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.

細粒度プロジェクトモニタリングのための DaaSを利用したソフトウェア開発PBL支援環境の提案

眞鍋　雄貴† 井垣　宏† 福安　直樹††
佐伯　幸郎††† 楠本　真二† 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 番 5 号
†† 和歌山大学システム工学部
††† 高知工科大学情報学群
E-mail: †y-manabe@ist.osaka-u.ac.jp

あらまし　受講生が主体となって PBL 形式で行われるソフトウェア開発演習では，プロジェクトの進捗に応じた振り返りが重要である．振り返りを行うためにはプロジェクトで行われた受講生ごとのタスクを正確に記録し，問題の抽出や今後の方針の提案を行う必要がある．しかしながら，受講生一人ひとりのタスク記録がどこまで正確かを講師が評価することは困難である．本稿では，DaaS 環境を用いた開発活動の正確なログデータを取得可能な開発環境の構築を行い，どの様なデータを取得できるのか検証を行う．
キーワード　ソフトウェア開発 PBL，振返り，タスク管理，DaaS

Toward software development PBL supporting environment for fine-grained project monitoring on DaaS

Yuki MANABE†, Hiroshi IGAKI†, Naoki FUKUYASU††,
Sachio SAIKI†††, Shinji KUSUMOTO†, and Katsuro INOUE†

† Department of Computer Science Graduate School of Information Science and Technology, Osaka University 1–5 Yamadaoka, Suita, Osaka 565–0871 Japan
†† Faculty of Systems Engineering, Wakayama University
††† School of Information, Kochi University of Technology
E-mail: †y-manabe@ist.osaka-u.ac.jp

Abstract　Retrospectives play an important role in a software development exercise of the project-based learning (PBL) style. To conduct retrospectives, students have to collect what they did in a project, extract problems based on their task records and make a new suggestion for project improvement. However, it is difficult for teachers to evaluate accuracy of their task records collected by students. In this paper, we propose a software development environment to obtain accurate records in development activities on DaaS (Desktop as a Service). In a case study, we show the examples of task records collected by our environment.
Key words　Software Development PBL, Retrospective, Project Monotoring, DaaS

1. まえがき

社会インフラとしての IT の必要性や重要性が高まるにつれて，円滑な業務進捗のためのプロジェクトマネジメント力といった高度な技能を備えたソフトウェア技術者の育成が求められている．そのような高度なソフトウェア技術者の育成を目的として，プロジェクト形式でのソフトウェア開発演習 (ソフトウェア開発 PBL，以降 SDPBL と呼ぶ) が多くの情報系教育機関で行われている [1].

SDPBL では，複数の受講生がグループを組み，ソフトウェア開発における要求分析から運用に至るまでの工程 (の一部分) を実際に体験するという形式で行われることが多い [2]．ただ

し，SDPBL の内容については標準化されたものが存在しないため，教育機関ごとにその内容が大きく異なっている．

一方でどのような SDPBL においても必ず行われる活動として，振り返りがある．振り返りとは，プロジェクトにおける受講生らのタスクとその進め方を評価し，改善するためのプロジェクトグループ全体によって行われる活動である [3].

振り返りでは，行われたプロジェクトからの問題の抽出や改善のための方策の提案，実施された方策に対する評価を具体的な指標や根拠に基づいて行うことが重要であるとされている [4]．そのため，SDPBL ではプロジェクト中に行われた受講生一人ひとりのタスクを正確に記録する方法とその記録に基づいた具体的な振り返り実施手法を講師が指導する必要がある

我々の研究グループでは，アジャイルソフトウェア開発手法の一つであるチケット駆動開発 [5] を用いて，SDPBL を行ってきた [6]．この開発手法では，受講生一人ひとりのタスクがチケットという単位でチケット管理システムによって記録されるため，振り返りの判断基準となるタスク記録を容易に収集することができる．一方で，タスクの開始・終了時刻やタスクに要した時間といった記録は受講生の負荷が大きいため，誤って記録されることも多い．実際に我々が行った SDPBL を対象として，受講生に記録されたチケットに含まれる誤りの数を分析したところ，平均して 40%程度のチケットにおいて，タスク開始 (着手) 時刻や終了時刻，タスク粒度といったタスクを構成する要素のいずれかが誤って記録されていた．

