自動計測データと機械学習に基づくソフトウェア開発の作業目的の推定

大橋 亮太　上野 秀剛　門田 暁人　荒木 健史　山田 欣吾

松本 健一

本稿はソフトウェア開発者の能力向上・プロセス改善手法である PSP における作業履歴の記録と作業目的に基づいた履歴の分析を支援するために，作業者の作業履歴から作業目的を推定する手法を提案する．提案手法は推定対象となる作業 (Task) の前後に行われた作業の特性から作業目的 (Aim) を推定する．連続する Task のうち，打鍵数やクリック数が多い Task をその Aim における主要 Task とし，特性として使用する．RandomForest による推定モデルを構築し実験した結果，提案手法は前後の Task の情報を用いない場合に比べ，高い精度で作業を推定できた．特に，前後の作業を多く見るほど推定精度の向上が見られ，推定の正答率は 97 %となった．提案手法を用いることで，実装やテストといった開発目的にどの程度の時間が費やされたかを把握することが可能となり，プロセス改善の基礎データとして役立てることができると期待される．

In this paper we propose a method to support Personal Software Process (PSP), which is a well known software process improvement framework for individual developers. The proposed method estimates developer's purposes (aims) from time-series data about developer's tasks, given by an execution history of software applications. We implemented the method by a machine learning algorithm, Random Forests. The experiment result shows the prediction with the time-series data is more accurate than the prediction without the time-series data. Especially, when using longer time-series data, accuracy of estimation became 97 %. It can be expected that the proposed method can help developers' process improvement as they become aware of how much time they spent on a specific aim such as implementation and testing.

1　はじめに

Tom DeMarco の「計測できない物は制御できない」という格言に代表されるように，開発プロセスの制御や改善には計測が必須である [1]．開発現場で生じる問題の多くは人的要因に起因するものであるから，プロダクトやプロセスを計測するよりも，開発の主体である人間やその作業を計測し，改善につなげることが自然であると考えられる．

ソフトウェア開発者の能力向上やプロセス改善を目的として設計された Personal Software Process (PSP) が提唱されている [2]．PSP は開発者の作業履歴を記録し，実装やテスト，設計，会議といった個々の開発作業にどれだけの時間を費やしているか分析し，効率の改善やプロセス改善に役立てる手法である．

これまでに PSP における作業履歴の記録を容易にするための支援システムが複数提案されている †1 †2 †3．これらのシステムは開発者が作業に費やした時間

Task Purpose Estimation in Software Development Based on Automatic Measurement Data and Machine Learning

Ryota Ohashi, Kenichi Matsumoto, 奈良先端科学技術大学院大学 情報科学研究科, Nara Institute of Science and Technology.

Hidetake Uwano, 奈良工業高等専門学校, National Institute of Technology, Nara College.

Akito Monden, 岡山大学, Okayama University.

Kenji Araki, NCS&A 株式会社, NCS&A CO.,LTD.

Kingo Yamada, 株式会社ファインバス, FineBus Co., Ltd.

コンピュータソフトウェア, Vol.29, No.1 (2012), pp.78–84.

[研究論文] 2011 年 12 月 15 日受付.

本論文は第 21 回ソフトウェア工学の基礎ワークショップ (FOSE2014) の発表論文を発展させたものである

†1　Process Dashboard, http://www.processdash.com/

†2　Task Coach, http://members.chello.nl/f.niessink/

†3　SlimTimer Time Tracking without the Timesheet,

を自動で計測できるが, どのような作業を現在行っているかは, 手動で入力する必要がある．そのため, データの取り忘れや計測に気を取られ作業に集中できないといった問題がある．

一方で PC 上におけるソフトウェアの使用を計測するシステムとして TaskPit や Manic Time がある [3][†4]． これらのツールは使用しているソフトウェアの名前やウィンドウ名，またはそれに関連づけられた作業名を記録すると同時にソフトウェア毎の作業時間や打鍵数，クリック数を記録する． 計測者はシステムで記録された作業履歴を設計や実装といった開発作業の種類ごとに集計することで個々の開発作業にどれだけの時間を費やしているか分析できる．このとき，同じ作業であっても，その目的によって異なる種類に集計する必要がある．例えば設計書に対する作業があった場合，設計書を作成するための作業であれば設計に，実装の過程で設計を確認するための作業であれば実装として分類する．開発プロセスを改善するにあたって，作業の目的に基づいた履歴分析が必要になる．このような作業の目的への分類はシステム側では判断できず，計測者が手動で行う必要があるため，計測者に負担がかかることやデータの漏れが発生する問題点がある．

本稿では TaskPit が計測する作業を Task，開発者が Task を行っている目的を Aim と定義し，Task の Aim を自動で推定する手法を提案する． Task の Aim を自動推定する事で，開発者の作業を妨げることなく Aim 毎の作業時間や各 Task の作業履歴を収集し，プロセス改善に利用できるようになる． 提案する手法は，推定対象 Task と前後に実行される Task の特徴から，機械学習アルゴリズムである Random Forests を用いて Aim を推定する．各 Task の特徴として，Task 名，打鍵数やマウスクリックの数を選択した．また，各 Aim にはそれぞれ打鍵数やクリック数が多い Task（主要な Task）があると考え，推定対象 Task 前後の Task に対して，作業量毎に順位付けを行い，特徴として使用する．

