様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成２４年３月３１日現在

機関番号：13801
研究種目：若手研究（Ｂ）
研究期間：2009～2011
課題番号：21700276
研究課題名（和文）プログラミング教育における問題解決能力獲得プロセスの評価と支援の研究
研究課題名（英文） Scaffolding and Evaluation of Problem-Solving Process Acquisition in Programming Education
研究代表者
松澤 芳昭（YOSHIAKI MATSUZAWA）
静岡大学・　情報学部・助教
研究者番号：40517017

研究成果の概要（和文）：
本研究ではプログラミング教育における問題解決のプロセスに注目した教育支援プラットフォームを構築と評価を行った．主要な成果は，プログラミングプロセスの視覚化ツール Programming Process Visualizer の開発と教育現場でのプロセス測定と支援の実験，プログラミング導入教育におけるコンパイルエラー修正プロセスの分析フレームワークの提案，ブロックエディタ方式によるプログラミング構造化教育支援システムの開発と評価である．

研究成果の概要（英文）：
We have developed a platform to support introductory programming education by focusing on problem-solving processes in programming. The results are summarized by the following three achievements: 1) the development of a programming process visualization tool named Programming Process Visualizer and its application in an actual programming class, 2) the proposal of an analysis framework for compilation error correction time and its declining rate in novice programmers, 3) the development of a block editing system to support structured programming and seamless migration to Java.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
2009 年度	1,100,000	330,000	1,430,000
2010 年度	900,000	270,000	1,170,000
2011 年度	900,000	270,000	1,170,000
総　計	2,900,000	870,000	3,770,000

研究分野：総合領域
科研費の分科・細目：情報学・図書館情報学・人文社会情報学
キーワード：プログラミング教育，教育情報システム，学習支援，活動記録，プロセス改善

１．研究開始当初の背景

プログラミング教育はコンピュータを使いこなすための情報教育として行われてきたが，近年ではアプリケーションソフトウェアの普及に伴って，プログラミングそのものはコンピュータを使用するためには必須のものではなくなりつつある．しかしながら，プログラミング教育は，コーディング能力の育成ではなく，そのプロセスを通して論理的な問題解決能力を育成することを目的とする教育であり，この目的を主とし，情報教育の柱として発展しなければならない．

しかしながら，プログラミング教育は旧態依然の方法で行われている．プログラミング教育はアルゴリズム教育から切り離され，言語を習得することが目的と考えられている．

近年のプログラミング教育に対する研究は習得しやすい言語の開発（例えばドリトル，Squeak)やプログラムの視覚化などの開発環境による学習支援（例えばPEN）などのアプローチなどの代表的な研究がある．しかしながら，それらの研究では，プログラミングの問題の正答率のみしか評価されておらず，最終目的である論理的な問題解決能力を育成できたかに関しては評価されていない．

当該研究者は小学校～大学生，新入社員まで幅広い年齢層を対象に，プログラミング教育を行ってきた．ここでは，これからの情報教育にふさわしいプログラミング教育として,「課題を分析し，系統的に解決策を考え，コンピュータに実行可能な形で明示的に表現し，実行結果を検討し必要なら反復改良する」という情報システム学が掲げる総合的な問題解決プロセスを育成するカリキュラムを構築し，実践を行ってきた．しかしながら，実際には学習者が問題解決能力を獲得したかを評価するのは難しく，学習者の成果物から指導者がそのプロセスを汲み取って評価しているのが実情である．そこで，本研究においては，学習者のプロセスを追跡して，問題解決能力がどのように獲得されているか探る．

２．研究の目的

本研究は，問題解決能力の獲得プロセスに焦点をあて，それを学習者，指導者に認知的負担をかけずに，教育現場で実践できる方法の試作と支援のプラットフォームを提供すること，およびその実証試験によって，実際の学習者の問題解決能力獲得のプロセスを分析し，それらが改善できる枠組みを提供することを目的とする．本研究の成果は，プログラミング教育の学習者，指導者，研究者にとって有用なだけでなく，実プログラミングとプロセス観察による実際の技術者の診断プログラムにも応用可能することを目標とする．