そこで我々は受講生らによって記録されたタスク情報が正しいかどうかを講師が分析する際の支援を目的として，DaaS(Desktop as a Service) を用いた開発環境における細粒度プロジェクトモニタリング手法を提案する．DaaS とはデスクトップ環境を仮想化し，クラウドサービスとしてインターネット上で提供する枠組みを指す．ソフトウェア開発環境を DaaS 上に構築することで，開発者はどこからでも常に同じ環境でソフトウェア開発に携わることが可能となる．我々はこのような DaaS 上のソフトウェア開発環境を利用し，その開発環境で受講生が行っているアプリケーション操作を様々なログによって記録することで，細粒度のプロジェクトモニタリングを実現する．DaaS 上でのソフトウェア開発環境では，ログイン・ログアウト時刻やアプリケーションの起動・終了時刻，ファイルに行われた修正内容といった受講生の振る舞いが記録される．これらのログを収集し，受講生によって記録されたタスク情報と比較することで，どのタスク記録にどのような誤りがあるかを講師が検知することが容易となると考えている．

以降では，タスク記録において発生する誤りに関する予備実験と，提案するプロジェクトモニタリング環境のプロトタイプによるケーススタディについて詳述する．

2. 準　　備

2.1 ソフトウェア開発 PBL

SDPBL とはソフトウェア開発プロジェクトをテーマとし，実践的なソフトウェア開発プロセスやプロジェクトマネジメントを受講生に体験させることを目的とした PBL 形式の演習である．PBL 形式による教育は，通常の座学による講義・演習だけでは習得の難しいプロジェクトの運用に必要なスキルを，主体的なプロジェクト運営を通じて受講生に獲得させることを目指して実施される [7].

大学における SDPBL の代表的な例として，松澤らは，顧客を地域の商店や大学教授とし，PM を目指す企業の技術者と学生が協同することで顧客が望むソフトウェアを作成する SDPBL を報告している [2]．沢田らは，組み込みソフトウェア開発プロジェクトをテーマとし，分析から受け入れテストまでを行って飛行船制御ソフトウェアを作成する SDPBL を報告している [8]．これらの SDPBL の多くでは，納期の順守や成果物の高い品質，必要工数といった制約を受講生らのグループに課すことで，プロジェクトマネジメントの重要性を受講生に認識させることに成功している．

我々の研究グループでは開発実践演習という SDPBL をこれまで行ってきた．開発実践演習では，チケット駆動開発と呼ばれる開発方法論に基づき，詳細設計から実装・テストに至る工程がプロジェクト形式で行われる．チケット駆動開発は，一人の開発者が一つの成果物を対象として行う一種類のタスクをチケットとして登録し，開発者の割り当てやタスク開始の宣言といったチケットに対する操作によって開発を進めていく開発手法である．各受講生は，一人の受講生が一つの成果物を対象として一度に行う活動という粒度で，タスクを行い，その内容や成果物，開始・終了時刻といったタスクに含まれる情報がチケットとして記録される．

このように様々な種類の SDPBL が主に情報系の大学を中心とした教育機関で行われている．なお，以降では我々が行っている SDPBL を対象として提案を行う．

2.2 ソフトウェア開発 PBL における振り返り

教育機関ごとに内容が異なる SDPBL において，どこの教育機関でも必ず行う活動として振り返りがある．振り返りとはプロジェクト中あるいは終了後にプロジェクトメンバ自身によって行われるプロジェクト改善のための自主的な活動で，メンバ全員によるミーティングの形式で行われる [3]．ミーティングによって，プロジェクトを進めていく上での問題点を洗い出し，次のプロジェクトに向けた改善点の提案についての同意をメンバ間で得る．提案された改善点は次の振り返りのときにその有効性が評価される．このような活動を繰り返すことにより，プロジェクトの質を高めることが可能となる．