http://www.slimtimer.com/

†4 Manic Time, http://www.manictime.com/

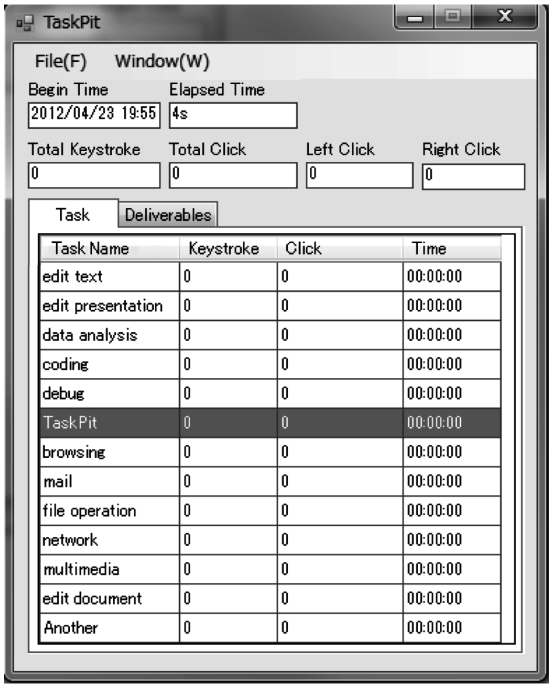


図 1 TaskPit

2 関連研究

2.1 PSP 支援システム

PSP とはソフトウェア開発者が開発作業に費やした時間を計測し，分析する事で開発プロセスを改善する手法である [2]．PSP を実施する開発者は測定する Aim を定義した上で作業履歴を計測し，見積りや予定との差異を分析することで Aim に対応するプロセスの問題を評価・改善する．

開発作業に費やす時間を計測するための支援システムとして Process Dashboard，Task Coach，Slim Timer などがある．これらのシステムは PC 上での作業時間を自動で計測できる一方で，遂行中の作業が切り替わった際に手動で入力をする必要がある．そのため，計測データに漏れが発生し，不正確なデータを記録してしまうといった問題がある．

そこで，作業の計測，判別を自動化したシステムとして TaskPit がある．図 1 に TaskPit のスクリーンショットを示す．TaskPit はアクティブになっているソフトウェアのウィンドウ名と作業 (Task) を関連

Task Name,	Begin Time,	End Time,	Lclick,	Rclick,	Keystroke
design document,	13:25:35,	13:28:48,	11,	2,	98
Req. specification,	13:28:48,	13:29:54,	7,	0,	0
design document,	13:29:54,	13:33:48,	23,	3,	120
desktop,	13:33:48,	13:33:56,	1,	0,	0
design document,	13:33:56,	13:34:48,	4,	0,	75
data analysis,	13:34:48,	13:36:41,	24,	0,	64
desktop,	13:36:41,	13:36:59,	2,	0,	0

図 2　作業履歴例

づけ，作業時間や打鍵数，マウスクリック数を記録する．例えば，“MS Word” で 2 分間作業を行った後，“Open Office Writer” において 3 分間作業を行った場合，5 分間の “書類閲覧編集”Task として記録される．これらの Task の判別は，TaskPit の設定でソフトウェアと Task 名の紐付けをしておくことによって可能となっている．図 2 に TaskPit で計測された作業履歴の例を示す．作業履歴には Task 名，開始時刻，終了時刻，左クリック数，右クリック数，打鍵数が含まれる．

同様に作業の計測を自動化したシステムとして Manic Time がある．Manic Time は使用しているソフトウェア名やウィンドウタイトルの変化を検出し，それらの名前と費やした時間を自動的に記録する．しかし，これらのシステムは各 Task の Aim を記録できないため，プロセス改善をする際には各 Task の Aim をインタビューなどの方法を用いて手動で補完した上で分析する必要がある．そのため，計測者への負担が大きく長期間の分析が困難である．

2.2　作業推定と分類

PC 上での作業支援や作業計測を目的とした作業推定と分類に関する研究がされている．Michael らは PC 上のアクティブなウィンドウから機械学習を用いて，PC 上での作業の目的を推定する手法を提案している [4]．この手法はアプリケーション名やウィンドウタイトル，ウィンドウ内のコンテンツを機械学習に使用している．Stumpf らは TaskTracker と TaskPredictor といったシステムを開発している [5]．TaskTracker は PC 上での操作を記録し，TaskPredictor は操作履歴から操作目的を推定する．TaskPredictor は操作の種類や時間，Window ID などを特徴として操作目的を推定している．Oliver らは SWISH と呼ばれるシステムを提案している [6]．このシステムはウィンドウタイトルやウィンドウの遷移順序を用いて作業目的を推定する．PC 上での作業を円滑にするために開発された TaskPose と呼ばれるシステムがある [7]．このシステムではクラスタリングを用いてウィンドウのグルーピングを行い，関連性のあるウィンドウを互いに近くにレイアウトしたり，重要なウィンドウは大きく表示することで，ウィンドウ間の移動を支援し，作業の円滑化を図っている．これらの既存研究には，推定対象となる作業に加え，前後の作業情報を用いている手法もあるが，目的に対して主要な作業を判別する特徴は使用していない．

3　提案手法

本研究ではプロセス改善の支援を目的として，Task の Aim を自動推定する手法を提案する．提案手法は，TaskPit が記録した Task を作業量で順位付けし，Aim に対する主要な Task が判別できる特徴を用いることで，推定精度の向上を目指す．

本章では Task と Aim の定義をした後，機械学習アルゴリズムの Random Forests を説明し，前後の Task に関する情報を用いた提案手法について述べる．

3.1　Task

Task とは 1 つのソフトウェアに対する連続したユーザの操作である．TaskPit はアクティブなアプリケーションに対する操作を監視し，連続した操作を 1

つの Task t として記録する．Task はアクティブ状態になっているウィンドウの実行ファイル名とタイトル名から識別される．同じ Task に対して複数の異なるアプリケーションが利用される場合，それらを区別せず，同じ Task として記録する．また，Web ブラウザのような複数の Task（メールやグループウェア）で利用されるアプリであっても，ウィンドウタイトルによって異なる Task として識別する．Task， アプリケーション， ウィンドウタイトルの関係を Extended Backus-Naur Form (EBNF) で示す．

```
<Task> ::= <Application>{|<Application>}
<Application> ::= <exe name> [<window title>]
```

