情報処理学会論文誌 Vol.58 No.4 855–860 (Apr. 2017)

テクニカルノート

ソフトウェア開発プロジェクトの生産性分析に対する傾向スコアの適用

角田 雅照[1,a)] 天嵜 聡介[2,b)]

受付日 2016年8月10日, 採録日 2017年1月10日

概要：定量的なプロジェクト管理を支援するために，ソフトウェア開発プロジェクトにおいて収集されたデータを分析し，知見を得ることが行われる．ただし，ソフトウェア開発において収集されるデータでは，項目間に相互に関連が見られる場合があり，これにより誤った結論を導いてしまう場合がある．そこで本稿では，相互関連を考慮する方法として，傾向スコア（propensity score）に着目する．傾向スコアは，着目する説明変数以外の説明変数，すなわち共変量を傾向スコアと呼ばれる 1 変数に集約して扱いを容易にしたものである．実験では傾向スコアに基づくマッチングを用いてソフトウェア開発の生産性に影響する要因を分析した．その結果，重回帰分析の分析結果と異なり，COBOL が多く使われるような条件（保険業の業種）では，その他の言語よりも COBOL 使用プロジェクトのほうが，生産性が高い傾向が見られた．

キーワード：生産性要因，交絡，共変量，PSM

Applying Propensity Score to Productivity Analysis of Software Development Project

Masateru Tsunoda[1,a)] Sousuke Amasaki[2,b)]

Received: August 10, 2016, Accepted: January 10, 2017

Abstract: To support quantitative project management, datasets collected in software development projects are often analyzed, and useful knowledges are acquired. However, explanatory variables often relate to other explanatory variables on the software development datasets, and it sometimes causes spurious relationship. To consider the relationships, we focus on the propensity score. The propensity score compresses other explanatory variables, i.e., covariates to one variable (propensity score). To consider the covariates, we only handle the propensity score. In the experiment, we used the matching based on the propensity score, to analyze productivity factors of software development. As a result, using COBOL improves productivity on the situation where it is often used. The result was different from the multiple regression analysis.

Keywords: productivity factor, confounding, covariate, PSM

1. はじめに

大規模なソフトウェアの開発において，品質の低下，納期の遅延，コストの超過などの失敗を防ぐためには，定量的なデータから得られる知見に基づいてプロジェクトを管理することが重要となる．具体的には，ソフトウェア開発プロジェクトにおいて収集された定量的なデータを分析し，その結果をプロジェクト計画やプロセス改善に役立てることでプロジェクト管理を支援する．生産性に影響する要因の分析 [8] はその一例である．

ソフトウェア開発において収集されるデータでは，項目間に相互に関連が見られる場合があり，適切な処置がなされないと誤った結論を導く恐れがある．たとえば，あるツールを導入することにより，生産性向上の効果があるか

[1] 近畿大学
Kindai University, Higashiosaka, Osaka 577–8502, Japan
[2] 岡山県立大学
Okayama Prefectural University, Soja, Okayama 719–1197, Japan
[a)] tsunoda@info.kindai.ac.jp
[b)] amasaki@cse.oka-pu.ac.jp

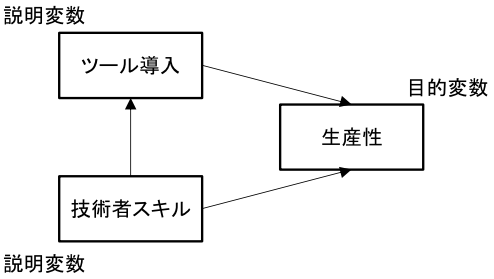

図 1 ソフトウェア開発データにおける交絡の例
Fig. 1 An example of confounding on the software development dataset.

どうかを明らかにしたいとする．データを分析した結果，ツールの導入有無により生産性に差異があったとしても，必ずしもツールの導入に効果があったと結論付けることはできない．ツールを導入している企業の技術者のスキルが高い傾向にあるとすると，生産性に差異を生じさせている主な要因はツールではなく技術者のスキルである可能性がある．このような，原因と結果の両方の変数に影響している変数が存在することを交絡と呼ぶ．**図 1** に交絡の例を示す．