３．研究の方法

本研究は，プログラミングのプロセスだけでなく，その問題解決能力構築のプロセスの追跡と評価・改善を目的とする．ここで，問題解決能力構築のプロセスとは，例えば，論理エラーをデバッグする方法や，理解していない関数をテストするプログラムを作る，小さな部品からテストして徐々に大きくしていくなどの，問題を解くためのメタなプロセスのことを指している．

問題の定義と見積もり，解法の設計，品質評価，生産性評価等ソフトウェア工学の基本的なプラクティス，および問題分析と効果検証等の情報システム学の理解と実践が評価の基準となるその方法として，データの分析・評価・改善の枠組みに関しては，PSP(Personal Software Process）を基礎理論として利用する．

教育現場でのPSP適用，例えばLisack, 2000やAbrahamsson, 2002などの試みにおいては，ほとんどの学生がPSPの利点を理解することに失敗し，時間がかかるだけという印象を与えてしまうということが報告されている．その結果，学生が効果に不満を持つことも報告されている．その要因の一つが作業記録のオーバヘッドである.Johnsonらは PSPの記録方式について，三世代にグループ化してその利点欠点をまとめている(Johnson, 2003)．先行研究でのツールは編集などの低レベルデータの収集にとどまっている．そのため本研究では，プロセス分析のために，記録の閲覧，作業項目の分割を支援し，作業項目レベルで情報を分析することのできるツールを開発する．

開発したツールのプログラミング現場での評価として，実際にPSPの実践を行い，そのオーバヘッド，および教育効果を測定する．

次に，開発したツールを教師がプログラミング教育の問題解決過程を分析することに利用できるようにする．この観点から学習記録を用い，修正時間とその変化に注目してコンパイルエラーを分析する方法を考案し，実際の記録データに適用しその傾向を明らかにする．学習記録から自動で修正時間を算出するために，複合エラーに対応する修正時間の定義を行う．

最後に，学生のための学習支援ツールの効果測定へのツールを利用を検討する．具体的には，OpenBlocksフレームワークを利用し，構造化技法をサポートし，Java言語と相互変換ができるプログラミング教育環境を開発する．開発したツールを利用することで，プロセスを含む学習支援ツールの効果測定ができることを示す．

４．研究成果

(1) Programming Process Visualizerの開発

本研究の目的は，プログラミングの教育現場で学生によるプロセス分析を可能にするプログラミングプロセス記録・観察ツールを開発することである．授業で実施可能なオーバヘッドで，かつ学生が受容できる方式であることが設計目標である．

自動作業記録を含む全体の設計は，開発環境にセンサを埋込み操作ログを獲得し，リポジトリに記録する第三世代のアーキテクチャを採用している．Hackystatに代表されるツールと異なるのは，リポジトリへの記録はサーバクライアントモデルではなく，開発環境が実行されるPC上の（ローカル）ファイルであることである．ローカルファイル方式の利点は，サーバの設置や登録設定が不要な

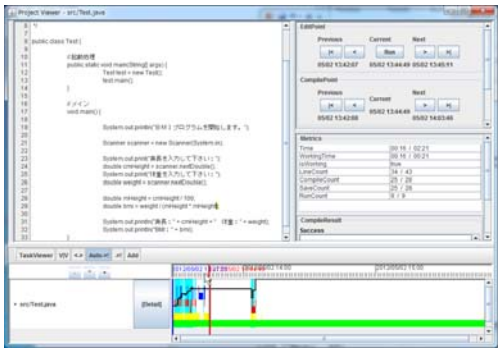

図１．プロジェクト・ビュー・ウインドウ

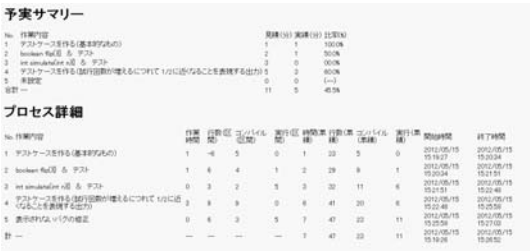

図２．レポート出力例

ことと，オフラインでの作業記録が容易である点で，欠点はリアルタイムな把握が出来なくなることであるが，我々の目的はプログラミング事後のプロセス観察であるので，この欠点は無視できる．

