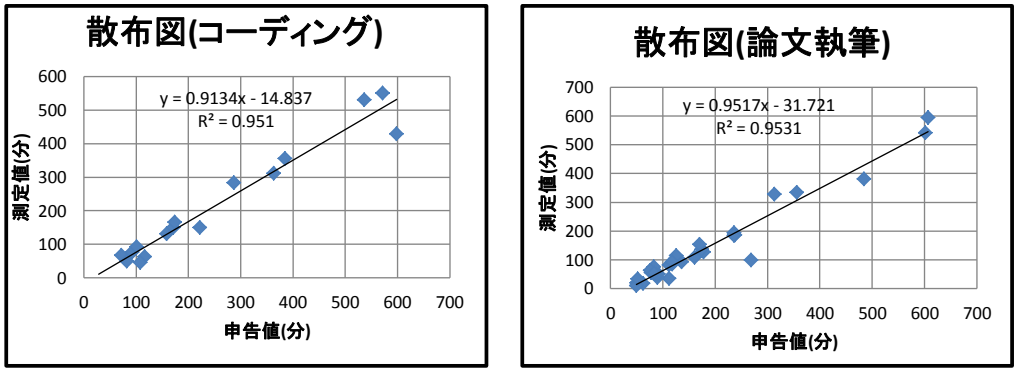

図 21: 申告値と測定値の散布図

相関係数の絶対値と相関の程度の表現の対応関係は，表 5 のように考えれば良いと言われている．

コーディング作業の相関係数は R=0.975，論文執筆作業の相関係数は R=0.976 であり，表

$1.0 \geq \|R\| \geq 0.7$	高い相関がある
$0.7 \geq \|R\| \geq 0.5$	かなり高い相関がある
$0.5 \geq \|R\| \geq 0.4$	中程度の相関がある
$0.4 \geq \|R\| \geq 0.3$	ある程度の相関がある
$0.3 \geq \|R\| \geq 0.2$	弱い相関がある
$0.2 \geq \|R\| \geq 0.0$	ほとんど相関がない
出典：「社会調査の基礎」放送大学テキスト	

表 5: 相関係数の絶対値と相関の程度の表現の対応関係

5 に照らし合わせると，どちらも高い相関を持つといえる．

5.5 実験結果についての考察

申告値と測定値の相関関係を調べた結果，閾値の設定によって高い相関が得られた．このことで，本手法を用いた作業時間には妥当性があることが証明されたといえる．ただしこの実験では，5.1 節で説明したように閾値を大きくし過ぎると逆に作業時間外であるリビジョン間を作業中と判断してしまうという問題を考慮に入れていない．そのため，今後は 1 日全体において作業をしている時間と作業をしていない時間に関する詳しい情報を収集し，適切な閾値の設定と設定後の誤差に関して評価を行う必要がある．

コーディング作業と論文執筆作業において，誤差が収束する閾値に差が生じた．これは，コーディング作業が論文執筆作業に比べてファイルを保存する頻度が高く，リビジョン間の経過時間が相対的に短いことに由来すると考えられる．ファイルの保存頻度の高さを生んだ原因として，コーディングにおいては論文執筆などの文章を書く作業と違い，プログラムを実行する際にソースコードを全て保存する必要があるということが考えられる．また，ファイルの保存頻度は人によってある程度の差が存在する可能性がある．これらの保存頻度や保存間隔における作業環境や作業内容による差異，属人性の排除のためには，定期的なファイルのオートセーブ機能を備えた作業環境を用いることが有効な可能性がある．

6　あとがき

本研究では，ソフトウェア開発 PBL における学生毎の成果物の量，成果物開発に要した時間といったエフォートの定量化・可視化を行うことを目標に，学生評価メトリクスを提案した．

オンラインストレージを用いたメトリクス計測手法により，個別の学生の開発内容を学生や教員のコストをかけずに計測することが可能となった．

評価実験では，学生評価メトリクスのうち，誤差の発生しやすい作業時間メトリクス (学生が成果物の開発に要した時間) について，実際の作業時間と提案システムによって計測される作業時間の比較を行った．結果として，学生の利用する開発環境によって適切な閾値が変動することや，概ね 20〜30%の範囲で実際の作業時間よりも少ない値を提案システムが示すことが確認できた．

今後はオートセーブ機能のついた開発環境の利用や，作業量からの作業時間推定といった手法により，より精度の高い作業時間メトリクスの計測を行っていきたい．また，実際のソフトウェア開発 PBL に本手法を適用することで，メトリクスの有用性を確認したい．

謝辞

日頃から御教授下さり, 本研究ならびに本報告書の作成に当たって様々な御指摘と御指導を賜った大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻 楠本真二教授に深く感謝の意を表します.

本研究ならびに本報告書の作成に当たり, 細部に至るまで貴重な御指摘と熱心な御指導を賜った大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻 井垣 宏特任准教授に深く感謝の意を表します.

本報告書の作成に当たり適切な御指摘とご助言を頂きました大阪大学 大学院情報科学研究科コンピュータサイエンス専攻 岡野浩三准教授に深く感謝の意を表します.

本研究に多大なるご助言ご指導を頂きました大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻 肥後芳樹助教に深く感謝の意を表します.

本報告書の作成に当たり，的確かつご丁寧なご指導を頂きました大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻 博士前期課程 1 年の長瀬 義大 氏に厚くお礼申し上げます.