ツールの導入と技術者のスキルに関連がなければ，ツールの導入と技術者のスキルを説明変数，生産性を目的変数として重回帰分析を行い，ツールの導入効果を確かめることができる．また，ツールの生産性に対する効果を明らかにする必要がなく，生産性を予測することが目的であれば，交絡が存在する場合でも重回帰分析などが適用可能である．しかし，上記の例のように，ツールが使われている企業では技術者のスキルが高いといった交絡が存在する場合，重回帰分析ではツール単独の効果を確かめることはできない．

そこで本稿では，ソフトウェア開発において収集される項目に相互に関連が見られる場合に，その関連を考慮する方法として，傾向スコア（propensity score）[10] に着目し有用性を評価した．傾向スコアは，着目する説明変数（上記の例ではツール導入）以外の説明変数，すなわち共変量を傾向スコアと呼ばれる 1 変数に集約して扱いを容易にしたものである．これにより，分析結果に対する交絡の影響を抑えることができる．

傾向スコアの有用性を確かめるために，傾向スコアに基づくマッチングを用いてソフトウェア開発の生産性に影響する要因を分析した．その結果を，従来用いられている重回帰分析の結果と比較することで，ソフトウェア開発データの分析において傾向スコアの利用を考慮すべきかについて評価を行った．

2. 傾向スコア

2.1 概要

説明変数の間に関連があり，その影響を除外する場合，これまでデータの層別や，SEM（Structural Equation Modeling）などの共分散分析などが利用されてきた．ただし，説明変数が多数存在する場合，層別の適用が困難となる．たとえば，説明変数として開発言語，業種，開発種別，アーキテクチャが存在する場合，それらをすべて考慮して層別すると，各サブセットのデータ数が非常に少なくなる．共分散分析の場合，目的変数と共変量の関係をモデル化する必要があるが，このモデルが正しくないと誤った分析結果になるという問題がある [4]．

原因と結果の両方の変数に影響している変数が存在する場合，それを考慮して分析する方法として，傾向スコア（propensity score）を用いた分析方法が提案されている [10]．傾向スコアを用いた分析では，前章で述べたようなデータの偏りが存在する場合でも，ある説明変数（二値をとる変数）の目的変数に対する影響を，その他の説明変数の影響を除外して明らかにすることができる．傾向スコアは着目する説明変数以外の説明変数，すなわち共変量を 1 変数に集約したものであり，これにより共変量の扱いが容易となる．

傾向スコアを算出するには，着目する説明変数を目的変数とし，その他の説明変数（共変量）を説明変数としたロジスティック回帰分析を行い，構築されたモデルを用いて傾向スコアを算出する．たとえば，ツールの利用有無が生産性に与える影響を知りたいとする．この場合，ツールの利用有無を目的変数，技術者のスキル，開発ソフトウェアの複雑度などの共変量を説明変数とするモデルを作る．傾向スコアとは，ロジスティック回帰分析による判別分析の予測値（確率で表される）であり，この例ではツール利用有となる確率が回帰モデルにより算出される．

傾向スコアを算出した後は，マッチング，層別，共分散分析のいずれかを行う [10]．本稿ではマッチングについて述べる．マッチングとは，傾向スコアが近く，かつ着目する説明変数の値（この例ではツールの利用有無）が異なるケース（データ行）どうしをペアにして分析することである．傾向スコアが類似している場合，スコアの算出に用いた説明変数，すなわち技術者のスキルや開発ソフトウェアの複雑度などが同程度であり，これらを無視することができる．ペア間の目的変数の差（この例では生産性）が，着目する説明変数（ツールの利用有無）の効果（効果量）となる．効果量は正規化されておらず，値域は限定されない．

図 2 に傾向スコアによるマッチングの例を示す．傾向スコアが同じ場合，共変量である COBOL，メインフレーム，FP などが 2 つのケースどうしで近い値となる．このため，生産性とツールの利用有無との関係を，共変量の影響を除外して分析できる．

傾向スコアを算出するモデルが適切かどうかは，c 統計量（ROC 曲線下面積．Area Under the Curve）を用いたり，共変量の平均値の差がマッチングにより小さくなって

いるかどうかなどによって判断する．傾向スコアの計算対象となる変数は二値をとる変数であるが，着目する変数が 3 つ以上のカテゴリを含む場合，一般化傾向スコア（generalized propensity score）を用いたり，「あるカテゴリかそれ以外のカテゴリか」を表す変数を作成したりして分析を行う [4]．