代表的な振り返り手法として，KPT 法 [4] が提案されている．KPT 法では，効果があったため引き続き行うこと (Keep)，問題があったので改善すべきこと (Problem)，次回から行いたいこと (Try) の 3 つの観点について，メンバ間で議論が行われる．プロジェクトにおける問題の発見や改善点の提案に対する効果の確認のためには，プロジェクトで各受講生が行ったタスクが記録されていることが重要である．例えば，納期遅れという問題とその原因を調査するためには，特に時間のかかったタスクが何であるかという情報が必要となる．また，タスクの割り当てに無駄がなかったかどうかを調べるためには，ガントチャートのような形式で，誰がいつ何をやっていたのかといった記録

が重要である．実際に松澤ら [9] は，プロジェクト中の全てのタスクについて分単位の作業時間を受講生に入力させることで，プロジェクトの進捗と予定されたスケジュールとの乖離を分析する手法を SDPBL で行っている．我々が実施している開発実践演習においても，チケット駆動開発によって得られた受講生一人ひとりのタスク記録を用いて，納期やタスク割り当てといった複数の観点に基づいて，振り返りを支援する取り組みを行ってきた [10].

一方で，チケット駆動開発における一タスクのような粒度の細かい情報は誤って記録されることも多い．特にタスクの着手時刻や終了時刻，開発時間といった時間に関する情報は，タスクの粒度が細かくなればなるほど正確な記録が難しくなる．また，受講生にタスク記録を正確に行うよう指導するためには，講師自身が正確な値を認識してなければならないが，そのためには非常に大きなコストが講師にかかってしまう．そこで本研究では受講生のプロジェクトの進捗状況と受講生によって行われたタスクの記録が正確であるかを講師が容易に評価するためのプロジェクトモニタリング環境を提案する．受講生によるタスク記録の正確性を評価するためには，実際に受講生がどのような点において記録を誤る傾向にあるのかを分析する必要がある．そこで我々は実際に行われた SDPBL において，どのような種類の誤りがどの程度含まれるかを明らかにするべく，予備実験を行った.

3. 予備実験：タスク記録に含まれる誤り分析

3.1 実験設定

本研究において我々は，2011 年度に 3 週間にわたって行われた SDPBL「開発実践演習」において受講生が行ったタスク記録を対象として，どのような種類の誤りがどの程度の量含まれるかの分析を行った．開発実践演習では 5~6 名からなる 6 グループの受講生らが，与えられた詳細設計書に基づいて実装・レビュー・単体試験・結合試験といったタスクを成果物ごとに遂行し，原則として全てのタスク情報をチケットとして記録した．チケットの記録にはチケット管理システム「Trac」[11] を利用している．また，本演習で記録されたチケットは 1 グループあたり平均 240 枚程度であった．開発実践演習で記録されたタスク情報は下記に示す通りである.

a. 着手時刻:　タスクに着手した日時
b. 終了時刻:　タスクを終了した日時
c. 担当者:　タスクの担当者名
d. 見積時間:　タスク実施に必要と見積もられた時間
e. 総時間:　タスク完了にかかった時間
f. 成果物:　タスクで作成・編集される対象物
g. 種類:　タスクの種類
h. マイルストーン:　このタスクを含むあるまとまりのタスク群が実施されるべき期限

本予備実験では，開発実践演習終了後に各グループより受講生を 1 人選び，自分が所属したグループの全てのタスク情報について誤りが存在しないかを調査してもらった．調査の際には，講師より Trac に記録された全てのチケットに含まれるタスク情報と開発実践演習で受講生らが用いた版管理システムのログ情報を受講生に提示した．ここで版管理システムのログ情報には，check-out，commit，update といった版管理システムに対する受講生の全ての操作内容が含まれている．調査では，受講生が講師によって与えられたタスク情報および版管理システムのログ情報と，必要であれば受講生自身が記録していた受講生間のコミュニケーションの記録 (Skype 等のチャットツールが利用されていた）を利用し，全てのタスク情報の誤りを指摘してもらった．受講生によるタスク情報の誤り指摘終了後に，指摘された誤りの集計を行った.