記録される t の特徴を以下に示す．

- Task Name
- Left Clicks
- Right Clicks
- Keystrokes

Task Name は exe のファイル名とウィンドウタイトルから識別される Task の名称を表わす．Left Clicks, Right Clicks，Keystrokes はそれぞれ左，右クリック数，打鍵数を表わす．なお打鍵内容については記録しない．提案手法では，Task 名は実行されていた Task の種類を，左，右クリック数， 打鍵数からは Task の作業量を判別するのに用いる．

3.2 Aim

Aim とはユーザが Task を行う目的であり，各 Task t は Aim を達成するために行われる． 本研究では連続した t の集合 T で構成される Aim A を $AIM(T)$ として定義する．

$$A = AIM(T), T = [t_0, t_1, t_2, \cdots, t_n] \qquad (1)$$

図 3 に Aim が設計書作成である Task 列を示す．図の四角は 1 つの Task を表し，マウス/キーボードアイコンの数は，ユーザの相対的な作業量を表す．ここで例に示した Aim“設計書作成” は “仕様書閲覧編集” と “設計書閲覧編集” の 3 つの Task から構成されており，“設計書閲覧編集” の作業量は “仕様書閲覧編集” と比べて多いことを示している．

3.3 Random Forests

Random Forests は決定木を弱学習器としたアンサンブル学習を用いる機械学習アルゴリズムである [8]．アンサンブル学習とは，高精度でない分類器を複数組み合わせることで精度を向上させる手法である．Random Forests は以下の手順によって学習を行う．

手順 1 与えられたデータセットから *ntree* 組のブートストラップサンプルを作成する．その際，用いるデータセットの約 3 分の 1 はテスト用データとして取り除かれる．このテスト用データは Out-Of-Bag(OOB) データと呼ばれる．残った 3 分の 2 を学習用データとして用いる．

手順 2 作製したブートストラップ毎に分類木を作成し，木の生成に用いていない OOB データを用いてテストを行う．テスト時の誤り率は OOB 推定値と呼ばれる．分類器の構築を行う際の各分岐ノードは，異なる木を多数生成するため，ランダムに *mtry* 個の変数をサンプリングし，その中から最も分岐が良い変数を用いる．

手順 3 分類器は，すべてのブートストラップサンプルの OOB 推定値に基づいて多数決を取る．

Random Forests では，組み合わせるアンサンブル学習に用いる分類器の数 *ntree* と分岐の際に用いる説明変数の数 *mtry* を調整できる．本稿では，説明変数の総数を M としたとき，Breiman の推奨する *ntree* $= 500$， $mtry = \sqrt{M}$ とする [8]．

3.4 前後の Task 情報を用いた推定

ある 1 つの Aim は連続した複数の Task で構成される．Task 列には Aim 毎に，Task の種類や順序，作業量（クリック数や打鍵数）に特徴がある．図 4 に設計，実装，テストを Aim とした Task 列の例を示す． 実装と設計の Task を比較すると，実装は 3 個目

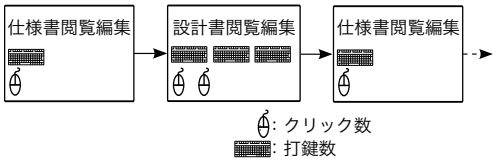


図 3 設計書作成の Task 列

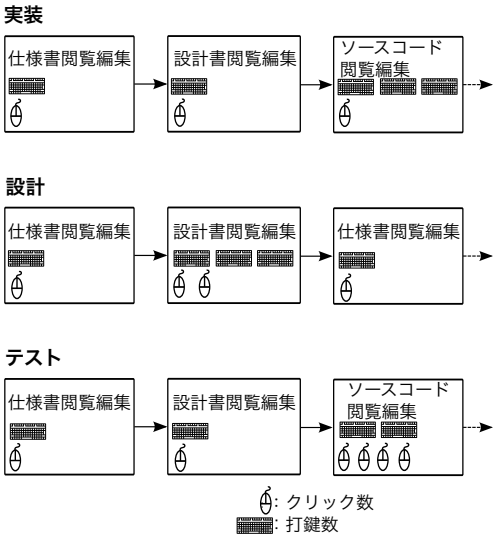


図 4　異なる Aim における Task 列

の Task が “ソースコード閲覧編集”であるのに対して設計では 3 個目の Task が “仕様書閲覧編集”と異なっている．また，実装とテストは同じ Task で構成されているが，実装の “ソースコード閲覧編集”は打鍵数が多いのに対し，テストではクリック数が多い．

Task は種類によっては複数の Aim で実施されるため，Task 名だけではどの Aim に分類するべきか判断できない．例えば，図 4 において “設計書閲覧編集”は全ての Aim で実施されるため，Task の名前や作業量のみから Aim を推定することは難しい．しかし，Aim を構成する Task 列に対して，クリック数や打鍵数が多い Task(主要な Task) や特有の Task の有無など，複数 Task の特徴を推定に用いることで，各 Task の Aim を判別できると考えられる．ソフトウェア開発プロセスにおいては特定の行程でのみ使用されるツール（例えばユニットテスト支援ツール）や，ある行程で特によく使われるプログラム（例えば IDE）が存在するため，ある Task の前後に現れる主要な Task は Aim 推定に有用である．

提案手法ではこれらの点に着目し，予測対象 Task と前後 Task の情報から各 Task の Aim を推定する．推定には機械学習アルゴリズムの 1 つである Random Forests を用いる．推定に用いる特徴として Aim の推定対象である Task の情報と，その前後の Task の情報を用いる．推定対象である Task t_p の Aim 推定には Task の集合 $T = [t_{p-x}, \ldots, t_p, \ldots, t_{p+x}]$ を用いる．x は考慮する前後の Task の数を示す．推定に用いる特徴として以下の 8 種類の特徴を選択した．