2.2 定量的分析における活用

ソフトウェア工学のデータでは，説明変数間に関連があることが多い．たとえば，金融業向けのソフトウェアはメインフレームで開発することが多い．従来法では金融業，メインフレームそれぞれの生産性への影響を個別に明らかにすることは難しい．傾向スコアを適用することにより，個別の影響を明らかにすることができると期待される．ソフトウェア工学分野における傾向スコアの適用は，著者らの知る限り文献 [9] などのごく一部であるが，データに偏りがあることが多いため，傾向スコアの適用が向いていると考えられる．

傾向スコアの一般的な使い方は，ある説明変数の効果を分析することであるが，それ以外にインターネット調査のデータ補正 [3] に用いられることがある．無作為標本抽出による調査と異なり，インターネット調査は偏りがある．その偏りを除去するために傾向スコアを利用している．まず，Web 調査と無作為標本抽出による調査の両方からデータを収集する．このとき，一般的なアンケート項目とともに，性別や都市規模，学歴などを収集する．次に，「Web 調査かそうでないか」という変数を作成し，性別や都市規模，学歴からその変数を予測するモデルを作成し，予測結果を傾向スコアとする．最後に，算出された傾向スコアを重みとして，インターネット調査のアンケート項目の回答の補正を行う．

我々の知る限り，ソフトウェア工学分野において，傾向スコアによるインターネット調査の補正を行った研究は存在しない．ソフトウェア工学においても，インターネット調査が活用されつつあるが [5]，同様のデータ補正を行うことにより，研究の信頼性がさらに高まる可能性がある．

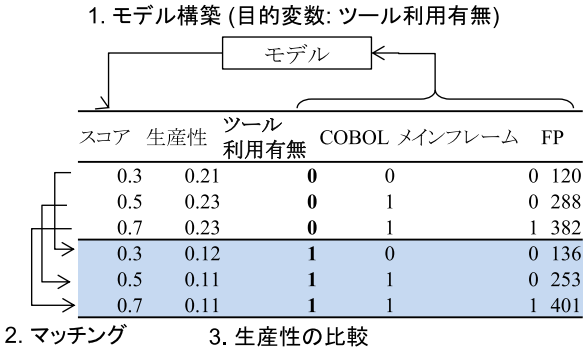

図 2 傾向スコアによるマッチングの例
Fig. 2 An example of propensity score matching.

3. 実験

本章では，ソフトウェア開発データの分析における傾向スコアの有用性を実証的に示す．そのために，傾向スコアによるマッチングに基づき，ソフトウェア開発の生産性に影響する要因（業種，開発言語など）を分析する．その結果を従来の重回帰分析による結果と比較し，分析結果に違いが生じるか，異なるのはどの要因かを明らかにする．

3.1 データセット

分析では，ISBSG（International Software Benchmark Standard Group）が収集および提供しているソフトウェア開発プロジェクトのデータセットを用いた．ISBSG は 20 カ国のソフトウェア開発企業から 3026 件のプロジェクトのデータを収集しており，各プロジェクトについて 99 種類の変数が記録されている（リリース 9 の場合）．複数の企業から収集されているため，ISBSG のデータセットは企業横断的データセットと呼ばれることもある．

データセットに含まれる信頼性の低いデータを除外するため，また，分析対象のプロジェクトの条件を整えるために，以下の条件にあてはまるデータを抽出した．

- データ品質評価が A または B
- FP（Function Point）計測の評価が A または B
- FP 計測方法が IFPUG
- 工数の計測対象が開発チームのみとしている（間接部門の工数などを含んでいない）

上記条件は，ISBSG データを用いて工数見積りの研究行っている Lokan ら [6] のデータ抽出条件を参考にした．さらに欠損値が含まれるケースは除外した．上記の条件に従ってデータを抽出した結果，分析対象のデータ件数は 610 件となった．

各変数の定義を**表 1** に示す．分析では新たに生産性を定義した．また，全体のデータ件数の 5%（31 件）を下回る

表 1 分析に用いた変数
Table 1 Variables used in the analysis.

変数	詳細
開発種別	新規開発/再開発，保守開発
FP	開発総規模．未調整 FP
工数	開発工数（正規化済み）
工期	休止期間を除く開発工期
業種	金融業，情報通信業など
プラットフォーム	メインフレーム，PC など
開発言語	C/C++/C#，Visual Basic など
EI	外部データ入力機能数
EO	外部データ出力機能数
EQ	外部データ参照機能数
ILF	更新ファイル数
EIF	参照ファイル数
追加 FP	追加した機能の FP
変更 FP	変更した機能の FP
削除 FP	削除した機能の FP
生産性	FP÷工数

表 2 各説明変数と生産性との相関比
Table 2 Correlation ratio between productivity and explanatory variables.