誤り分類では 1. タスク情報誤り，2. タスク粒度誤り，3. タスク記録漏れの 3 種類のカテゴリを規定した．タスク情報誤りはタスク情報に含まれる a~h までの要素が誤ってチケットに入力されていることを表す．タスク粒度誤りは，2(a) の “記録されたタスクの粒度が大きすぎる” あるいは 2(b) の “小さすぎる場合” を示す．我々は開発実践演習において，記録されるべきタスクの粒度を 2.1 節で述べた通り,「一人の受講生が一つの成果物を対象として一度に行う活動という粒度」と規定し，受講生らに通知した．この定義と異なる粒度で記録されているチケットがあった場合に，誤りと分類した．タスク記録漏れは，3(a) の “実施されたにも関わらず記録されていないタスク” や 3(b) の “実施されていないにも関わらず記録されているタスク” があった場合に分類される項目を示している.

3.2 実験結果

予備実験の結果を表 1 に示す．受講生グループによって記録された合計 1419 件のタスク記録のうち，1 つでも誤りを含むタスク記録は 570 件であった．なお，誤りを含むタスク記録件数の中に分類 3(a) の “記録されていないタスク” の件数は含まれていない.

結果より，どのグループでも多かった誤りは 1(a) 着手時刻，1(b) 終了時刻，1(d) 見積時間，1(e) 総時間，の時間に関する誤りの 4 種類と 2(a) の記録されているタスク粒度が大きすぎるものであった．着手時刻は，受講生がチケットに対して着手という操作をしたときに記録される．この操作は本来実際にタスクを開始する際に行われるべきものであるが，着手操作を行うのを失念しているケースが多くあった．終了時刻についても同様に，タスク終了時にチケットに対する終了操作をするのを忘れているケースが存在した．見積時間および総時間の項目については，タスク開始時あるいは終了時にそのタスクを遂行するのに受講生が要すると推測した時間，あるいは実際に要した時間を手入力する形式になっていたが，入力値が 0 のままになっているものが非常に多く存在した．着手・終了と同様入力漏れがあったものと考えられる．タスク粒度については，一度終了したタスクに対応するチケットを使いまわすことで，見かけ上のタスク数が少なくなっている事例が散見された．例えば，ある成果物の実装タスクが終了し，対応するチケットに対して終了という操作を行ったあとにも関わらず，レビューやテストで検出されたバグ修正タスクの記録に，同じチケットを使いまわしている事例があった.

これらの誤りのうち見積時間については，受講生の想定した

表 1　予備実験の結果

	記録されたチケット数	誤りを含むチケット詳細												誤りを含むチケット数
		1. タスク情報誤り								2. タスク粒度誤り		3. タスク記録漏れ		
		(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(a)	(b)	(a)	(b)	
G1	208	18	11	6	4	25	0	1	0	22	4	2	0	67(32.2%)
G2	302	16	5	5	10	47	2	0	2	15	0	1	1	88(29.1%)
G3	270	2	10	1	23	72	0	1	0	34	0	0	1	112(41.4%)
G4	242	29	6	3	49	74	0	1	1	57	4	0	5	146(60.3%)
G5	182	24	16	0	26	38	6	2	0	19	0	0	1	83(45.6%)
G6	215	34	8	1	8	36	0	8	1	8	0	9	1	74(34.4%)
合計	1419	123	56	16	120	292	19	13	4	155	8	12	9	570(40.1%)

値が記録される項目であるため，タスク記録より誤りを推定することは非常に困難である．そのため，本研究では入力が行われていない場合のみを誤りの対象とした．また，タスク粒度以外の誤りについては，入力漏れのものや，チケット情報や版管理システムのログ情報から明らかに間違っていると推定されるものしか今回の調査では検出されていない．そのため実際にはもっと多くのチケットに誤りが含まれている可能性がある．