1. 推定対象 Task 名: $t_p.Name$
2. 推定対象 Task の左クリック数: $t_p.LClick$
3. 推定対象 Task の右クリック数: $t_p.RClick$
4. 推定対象 Task の打鍵数: $t_p.Key$
5. T の各 Task 名: $t_{p-x}.Name$, …, $t_p.Name$, …, $t_{p+x}.Name$
6. T において左, 右クリック数, 打鍵数それぞれの数値が最も高い Task 名: $T.1stLClick$, $T.1stRClick$, $T.1stKey$
7. T において左, 右クリック数, 打鍵数それぞれの数値が 2 番目に高い Task 名: $T.2ndLClick$, $T.2ndRClick$, $T.2ndKey$
8. T において左, 右クリック数, 打鍵数それぞれの数値が 3 番目に高い Task 名: $T.3rdLClick$, $T.3rdRClick$, $T.3rdKey$

提案する手法では Task の Aim が記録された作業履歴に対して，上に挙げた特徴の 5 から 8 を追加し，機械学習のデータセットとして用いる．その後，学習したアルゴリズムを用いて推定対象である作業履歴に記録された Task の Aim を推定する．なお，6 から 8 の特徴については，該当 Task が存在しなかった場合，”none”として処理される．

4　実験

4.1　実験環境

実験にはソフトウェア開発組織の開発者 3 名の作業履歴を使用した．開発者は組織で割り当てられた PC を用いて実装やテストといった作業をしている．開発者が組織で使用している Windows PC に TaskPit をインストールし，34 営業日の間の Task を計測した．記録される Task は事前のインタビューから設定した．図 5 に Task の設定例を示す．

計測後，インタビューと計測されたウィンドウタイトルからそれぞれの Task の Aim を人手で入力した．以下に Aim を示す． なお，その他は複数の Aim の集合ではなく，単一の Aim である．業務内容の詳細

```
ネットワーク = FFFTP.exe|ttermpro.exe
TaskPit = TaskPit.exe|TaskAnalyzer.exe
ブラウザ = iexplore.exe|firefox.exe|Safari.exe|chrome.exe
メール = thunderbird.exe|Outlook.exe|iexplore.exe:gmail
ファイル操作 = explorer.exe
調査 = sakura.exe|noediter2.exe
SQL サーバ = SqlWb.exe
Excel = EXCEL.EXE|scalc.exe
Word・Powerpoint = WINWORD.EXE|POWERPNT.EXE
OOo = soffice.exe
文書閲覧 = AcroRd32.exe
辞書閲覧 = cjdic.exe|xdict.exe
テスト = mstsc.exe|Beyond32.exe|DF.exe|reverse.exe|reverseserver.exe|
perfmon.exe|cmd.exe|WinMergeU.exe
プログラミング・デバッグ = eclipse.exe|devenv.exe|VPC.exe|VMWindows.exe|hh.exe
trac = iexplore.exe:#|firefox.exe:#|Safari.exe:#|chrome.exe:#
キャプチャ = snagit.exe
メッセンジャー = ipmsg.exe
```

図 5 Task の設定例

表 1 Task 数の内訳

	開発者 A	開発者 B	開発者 C
実装日	553	1555	951
テスト日	1304	597	1225
合計 Task 数	1857	2152	2176

に触れるため，本稿ではその他として表す．

- 実装
- テスト
- その他
- 休憩

実験には，実装を主な目的とした一日とテストを主な目的とした一日をまとめた作業履歴を用いた．各開発者のそれぞれの日の Task 数の内訳を表 1 に示す．

4.2 評価

提案手法の精度をインタビューに基づいて入力した Aim と提案手法が推定した Aim を比較し評価する．Task の時系列を推定に用いることの有効性を評価するために，推定対象 Task の特徴のみを用いた場合と，推定対象の前後の Task の特徴を加えた場合の推定精度を比較する．実験では，前後の Task 考慮数を 1 から 10 まで変化させたときの推定精度を確認する．

本研究では推定精度の評価に実験データ全体の推定精度と Aim 毎の推定精度を求める．実験データ全体に対する推定精度は以下の式で求められる．

$$All\_accuracy = \frac{\text{全ての Task に対する正答数}}{\text{データセットの Task の総数}} \quad (2)$$

ある Aim A の推定精度は以下の式で求められる．

$$Aim\_accuracy = \frac{\text{A の正答数}}{\text{A を Aim とする Task の総数}} \quad (3)$$

評価実験では，two-fold-cross-validation を用いる．すなわちデータセットの半分を学習用に使用し，残りの半分を推定に使用する．また，各データセットに対し 10 回実験を行い，結果の平均値で評価する．

5 結果と考察

5.1 全体の結果

図 6 に推定精度を示す．縦軸が推定精度，横軸が前後の Task の考慮数を示している．図 6 から前後の Task を考慮した場合の推定精度は，考慮しなかった場合に比べ大幅に向上していることがわかる．考慮数を 0 から 1 にすると，開発者 A では，69.2%から 86.1%と 16.9%精度が向上している．また開発者 B では，72.2%から 85.8%と 13.6%，開発者 C では 87.8%から 95.1%と 7.3%精度が向上している．考慮数を 10 まで増やすと精度がさらに向上し，平均で 97%と高い推定精度が確認できた．

また，図 6 から分かるように，前後の Task を考慮しない場合においても，各開発者のデータにおける推定精度は約 70%以上と高い数値となっている．これは Aim 毎に固有の Task が存在し，それらが一定の割合を占めているからである．例えば，“テスト (Task)” や “キャプチャ”などの Task はテストが Aim の時にのみ出現するため，Task 名から一意に Aim を判別することができる．そのため，図 6 の結果だけでは，推定精度の向上について，複数の Aim で用いられるが強い偏りがある，もしくは，単一の Aim に属する Task の影響が排除できていない可能性がある．

図 7 に複数の Aim に用いられる Task のみに限定した場合の推定精度を示す．図 7 から，複数の Aim に用いられる Task に対しても，提案手法を用いることで，高い精度で Aim が推定できることが分かる．加えて，図 6 と同様に，考慮する前後の Task を増加すると，推定精度が大幅に向上する傾向が確認できた．