業種	プラットフォーム	開発言語
0.46	0.24	0.37

表 3 説明変数間のクラメールの V
Table 3 Cramer's V between explanatory variables.

	プラットフォーム	開発言語
業種	0.51	0.44
プラットフォーム	-	0.67

表 4 業種のカテゴリ別生産性
Table 4 Productivity stratified by business sector.

	ケース数	平均値	中央値
保険業	185	**0.11**	**0.06**
情報通信業	127	0.18	0.09
金融業	100	0.13	0.08
製造業	41	0.27	0.20

表 5 プラットフォームのカテゴリ別生産性
Table 5 Productivity stratified by hardware platform.

	ケース数	平均値	中央値
メインフレーム	259	**0.14**	**0.08**
ミッドレンジ	107	0.16	0.09
マルチ	132	0.19	0.13
PC	112	0.27	0.15

件数のカテゴリについては，分析から除外した．比例尺度については，平均値などの基本統計量の算出以外では，外れ値の影響を避けるために対数変換を行った．

分析では開発者のスキルに関する変数を含んでいない．この変数は COCOMO では含まれており，無視すべきでない要因の 1 つであるが，ISBSG データに記録されていないため分析に含めていない．生産性要因に関する分析で，開発者スキルを対象に含めていない研究は比較的多く見られる [7], [11].

3.2 予備分析

生産性は業種やプログラミング言語との関連が強いことが多い [7], [8]．また，それらの変数に相互に関連が見られることがある [11]．そこで本稿でも業種，プラットフォーム，プログラミング言語と生産性との関係に着目する．これらの変数に関して，名義尺度と比例尺度との関連の強さを示す相関比を求めた．**表 2** に示すように，業種と生産性との関連が強く，プラットフォームとは若干関連が見られた．これらの変数に関して，名義尺度の変数間の関連の強さを表すクラメールの V を求めると，**表 3** に示すように，業種，プラットフォーム，プログラミング言語の相互に関連が見られた．

表 4，**表 5**，**表 6** に各変数のカテゴリ別の，生産性の平均値と中央値を示す．最も生産性が低い箇所を太字で

表 6 開発言語のカテゴリ別生産性
Table 6 Productivity stratified by programming language.

	ケース数	平均値	中央値
VISUAL BASIC	150	0.16	0.10
COBOL	116	**0.12**	**0.06**
PL/I	71	0.14	0.08
C/C++/C#	69	0.20	0.08
Java	44	0.14	0.11
NATURAL	44	0.25	0.15
SQL	42	0.17	0.09

表 7 傾向スコアに基づく生産性分析
Table 7 Productivity analysis based on propensity score.

	保険業	メインフレーム	COBOL
効果量	-0.92	-0.32	0.19
c 統計量	0.91	0.97	0.97

示す．業種の場合は保険業が最も生産性が低く，プラットフォームはメインフレームが低かった．開発言語については COBOL の生産性が最も低かった．そこで，「業種が保険業の場合とそれ以外の場合で，生産性に差があるのか」，「プラットフォームがメインフレームとそれ以外の場合で，生産性に差があるのか」，「開発言語が COBOL とそれ以外の場合で，生産性に差があるのか」のそれぞれについて，傾向スコアを用いて分析した．

3.3 傾向スコアによる分析

傾向スコアを算出するために，「業種が保険業かそれ以外か」，「プラットフォームがメインフレームかそれ以外か」，「開発言語が COBOL かそれ以外か」を示すダミー変数を作成し，そのダミー変数を目的変数としてロジスティック回帰分析を行った．ダミー変数は名義尺度の変数を重回帰分析などで扱えるように数値化したものであり，たとえば「COBOL であれば 1，そうでなければ 0」のように定義され，開発言語に複数の種類が含まれる場合，複数の（各言語を表す）ダミー変数を作成することが通常である．

説明変数は，たとえば目的変数が保険業を表す場合，表 1 に示す変数のうち，業種と生産性以外を用いた．共変量，すなわち表 1 に示す変数の値を構築されたモデルに説明変数として与え，算出された目的変数の予測値が傾向スコアとなる．

表 7 に示すように，どのモデルでも c 統計量が 0.9 を超えていたことから，適切にモデルが構築されているといえる．表 7 において効果量は，各カテゴリが（対数変換後の）生産性をどの程度変化させたかを示しており，負の値の場合は生産性を低下させたことを示す．3.2 節の予備分析の結果と異なり，COBOL の場合，生産性が高まるという分析結果となった．

3.4 回帰分析との比較

目的変数を生産性として，「業種が保険業かそれ以外か」，

表 8 回帰分析に基づく生産性分析
Table 8 Productivity analysis based on regression analysis.