以上の分析から分かるように，手作業で行われるタスク記録には誤りが含まれる可能性が非常に多い．そのため，SDPBL における振り返りを指導する際にはこれらの誤りが含まれていることを講師が認識し，適切な記録方法や誤りそのものの指摘を振り返り前に行う必要がある．そこで本研究では，予備実験で明らかになった 5 種類の誤りのうち，見積時間を除く 4 種類の誤りについて，誤っていることを講師が検知できるように支援するシステムを構築する．

4. DaaS を用いたソフトウェア開発環境における細粒度プロジェクトモニタリング

我々はタスク記録における誤りを講師が検証できるようにすることを目的とした細粒度プロジェクトモニタリング環境を提案する．提案する細粒度プロジェクトモニタリング環境は DaaS(Desktop as a Service) を利用して構築されるソフトウェア開発環境とサーバ上に配備される版管理システム等の開発支援環境およびこれらの環境それぞれに組み込まれた開発ログ取得機能によって構成される．以降では，それぞれの環境とその環境における開発ログ取得機能について詳述する．

4.1　ソフトウェア開発環境における開発ログ取得

DaaS とはクラウド上に仮想化されたデスクトップ環境を配置し，インターネットを介してユーザに提供するための枠組みである．インターネット越しにデスクトップ環境を利用できるようにすることで，一定のセキュリティを確保した状態で共通の開発環境をどこからでも利用することが可能となる．DaaS を用いることで，受講生のタスク遂行時の開発作業をログとして収集する機能を組み込んだ共通のソフトウェア開発環境を容易に構築することが可能となった．我々のソフトウェア開発環境で収集される開発ログは下記の 2 種類である．

L1(アプリケーション操作ログ)： 受講生が任意のアプリケーションの操作を開始・終了した時刻を表すログ．

L2(ファイル操作ログ)： 受講生がファイルに対して行った追記・削除・編集の内容とその日時を記録したログ．

SDPBL では，受講生が実施するタスクの種類に応じて，利用するアプリケーションが異なることがある．例えば，実装・単体テストは統合開発環境を用い，レビューはテキストエディタや表計算ソフトを用いるといったケースが存在する．そのため，受講生が実施するタスクの種類が事前に分かっていれば，アプリケーション操作ログによって作業開始・終了時刻や総時間の推定を支援できると考えられる．

一方で，受講生がタスクを実行する際，複数のアプリケーションを利用したり，複数ファイルを参照したりすることがある．例えば，レビューを行う際にはレビュー対象のファイルと仕様書やチェックリストを参照しつつ，テキストエディタ等にレビュー結果を記述する．そのため，アプリケーション操作ログのみでは，正確に受講生がどのタスクを実施しているかを推定することは困難である．そこで，L2 としてファイルに対する編集作業の内容をログとして収集する．L2 のログによって，受講生がどの成果物を対象としてタスクを実施しているかを推定できるようになる．

L1，L2 のログにより，DaaS 上のソフトウェア開発環境を利用した開発ログ収集によって，受講生がどの成果物を対象としてどのアプリケーションをいつからいつまで操作していたかを確認することが可能となる．

4.2　サーバ側開発支援環境を用いた開発ログ取得

SDPBL では，統合開発環境やエディタといったソフトウェア開発環境だけでなく，チームでの開発を支援する開発支援機能が用いられることが多い．我々が想定する SDPBL では，チケット管理システムと版管理システムが開発支援環境に相当する．開発支援環境における受講生の操作内容を下記に示すようなログとして収集することで，受講生に現在割り当てられているタスクの情報や成果物の開発状況を確認することができる．

L3(チケット情報ログ)： チケット管理システムに対して行われたチケット情報およびチケットに対して行われた操作ログ．

L4(版管理システム操作ログ)： check-out， commit，update といった版管理システムに対して行われる受講生の操作ログ．

3. で述べた予備実験で，タスク情報における担当者や成果物の誤りは比較的少ないことが分かった．そのため，チケットとして登録されたタスクの担当者と成果物が正しいと仮定するこ