図 8 に実験で計測した開発者 A の作業履歴の一部を示す．各行の「Aim(正解)」はインタビューによって確認した作業者の実際の Aim,「前後無し」は前後の Task を考慮しない場合の推定結果,「前後有り」は前後の Task を考慮した場合の推定結果をそれぞれ表す．図に含まれる Task は全て Aim が実装である．また，データセットの半分を学習用に使用し，残りの半分に対して推定をしているため，一部の行には推定が行われていない．

図の 3 行目にある “調査”はエラーログやテストデータの閲覧を表す Task である．この Task に対して,「前後無し」の推定ではテストと誤判定しているが,これは “調査”がテストに頻繁に出現しており，かつ,テストを Aim とする “調査”の数が多いためと考えられる．一方，前後有りの推定ではこの Task に対して正しく Aim を推定できた．この開発者の Task 列では “調査”の前後にクリック数と打鍵数の多い “プログラミング・デバッグ”が頻繁に出現している（図 2,4 行目）．これは開発者が Eclipse のような開発環境を集中的に操作し，その合間に調査（エラーログやテストデータの閲覧）をしていることを表している.「前後有り」の推定では付近に出現する Task の種類や作

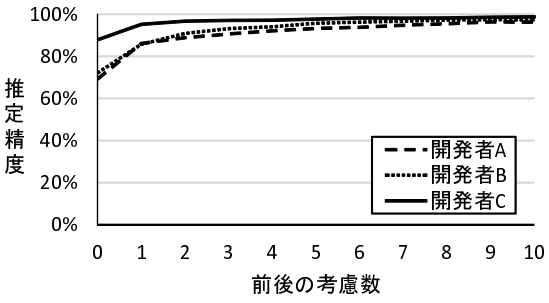


図 6　前後の考慮数と推定精度

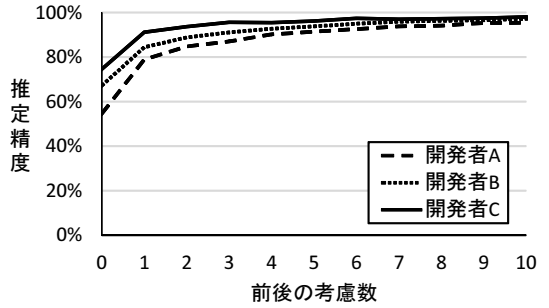


図 7　複数の Aim に用いられる Task に限定した推定精度

業量を用いることで，高い推定精度になったと考えられる．この結果は，前後の Task 情報を推定に用いることで推定の精度が向上する可能性を示唆している．

5.2　Aim 毎の精度

図 9 に開発者 A の各 Aim への推定精度を示す．図から休憩を除いた他の 3 つの Aim に対する推定精度において，前後の Task を考慮すると大幅に向上していることが分かる．特に考慮数 0 と 1 を比べた時，その他では 35.6%から 92.7%と 57.1%の向上，実装では 58.4%から 76.2%と 17.8%の向上が見られた． またテストと実装，その他では考慮数が 10 の時，それぞれ 90%を超える高い精度が確認できた．これらの Aim は，連続する Task 群の中に Aim の主要な Task があるため，高い精度で推定できたと考えられる．Aim における主要な Task による精度の向上については，5.5 節で詳述する．

	Task Name,	Lclick,	Rclick,	Keystroke,	Aim(正解),	前後無し,	前後有り
1:	ファイル操作,	1,	0,	0,	実装,	none,	none
2:	プログラミング・デバッグ,	30,	0,	78,	実装,	実装,	実装
3:	調査,	15,	0,	0,	実装,	テスト,	実装
4:	プログラミング・デバッグ,	47,	0,	13,	実装,	none,	none
5:	調査,	7,	0,	0,	実装,	テスト,	実装

図 8 実装における作業履歴（作業者 A）

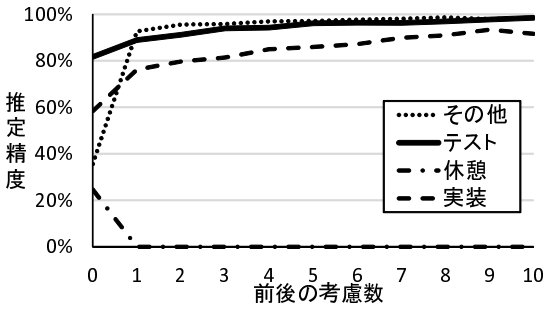


図 9 Aim 毎の推定精度（開発者 A）

一方で，休憩に関しては考慮数を増加させても精度の向上を見ることはできなかった．これは，休憩が 1 つの Task“ブラウザ”のみで構成されているため，前後の情報を考慮することで，前後の Aim である実装やテストに誤判定されるためと考えられる．以上のことから，提案手法では，少数の Task で構成される Aim に対しての推定精度が低下する可能性がある．

5.3 Aim を構成する Task 列と切り替わり

プロセス改善を適切に行うためには作業時間の長い Task に対して正しく Aim を推定することが重要である．本節では，実験で得られた作業履歴から Task の長さの分布と長さごとの推定精度について考察する．図 10 に開発者 A の Task における長さの分布と，長さごとの推定精度を示す．図の横軸は Task の長さ（分）を示し，棒グラフが各 Task 長に該当する Task の件数，折れ線グラフが長さごとの推定精度を表している．推定の対象となったテストデータ 928 件のうち，836 件 (90.1%) が 1 分以内の短い Task であり，5 分を超える Task は 11 件 (2.3%) しか存在しなかった．これは開発者が頻繁に（平均 36.4 秒）使用するアプリケーションを切り替えていることを示している．