本報告書の作成に当たり，実験にご協力いただき大変重要なデータを提供してくださった大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻 楠本研究室の 5 名の学生の皆様に厚くお礼申し上げます.

最後に, お世話になった大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻 楠本研究室の皆様に厚くお礼申し上げます.

参考文献

[1] 情報サービス産業協会. 情報サービス産業白書. 日経 BPP 社, 2009.

[2] 『先導的 IT スペシャリスト育成推進プログラム』の概要について, 2007. http://www.ipa.go.jp/jinzai/sangaku/pdf/shiryo4.pdf.

[3] 経済産業省「高度 IT 人材の育成を目指して (報告書案) に関する意見」, 2007. http://www.ipsj.or.jp/03somu/teigen/meti070521.html.

[4] E. d. Graaff and A. Kolmos. *Management of Change Implementation of Problem-Based and Project-Based Learning in Engineering*. Sense Publishers, 2006.

[5] 沢田篤史, 小林隆志, 金子信幸, 中道上, 大久保弘崇, 山本晋一郎: “飛行機制御を題材としたプロジェクト型ソフトウェア開発実習”, 情報処理学会論文誌, Vol. 50, No. 11, pp. 2677–2689, 2009.

[6] H. Batatia: “A model for an innovative project-based learning management system for engineering education,” *Computer aided learning in Engineering Education(CALIE2001)*, 2001.

[7] “ThinkQuest,”. http://www.thinkquest.org/en/.

[8] C. Hmelo-Silver: “Problem-Based Learning: What and How Do Students Learn?,” *Educational Psychology Review*, Vol. 16, pp. 235–266, 2004. 10.1023/B:EDPR.0000034022.16470.f3.

[9] 湯浅且敏, 大島純, 大島律子: “PBL デザインの特徴とその効果の検討”, 静岡大学情報学研究, Vol. 16, pp. 15–22, 2010. http://ci.nii.ac.jp/naid/110008144617/.

[10] 上田勇仁: “Project Based Learning 先行研究調査報告”, e ラーニング授業設計支援室ランチョンセミナー, Vol. 13, , 2011. http://cvs.ield.kumamoto-u.ac.jp/wpk/wp-content/uploads/2011/07/luncheon_ppt2011no99hayatoueda.pdf.

[11] 福安直樹, 佐伯幸朗, 水谷泰治: “プロジェクトの可視化に基づく PBL 指導支援”, ソフトウェア工学の基礎 XVII 日本ソフトウェア科学会 FOSE 2010, 2010.

[12] 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 (教育実践研究論文,< 特集 > 学習・教育支援のための技術開発)”, 日本教育工学会論文誌, Vol. 34, No. 3, pp. 289–298, 2010-12-01.

[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”, 第 73 回全国大会講演論文集, Vol. 2011, No. 1, pp. 539–541, 2011.

[14] “IT Spiral,”. http://it-spiral.ist.osaka-u.ac.jp/index.html.

[15] 古宮誠一: “PBL に基づくオブジェクト指向ソフトウェア開発技術の演習授業”, *http://www.grace-center.jp/downloads/itsp/talk7.pdf*, 2011.

[16] 橋浦弘明, 大蒔和仁, 土田賢省: “東洋大学総合情報学部の実践的ソフトウェア開発 PBL への取り組み”, *IPSJ/SIGSE Software Engineering Symposium 2011*, 2011.

[17] M. Mason, でびあんぐる監訳. Subversion 実践入門:達人プログラマに学ぶバージョン管理 (第 2 版). オーム社, 2007.

[18] まちゅ: “チケット駆動開発…ITpro Challenge のライトニングトーク (4)”,. http://www.machu.jp/diary/20070907.html(参照日:2012/02/02).

[19] PBL 教材洗練ワーキンググループ: “PBL(ProjectBasedLearning) 型授業実施におけるノウハウ集”,, 2011. http://grace-center.jp/downloads/itsp/pblknowhow20110726.pdf.

[20] M. Pomeroy-Huff, R. Cannon, T. A. Chick, and J. L. M. W. Nichols. “The Personal Software $Process^{SM}$(PSP^{SM})Body of Knowledge, Version 2.0,”, 2010.

[21] 吉村巧朗, 亀井靖高, 上野秀剛, 門田暁人, 松本健一. “ブレークポイント使用履歴に基づくデバッグ行動の分析”,. 電子情報通信学会技術報告, 第 109 巻, pp. 85–90, 2009.

[22] 松浦佐江子: “実践的ソフトウェア開発実習によるソフトウェア工学教育 (分析・設計技法,¡特集¿ソフトウェア工学の理論と実践)”, 情報処理学会論文誌, Vol. 48, No. 8, pp. 2578–2595, 2007-08-15.

[23] J. W. Hunt and M. D. McIlroy: “An Algorithm for Differential File Comparison,” *Computing Science Technical Report, Bell Laboratories*, 1976.

[24] “Dropbox4j,”. https://github.com/Frostman/dropbox4j.

[25] “db4o,”. http://www.db4o.com/japan/.

[26] “DWR - Easy Ajax for JAVA,”. http://directwebremoting.org/dwr/index.html.

[27] “jqPlot,”. http://www.jqplot.com/.

[28] “WorkTimer/作業時間計測ツール”,. http://www.bzwind.com/~ntak/wtimer.html.