とで，L3 によって特定の受講生が実施すべきタスクの内容を推定することが可能となる．また，チケットに対する着手操作や終了操作のログ情報によって，受講生の意図するタスク開始・終了時刻を取得できる．

我々が想定する SDPBL では，原則として全てのタスクにおいて成果物の作成・編集が行われる．そのため，タスクの終了は版管理システムに対して成果物を登録・更新する日時とほぼ等しくなる．L1，L2 のログと L4 のログを合わせることで，ある受講生の特定の成果物に対するタスクが終了した日時の推定が可能となると考えられる．

以上のログを分析することで，L3 によって受講生に割り当てられているタスクの情報を認識し，L1，L2 によって対象成果物に対する編集内容およびタスクの実施時間を推定し，L4 によってタスク終了時間を検知することが可能となる．結果として，完全に自動でタスク情報を記録することは難しいとしても，受講生によって記録されたタスク情報との間の齟齬を検知することは可能となると考えている．

我々はここで提案したログ収集機能を備えたソフトウェア環境およびソフトウェア開発支援環境のプロトタイプを実際に構築し，実験を行った．次節ではプロトタイプシステムの紹介とプロトタイプシステムを用いて行った実験内容をケーススタディとして示す．

5. ケーススタディ

ケーススタディとして，本稿で提案するプロジェクトモニタリング機能を備えたソフトウェア開発環境およびソフトウェア開発支援環境を edubase Cloud [12] を用いて実装し，実際に小規模なグループ開発を行った．本章ではプロトタイプとして構築した各環境とグループ開発の内容および提案環境によって収集されたログ情報について詳述する．

5.1 ソフトウェア開発環境および開発支援環境

ソフトウェア開発環境および支援環境のプロトタイプを以下に示す．

ソフトウェア開発環境

仮想デスクトップ仕様

- OS:Windows XP Professional 32bit
- CPU:Intel Xeon 2.4GHz 2 コア
- Memory:4GB，HDD:20GB

本ケーススタディでは全ての被験者がこの仕様の仮想デスクトップにリモートデスクトップ接続機能を利用してアクセスし，ソフトウェア開発を行う．

導入ソフトウェア

- Java SE Development Kit 7 Update 5
- Eclipse IDE Juno for Java EE Developers
- MySQL Community Server 5.5.24
- Apache Tomcat 7.0.28
- ManicTime [13]
- Dropbox [14]

JDK，Eclipse，MySQL，Apache Tomcat はそれぞれ言語環境や統合開発環境，データベースサーバにアプリケーションサーバといったソフトウェア開発に用いられるソフトウェアである．また，ManicTime は L1 のログを，Dropbox は L2 のログを収集するために導入した．ManicTime では，アプリケーションの稼働状況，ドキュメントへのアクセス状況についてログを取ることができる．アプリケーションの稼働状況ログは，アプリケーション名，プロセス名，稼働開始日時，稼働終了日時，稼働時間からなる．ドキュメントへのアクセス状況ログは，アクセスされていたドキュメント名，アクセス開始時刻，稼働終了時刻，アクセス時間からなる．Dropbox はクラウドを用いたファイル共有サービスである．Dropbox は，ファイルが編集され，保存されるたびに，各ファイルが編集された時刻と編集されたファイルそのものを記録することができる．

ソフトウェア開発支援環境

- Trac 0.11.7.ja1
- Subversion 1.6.11-2

Trac をチケット管理システムとして，Subversion を版管理システムとして利用した．Trac におけるチケット情報を L3 のログとして記録し，Subversion に対して行われた受講生の操作ログを L4 として記録する．本ケーススタディでは，これらを CentOS 6.0 の搭載されたサーバ上で用いた．