Task の長さごとの推定精度に着目すると，件数が 3 件以上ある 7 分未満の Task において，Task の長さによる精度の違いは見られず，前後の Task 無しの場合と有り（前後 10 個）の場合の双方で安定していることがわかる．Task に費やした時間全体における正しく推定できた割合を見ると，全時間（9.39 時間）に対して，前後情報無しでは 4.91 時間 (52.3%) に相当する Task の Aim を正しく推定しているのに対して，前後情報有りでは 7.74 時間 (82.5%) に相当する Task を正しく推定している．これは，提案手法が作業件数のみならず作業時間に対しても高い割合で Aim を正しく推定できることを示しており，PSP の支援を目的とした手法としての有用性があることを示唆している．

短時間に頻繁に切り替わる Task に対して，どのタイミングで Aim が切り替わるのか正しく推定することは，正確な作業時間把握のために重要である．図 11 に開発者 A の Aim（正解）と，前後情報無しの Aim 推定結果，前後情報有り（前後 10 個）の Aim 推定結果の時系列変化を示す．図の横軸は Task を表し，横軸は Aim を表す．図 11 a) から開発者 A は Aim を頻繁には切り替えておらず，各 Aim に対して集中的に取り組んでいたことがわかる．

図 11 b) の前後情報無しの推定結果に着目すると，頻繁に Aim の切り替えが起きているように推定されている．複数の Aim を取りうる Task に対して，推定対象の Task 情報のみでは適切な Aim を推定できず，誤推定が頻繁に起きていると考えられる．一方で，図 11 c) の前後有りの推定では Task が切り替わっていても Aim の切り替えが起きていないことを高い精度で判断できている．これは各 Aim の固有の Task や

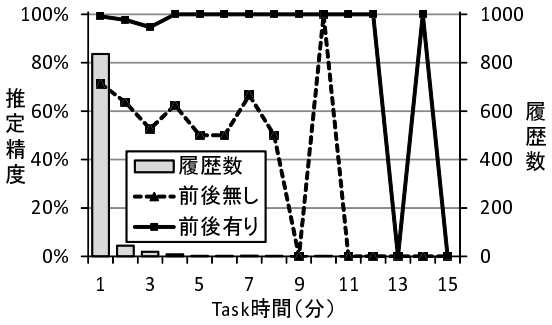
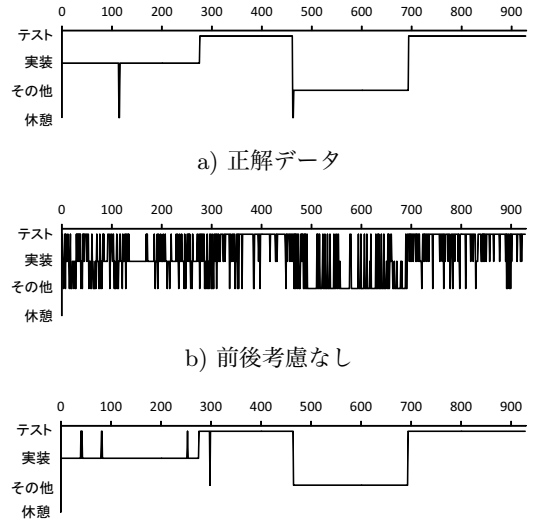


図 10　Task 長の分布と推定精度（開発者 A）



a) 正解データ

b) 前後考慮なし

c) 前後考慮あり

図 11　Aim の切り替わり

主要な Task が Aim 推定対象 Task の前後に存在するかどうかを判別し，Aim が切り替わっていないことを判断できるためと考えられる．以上の結果から，提案手法は Aim の切り替えが頻繁に起こらない作業履歴に対して，Aim を高い精度で推定できるといえる．頻繁に Aim が切り替わるような環境における提案手法の有用性の評価は今後の課題である．

5.4　特徴の重要度

Random Forests では，モデル構築時に使用した OOB データを用いて，使用している変数 (特徴) の重要度を推定できる．具体的には，重要度を推定したい特徴の値をランダムに変更することで，次元数を

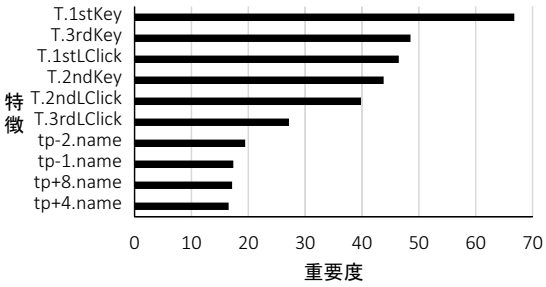


図 12　特徴の重要度

保ったまま目的変数 (Aim) との関係性を無くし，どの程度 OOB 推定値が上昇したかで重要度を決定する．図 12 に開発者 A の前後の Task を 10 個考慮した場合の重要度上位 10 件の特徴を示す．縦軸が特徴名，横軸が重要度を示す．重要度の値が高ければ Aim の推定に役立つ特徴である．図から，前後に存在する Task の名前よりも，T.1stKey や T.1stLclick などの前後の中で作業量が多い Task の名前の方が，重要度が高いことが分かる．これは，1 つの Task は複数の Aim で表れるため，Task 名のみでは Aim の識別が困難であるのに対して，作業量が多い (主要な)Task が Aim の識別に有用だったためであると考えられる．この結果は他の考慮数においても同じ傾向が見られたため，提案手法による推定精度の向上は，これらの特徴を用いて Aim 毎の主要な Task を判別できていることを示唆している．

5.5　Aim 毎の主要な Task

5.3 節にて示した特徴のうち，重要度が高かった T.1stKey，T.3rdKey，T.1stLClick の 3 つについて，Aim 毎に異なる値を持つか調査した．表 2 に開発者 A のデータに対して，それぞれの特徴に見られた Task 名上位 3 件を示す．なお，“休憩”については Task 件数が 12 件しかなかったため，表から省いてある．表からそれぞれの Aim において主要な Task に違いがあることが分かる． 実装では “プログラミング・デバッグ”が打鍵数，クリック数において最も多く，次に調査が多い．テストでは “Excel”や “調査”，その他では “Excel”や “OOo(soffice.exe)”，“文書閲覧”の作業量が多い．