	保険業	メインフレーム	COBOL
単回帰	-0.69	-0.38	-0.46
全ダミー変数	-0.91	-0.73	0.06
最小ダミー変数	-0.77	-0.04	-0.07

「プラットフォームがメインフレームかそれ以外か」,「開発言語が COBOL かそれ以外か」を示すダミー変数などを説明変数として単回帰分析および重回帰分析を行った．(1) 前述のダミー変数のみを用いた単回帰分析，(2) 表 1 に示す説明変数（工数を除く）をすべて用いた重回帰分析，(3) 業種，プラットフォーム，開発言語については前述のダミー変数のみを用い，その他は表 1 に示す説明変数（工数を除く）を用いた重回帰分析，の 3 パターンで分析した．

ここで，(3) のモデルはダミー変数による業種，プラットフォーム，開発言語の扱いを傾向スコアと同様にするために構築しており，たとえば開発言語については「開発言語が COBOL かそれ以外か」のみを用いている．これに対し (2) のモデルでは，傾向スコアでのダミー変数の扱いに揃えることはせず，たとえば開発言語については「開発言語が COBOL かそれ以外か」だけではなく「開発言語が Java かそれ以外か」なども用いている．

構築されたモデルにおける，これらのダミー変数の偏回帰係数を**表 8** に示す．モデルの偏回帰係数を見ると，保険業のダミー変数については構築されたモデルごとに大きく変化していなかったが，メインフレーム，COBOL のダミー変数では，モデルによって偏回帰係数が大きく異なっていた．表 7 の傾向スコアによるマッチングと比較すると，保険業のダミー変数については大きな差はなく，メインフレームは単回帰の場合と近かった．ただし，COBOL については，回帰分析では生産性に対して負の影響，もしくはほとんど影響しないという結果であり，傾向スコアによる分析結果（生産性に対する正の影響）と比較し，大きく異なっていた．

前節と本節の分析結果を以下にまとめる．

- 回帰分析と傾向スコアのどちらを分析手法として用いるかによって，変数によっては結果が異なる場合があった．
- 傾向スコアによる分析結果では，保険業の場合は生産性が低く，COBOL の場合は生産性が高い傾向があった．

3.5 考察

傾向スコアと重回帰分析の結果が異なる理由は，以下のようなことであると考えられる．たとえば COBOL は保険業を中心に使われており，保険業の場合は COBOL のほうが他の言語よりも相対的に生産性が高く（たとえば COBOL では 0.1，その他の言語では 0.05），かつその生産性が保険業以外での他の開発言語よりも低い（たとえば保険業以外でのその他の言語では 0.2）とする．この場合，傾向スコアと重回帰分析で異なる結果が観察される．

よって，COBOL を用いることが一般的な開発に対し，他の開発言語を用いると生産性が下がる可能があり，COBOL をあらゆる開発に用いると，逆に生産性が低下する可能性があると考えられる．すなわち，きわめて当然ではあるが，プロジェクトを計画する際には，適切な開発に対してのみ COBOL を用いるべきであるといえる．また，傾向スコアと重回帰分析で結果の異なる要因については，適用する場面について慎重に考慮するべきであるといえる．

3.6 傾向スコアの活用

従来手法による分析では「COBOL による開発は生産性が低い」という結果となっていた．文献 [2] においても，COBOL の生産性は比較的低い．これらより「ソフトウェア開発の生産性を高めるためには，開発言語として COBOL を使うべきでない」という結論が導かれていた．これに対し，傾向スコアを適用した分析では，「COBOL の生産性が低い理由は，利用されることが多い条件下の生産性が低いためであり，同様の条件下では，その他の言語よりも COBOL の生産性のほうが高い」という結果が示された．すなわち，「COBOL が多く使われるような条件（保険業の業種）では，その他の言語よりも COBOL 使用プロジェクトのほうが，生産性が高まる」という，従来と異なる結論が導かれた．