5.2 グループ開発内容

本ケーススタディでは，事前に用意された Web アプリケーションのひな形に対して，詳細設計書に基づいてある機能を追加するという開発プロジェクトにグループで取り組んでもらった．被験者は大学教員 4 名で，全員が Java 言語による Web アプリケーション開発経験とチケット管理システム利用経験を持っている．各被験者が取り組んだタスクは実装・レビュー・単体試験・結合試験の 4 種類で，対象成果物は実装対象の Java ソースコード，単体試験の Java ソースコード，コードレビュー報告書，試験報告書の 4 種類であった．開発中は行われたタスクをより正確に把握するため，別途チャットシステムを用いて，タスクの開始・終了時刻を被験者が宣言し，記録した．

5.3 グループ開発結果

グループ開発は 4.5 時間かけて行われた．また，Trac に記録されたタスク数は 26 件で，これはチャットシステムで確認されたタスク数と一致した．被験者全員がチケット管理システムを用いた経験があることとチャットシステムによって開発進捗を厳密に管理していたこともあり，今回の開発ではチケットに対する操作もほとんど誤りなく行われた．

一人の被験者について，チケットに対する着手を忘れていた事例があったため，着手忘れを提案システムによって検知できるか評価を行った．図 1~4 は，対象の被験者が着手を忘れていた時刻前後の L1~L4 のログを表している．チケット管理システムにおいて記録されていた，被験者 X が着手を忘れていたタスクの情報を図 3 に示す．タスク情報から，担当者が X であり，対象タスクの種類はコードレビューで，作成する成果物が classAreview.txt であることが分かる．図 2 から被験者 X が classAreview.txt を作成した時刻が 2012/7/6 13:40 であると分かる．さらに，図 1 から classAreview.txt ファイルを eclipse で 13:40 に作成し，開いていることが分かるため，この時刻が

着手時刻である可能性が高いといえる．このあと被験者 X は図 2 のログより，13:51 まで作業を続け，図 4 のログより 13:52 にコミットし，タスクを終了していることが分かる．図 1 のログより，途中で休憩を挟んでいる様子も見られないので，タスクの総時間は約 12 分であったと判断できる．

以上の分析より，図 3 のタスク情報は着手時刻が約 20 分遅れており，終了時刻が 8 分遅れていることが分かる．また，総時間も実際は 0.2 時間であることから，0.1 時間少ない値が記録されていたことが分かる．

```
新規ファイル 2012/7/6 13:40 2012/7/6 13:40 0:00:08 eclipse
classAreview.txt 2012/7/6 13:40 2012/7/6 13:41 0:01:23 eclipse
classA.java 2012/7/6 13:41 2012/7/6 13:42 0:01:11 eclipse
classAreview.txt 2012/7/6 13:42 2012/7/6 13:45 0:02:37 eclipse
：(省略)
classA.java 2012/7/6 13:51 2012/7/6 13:51 0:00:06 eclipse
classAreview.txt 2012/7/6 13:51 2012/7/6 13:51 0:00:31 eclipse
コミット 2012/7/6 13:51 2012/7/6 13:52 0:00:14 eclipse
classAreview.txt 2012/7/6 13:52 2012/7/6 13:52 0:00:28 eclipse
```

図 1　L1:被験者 X のアプリケーション操作ログ

```
バージョン 1 DeveloperX による Added 2012/7/6 13:40 0 bytes
バージョン 2 DeveloperX による Edited 2012/7/6 13:50 27 bytes
バージョン 3 DeveloperX による Edited 2012/7/6 13:51 79 bytes
```

図 2　L2:classAreview.txt のファイル操作ログ

```
着手時刻:2012-07-06T14:00:04+0900
終了時刻:2012-07-06T14:00:09+0900
担当者:DeveloperX
見積時間:0.3 時間
総時間:0.1 時間
成果物:classAreview.txt
種類:コードレビュー
マイルストーン:2012-07-06T17:00:00+0900
```

図 3　L3:着手忘れタスク情報ログ

```
[06/Jul/2012 : 13 : 38 : 20] DeveloperX update /project r13
[06/Jul/2012 : 13 : 50 : 50] DeveloperX update /project r15
[06/Jul/2012 : 13 : 52 : 05] DeveloperX commit r16
```