表 2 3 つの特徴における Aim 毎の上位 3Task

Aim	T.1stKey		T.3rdKey		T.1stLClick	
	Name	#	Name	#	Name	#
実装	プログラミング・デバッグ	493	プログラミング・デバッグ	449	プログラミング・デバッグ	595
	調査	467	調査	398	調査	361
	ブラウザ	99	none	129	ブラウザ	116
テスト	Excel	731	調査	667	Excel	774
	調査	648	Excel	435	テスト (Task)	485
	テスト (Task)	324	テスト (Task)	398	調査	438
その他	Excel	319	Excel	177	Excel	233
	OOo(soffice.exe)	34	none	79	文書閲覧	110
	辞書閲覧	32	文書閲覧	52	ファイル操作	41

実装とテストの両方で行われた Task として “プログラミング・デバッグ”がある．図 13 にテストを Aim とする作業履歴の一部を示す．図 8 と比較すると，どちらにも “プログラミング・デバッグ”が記録されているが，実装（図 8）では左クリック数，打鍵数共に多く，主要な Task となっている．一方で，テスト（図 13）の “プログラミング・デバッグ”は左クリック数，打鍵数共に少なく，主要な Task とはなっていない．これらの特徴を用いない前後無しの推定では，付近にある Task（図 8 の 3,5 行目）の Aim が正しく推定できておらず，前後有りでは正しく推定できている．提案手法は前後 Task の特徴を推定に用いることで Task をそれぞれの Aim を構成する Task 群の一部としてとらえることができ，Aim における主要な Task の特徴を用いることで推定精度が向上したと考えられる．

5.6 Aim 推定による効果

開発者 A が実験期間中に行なった Task と Aim の割合をそれぞれ図 14，15 に示す．図 14 では，Task の種類が 20 と多く，また，調査，ファイル操作，エクセル，ブラウザのような汎用的な Task を含むため，どのソフトウェア開発作業にどの程度の時間を費やしていたかを把握することが困難であった．一方，図 15 では，ソフトウェアの実装とテストにそれぞれどの程度の時間を費やしていたかを把握することが可能となっており，実装やテストに十分な時間を投入できていない，もしくは，想定以上に時間がかかっているといった状況を把握できると期待される．

また，個々の Task の Aim を推定することで，具体的なアプリケーションの実行履歴からプロセス改善への糸口を見つけられると考えられる．例えば，今回の実験で計測された履歴には，実装時に “プログラミング・デバッグ”と “ファイル操作”を短時間に頻繁に切り替える動きが見られた．この作業履歴からは，IDE で編集するファイルを開くために IDE のファイル一覧機能を使わずに explorer を用いてファイルを開いている，非効率な作業プロセスが行われていることが伺える．Aim に基づいた Task ごとの作業履歴の分析は，開発作業時の様子をより具体的に思い出し，詳細な分析に基づいたプロセス改善に有用であると考えられる． また，今回の事例では，休憩とその他に費やされた時間は大きくなかったが，仮にこれらの時間の占める割合が大きい場合には，本来のソフトウェア開発業務の阻害要因となっている可能性がある．具体的なアプリケーションの実行履歴を参照することで，その原因を明らかにし，対策を講じることが可能となると期待される．

6 おわりに

本稿では，コンピュータ上で行われる Task の Aim を推定する手法を提案した．提案手法は，PSP の支援システムである TaskPit によって記録された作業履歴から，推定対象の前後の Task 情報と機械学習アルゴリズムである Random Forests を用いて Aim を推定する．実験の結果は，推定対象である Task の前後に実行された Task の特徴が，推定精度を向上させる上で有用である事を示した．Task の考慮数を増加させると，95%以上の精度を得ることが確認できた．

	Task Name,	Lclick,	Rclick,	Keystroke,	Aim(正解),	前後無し,	前後有り
1:	プログラミング・デバッグ,	1,	0,	0,	テスト,	none,	none
2:	テスト (Task),	16,	0,	4,	テスト,	テスト,	テスト
3:	プログラミング・デバッグ,	6,	0,	0,	テスト,	実装,	テスト
4:	テスト (Task),	45,	0,	7,	テスト,	none,	none
5:	調査,	36,	0,	66,	テスト,	テスト,	テスト

図 13　テストにおける作業履歴（作業者 A）

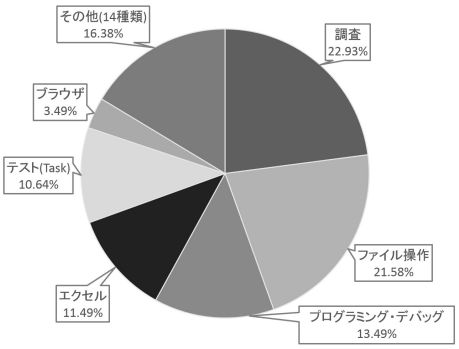


図 14　Task の割合

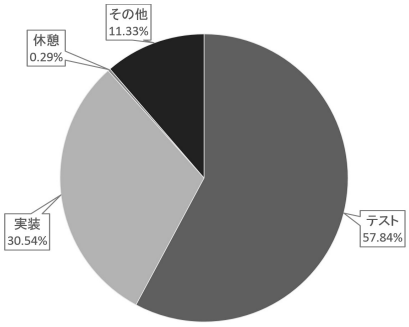


図 15　Aim の割合

特徴の重要度を調査した結果は，連続した Task の中で活発に行われている Task の識別が推定精度の改善に役立つ事を示唆している．実験期間中に記録された Aim のうち，Task が多かったものに対しての推定精度が高く，一部ではほぼ 100%の推定ができていた．加えて，Task 数のみならず Task に要した時間で比べても，Task の前後情報を用いることで Aim の推定精度が向上することを示した． 提案手法を用いることで従来よりも Aim の入力数を少なくし，ソフトウェア開発のプロセス計測を容易にできると考えられる．

今後の展望としては，Aim が頻繁に切り替わるような環境における提案手法の有用性の確認や他のソフトウェア開発組織の開発者による作業履歴を用いた実験，Random Forests のパラメータを変更することによる推定精度の変化を確認することなどが挙げられる．前の Task 情報のみを用いたリアルタイム推定のシステム提案も含まれる．

謝辞 本研究の一部は, 独立行政法人情報処理推進機構　技術本部　ソフトウェア高信頼化センター（SEC: Software Reliability Enhancement Center）が実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の支援を受けたものです．

参考文献

[1] T .DeMarco, “Controlling Software Projects: Management, Measurement & Estimation,” Yourdon Oress, New York, USA, 1982.