今回分析対象としなかった変数について，傾向スコアを用いて分析することにより，従来と異なる結果がさらに得られる可能性がある．COBOL の分析結果のように，従来の分析結果と異なる結果が得られた場合，ソフトウェア開発プロジェクトの計画立案のセオリが変わる可能性がある．

4. おわりに

ソフトウェア開発において収集されるデータでは，項目間に相互に関連が見られる場合があり，これにより誤った結論を導いてしまう可能性がある．この問題を回避するために，本稿では傾向スコアに基づく分析方法の適用を試みた．傾向スコアは着目する説明変数以外の説明変数（共変量）を 1 変数に集約したものであり，これにより共変量の影響を低減しやすくなる．

分析では ISBSG データセットを用いて，業種，プラットフォーム，プログラミング言語と生産性との関係を分析した．その結果，業種（保険業）の生産性に対する影響については，重回帰分析と傾向スコアの分析結果が類似していたが，開発言語（COBOL）については，両者で分析結果が大きく異なっていた．すなわち，従来手法の分析では「生産性を高めるためには COBOL を使うべきでない」という結論が導かれていたが，傾向スコアの分析では「COBOL

が多く使われるような条件（保険業の業種）では，その他の言語よりもCOBOL使用プロジェクトのほうが，生産性が高まる」という結論が導かれた．このことから，傾向スコアはソフトウェア開発データの分析に有用であり，今後活用すべきであるといえる．

謝辞 本研究の一部は，文部科学省科学研究補助費（基盤C：課題番号16K00113，若手B：課題番号15K15975）による助成を受けた．

参考文献

[1] Boehm, B.: *Software Engineering Economics*, Prentice Hall (1981).
[2] Jones, C.: *Programming Languages Table* (1996), available from ⟨http://www.cs.bsu.edu/homepages/dmz/cs697/langtbl.htm⟩.
[3] 星野崇宏，前田忠彦：傾向スコアを用いた補正法の有意抽出による標本調査への応用と共変量の選択法の提案，統計数理，Vol.54, No.1, pp.191–206 (2006).
[4] 星野崇宏，岡田謙介：傾向スコアを用いた共変量調整による因果効果の推定と臨床医学・疫学・薬学・公衆衛生分野での応用について，保健医療科学，Vol.55, No.3, pp.230–243 (2006).
[5] 河村智行，高野研一：情報システム開発の成否に影響を与える組織文化の要因の研究，情報処理学会論文誌，Vol.53, No.12, pp.2854–2864 (2012).
[6] Lokan, C. and Mendes, E.: Cross-company and single-company effort models using the ISBSG database: A further replicated study, *Proc. International Symposium on Empirical Software Engineering* (*ISESE*), pp.75–84 (2006).
[7] Lokan, C., Wright, T., Hill, P. and Stringer, M.: Organizational Benchmarking Using the ISBSG Data Repository, *IEEE Software*, Vol.18, No.5, pp.26–32 (2001).
[8] Maxwell, K., Wassenhove, L. and Dutta, S.: Software Development Productivity of European Space, Military, and Industrial Applications, *IEEE Trans. Softw. Eng.*, Vol.22, No.10, pp.706–718 (1996).
[9] Ramasubbu, N. and Balan, R.: The impact of process choice in high maturity environments: An empirical analysis, *Proc. International Conference on Software Engineering* (*ICSE*), pp.529–539 (2009).
[10] Rosenbaum, P. and Rubin, D.: The Central Role of the Propensity Score in Observational Studies for Causal Effects, *Biometrika*, Vol.70, No.1, pp.41–55 (1983).
[11] Tsunoda, M., Monden, A., Yadohisa, H., Kikuchi, N. and Matsumoto, K.: Software Development Productivity of Japanese Enterprise Applications, *Information Technology and Management*, Vol.10, No.4, pp.193–205 (2009).

角田 雅照 （正会員）

1997年和歌山大学経済学部卒業．2007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特任助教．2012年東洋大学総合情報学部助教．2013年近畿大学理工学部情報学科講師．博士（工学）．ソフトウェアメトリクスの研究に従事．電子情報通信学会，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，教育システム情報学会，IEEE各会員．

天嵜 聡介 （正会員）

2006年大阪大学大学院情報科学研究科博士後期課程修了．現在，岡山県立大学情報工学部助教．工数見積り手法，fault-proneモジュール判別手法等の研究に従事．博士（情報科学）．IEEE-CS，ACM各会員．