図 4　L4:被験者 X の版管理システム操作ログ

5.4 考　　察

タスクが対象とする成果物の履歴を L1 で取得し，L2 によってより詳細に分析することにより，タスクの時間に関する属性，すなわち着手・終了時刻および，総時間を推定することができた．この推定に基づき，チケットに記録された着手・終了時刻および総時間と比較した結果，一部のチケットにおいて入力された記録と異なることが分かった．チャットシステムの履歴からも，該当するチケットに入力された記録が誤りであることが確認されており，本手法による誤りの検知の可能性を確認することができた．一方，タスクの粒度誤りについては事例数が不十分であったためプロトタイプシステムで対応が可能であるか確認できなかった今後粒度誤りの事例においても，誤りを検知できるか分析していきたい．

誤り検知のためのログの収集は，同じ設定でインストールされたツールにより行われる必要がある．特に，各自のデスクトップ上にインストールされるソフトウェア開発環境は，個別に設定をしなければならない．各自が利用するデスクトップ環境を DaaS を利用して提供することで，L1 および L2 のログを収集する機能を含んだ共通のソフトウェア開発環境を受講生に容易に利用させることが可能となった．

6. まとめと今後の課題

本研究では，SDPBL において受講生により記録されたタスク情報の正しさを，講師が検証できるようにすることを目的とした細粒度プロジェクトモニタリング環境を提案した．提案に基づいて，開発ログを取得するための環境を DaaS 上に構築し，ケーススタディを実施した結果，予備実験において多かった，時間に関する誤り（着手・終了時間および総時間）を検出し，正しい値を推定することができた．一方で，タスク粒度の誤りについては今回の事例には含まれなかったため，検証することはできなかった．

今後の課題としては，タスク情報における誤り判定の自動化が挙げられる．また，実際の SDPBL に適用することにより，本手法のさらなる検証と有効性の確認が必要である．

謝辞　本研究は，日本学術振興会科学研究費補助金若手研究（B）（課題番号:24700030）の助成を得た．

文　　献

[1] 独立行政法人情報処理推進機構，“IT 人材白書 2010,” 2010.
[2] 松澤芳昭，杉浦　学，大岩　元，“産学協同の PBL における顧客と開発者の協創環境の構築と人材育成効果,” 情報処理学会論文誌，vol.49，no.2，pp.944–957，2008.
[3] 天野　勝，“プロジェクトファシリテーション実践編振り返りガイド第 20 版”.
[4] A. Cockburn, Agile Software Development, Addison-Wesley, 2003.
[5] まちゅ，“チケット駆動開発 … itpro challenge のライトニングトーク (4),” Sept. 2007. http://www.machu.jp/diary/20070907.html.
[6] 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.
[7] H. Batatia, A. Ayache, and H. Markkanen, “Netpro: an innovative approach to network project based learning,” Proc. International Conference on Computers in Education (ICCE’02), pp.382–386, 2002.
[8] 沢田篤史，小林隆志，金子伸幸，中道　上，大久保弘崇，山本晋一郎，“飛行船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.
[9] 松澤芳昭，塩見彰睦，秡川友宏，酒井三四郎，“ソフトウェア開発の教員主導型 PBL における反復プロセスと EVM 導入の効果,” 情報処理学会研究報告. コンピュータと教育研究会報告，第 2009 巻，pp.1–8，May 2009.
[10] 福安直樹，佐伯幸郎，水谷泰治，“リポジトリのリアルタイムな可視化にもとづく PBL の支援環境～継続的な実施を目的として～,” 電子情報通信学会技術研究報告，vol.110，no.458，pp.121–126(SS2010–73)，2011.
[11] “Trac”. http://trac.edgewall.org/ 参照 Jul. 6,2012.
[12] “edubase Cloud”. http://edubase.jp/cloud/ja/ 参照 Jul. 6,2012.
[13] “Manictime”. http://www.manictime.com/ 参照 Jul. 6,2012.
[14] “Dropbox”. https://www.dropbox.com/ 参照 Jul. 6,2012.