[2] W. S. Humphrey, “パーソナルソフトウェアプロセス入門,” 共立出版.

[3] 門田 暁人, 亀井 靖高, 上野 秀剛, 松本 健一, “プロセス改善のためのソフトウェア開発タスク計測システム,” ソフトウェア工学の基礎 XV, 日本ソフトウェア科学会 FOSE2008, pp.123-128, November 2008.

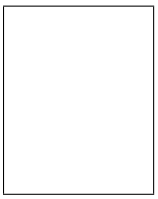
[4] M. Granitzer, A. S. Rath, M. Kroll, C. Seifert, D. Ipsmiller, D. Devaurs, N. Weber, and S. N. Lindstaedt, ”Machine Learning based Work Task Classification,” Journal of Digital Information Management, 2009, pp.306–313.

[5] S. Stumpf, X. Bao, A. Dragunov, T. G. Dietterich, J. Herlocker, K. Johnsrude, L. Li, and J. Shen, “Predicting User Tasks: I Know What You’re Doing!,” In Proc, the 20th National Conference on Artificial Intelligence (AAAI), 2005.

[6] N. Oliver, G. Smith, C. Thakkar, and A. C. Surendran, “SWISH: Semantic Analysis of Window Titles and Switching History,” In Proc. the 11th international conference on Intelligent user interfaces, 2006, pp.194–201.

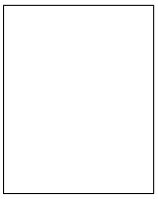
[7] M. Bernstein, J. Shrager, and T. Winograd, “Taskpose: Exploring Fluid Boundaries in an Associative Window Visualization,” In Proc. the 21st Annual ACM Symposium on User Interface Software and Technology (UIST), 2008, pp.231–234.

[8] L. Breiman, “Random Forests,” Journal of Machine Learning, Vol.45, No.1, 2001, pp.5–32.



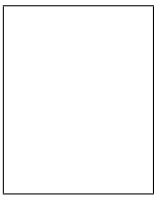
大橋亮太

2014 年奈良工業高等専門学校電子情報工学専攻修了．現在，奈良先端科学技術大学院大学情報科学研究科博士前期課程に在学中．ソフトウェア開発者のプロセス改善に関する研究に従事．IEEE 学生会員．



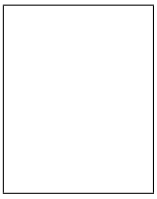
上野秀剛

2004 年岩手県立大学ソフトウェア情報学部卒業．2009 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年奈良工業高等専門学校情報工学助教．2015 年より講師．博士 (工学)．ソフトウェア開発におけるヒューマンファクタおよび，ユーザビリティの研究に従事．日本ソフトウェア科学会，ヒューマンインタフェース学会，電子情報通信学会，IEEE，ACM 各会員．



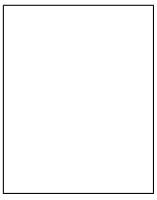
門田暁人

1994 年名古屋大学工学部電気学科卒業．1989 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年同大学同研究科助手．2004 年同大学助教授．2007 年同大学准教授．2003∼2004 年 Auckland 大学客員研究員．2015 年岡山大学大学院自然科学研究科教授．博士 (工学)．ソフトウェアメトリクス，ソフトウェアプロテクション，ヒューマンファクタなどの研究に従事．日本ソフトウェア科学会，情報処理学会，電子情報通信学会，IEEE，ACM 各会員．



荒木健史

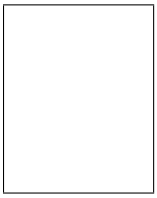
1993 年株式会社エルネット入社．3 年後株式会社アクセス転籍．複数の大規模プロジェクトに携わり，TL，PL を歴任．また，CASE ツール，リバースエンジニアリングツールの利用，研究，構築，導入に従事．2011 年オフショア開発推進 GL に就任．子会社の阿克塞斯軟件 (上海) 有限公司出向．日本側との調整，BSE マネジメントに従事．2014 年 NCS 株式会社と合併，社名が NCS&A 株式会社となる．



山田欣吾

1988 年関西大学工学部管理工学科卒業．同年日商エレクトロニクス入社，システムの分析，開発，コンサルティングを経験．1994 年アクセス入社．CASE ツール、リバースエンジニアリングツール (REVERSE PLANET) の研究、企画、開発、実プロジェクトへの適用に従事，2002 年リバースエンジニアリング装置で特許取得．2006 年同社取締役．2012 年同社常務取締役．2013 年同社子会社のファイ

ンバス副社長．2014 年ファインバス代表取締役社長 (現任)



松 本 健 一

1985 年大阪大学基礎工学部情報工学科卒業．1989 年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．1993 年奈良先端科学技術大学院大学助教授．2001 年同大学教授．工学博士．ソフトウェア開発データの収集技術，ソフトウェア開発プロセスの評価技術，定量的プロジェクト管理技術などの研究に従事．日本ソフトウェア科学会，プロジェクトマネジメント学会，ACM 各会員．電子情報通信学会フェロー，情報処理学会フェロー，IEEE Senior Member．