PPV の主要なウインドウである「プロジェクト・ビュー・ウインドウ」を図 1 に示す．「ソースコード提示区画」では，ソースコードの変遷を再現するアニメーションを提示する．アニメーションの時間軸は「指標提示区画」や「時間軸提示・操作区画」に配置されたボタンやバーで操作することができ，ユーザは任意の時点でのソースコードを閲覧することができる．「指標提示区画」には，その時点でのソースコードの指標が提示される．「時間軸提示・操作区画」は，色分けされた操作遍歴が時系列に配置される．

見積データの入力は，「作業項目（文字列）」と「見積もり時間（分）」の形式で行う．実績データの入力については，作業項目開始時点に青バーを設置し，作業項目の終了時点に赤バーを設置することで，作業範囲を設定する．作業範囲を設定して「追加」ボタンを押下することによって下部に示された作業項目入力画面が表示される．ここで対応する見積項目を選択することによって作業項目が自動入力され，実績データが追加される．

PPV は入力された作業項目データに基づいて，プロセス分析レポートを出力する．出力例を図２に示す．このレポートでは，見積/実績比率，作業項目毎の区間データ（行数やコンパイル回数），および最終成果物を総括

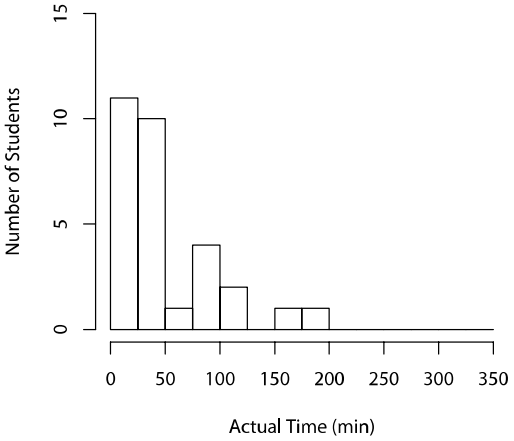

図３．実績時間測定結果（例）

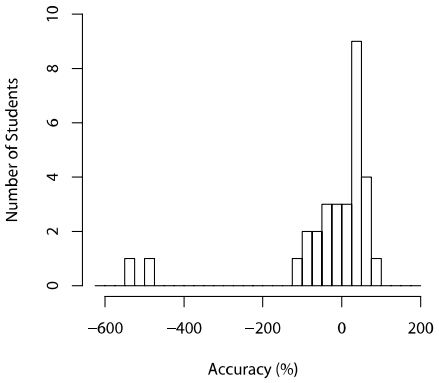

図４．見積精度測定結果（例）

して HTML(Hyper Text Markup Language)形式で出力する．ユーザはブラウザで閲覧することができ，Web に公開し共有することも可能である．

評価実験の対象科目は，プログラミング入門教育を修了した学生が履修できる，2 学期目のプログラミング必修科目である．対象科目での利用言語は Java である．作業見積もりは作業項目とそれに係る Effort(作業時間)のみについて実施した．

分析された PSP データの一部を図３，４に示す．図３はクラス全体の実績時間の分布，図４は同 Hayes，1997 の計算式による見積精度を示す．2 回の試行で見積精度改善の効果は見られなかったが，プログラミングプロセスの測定データが得られ，再帰プログラムの難易度が高いことが定量的データによって示された．

測定された作業項目の項目数の中央値は 6 要素付近あり，タスク間，見積/実績間の差は見られなかった．最大/最少や４分位の範囲は見積と比較して実績が広範囲になっていた．これは，タスクの区切り方に学生間のばらつきが大きいことを示しており，質的分析の結果ももそれを支持した．

分析時間の平均は 17.03 分，対象となるプ

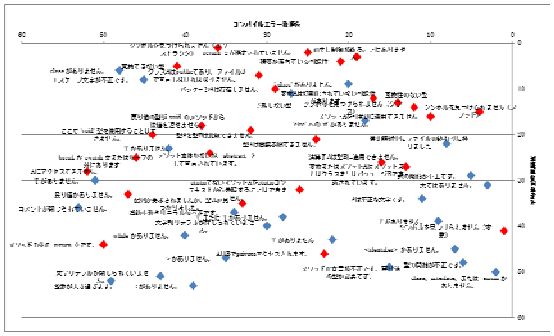

図５．エラー数順位と平均修正時間順位の相関

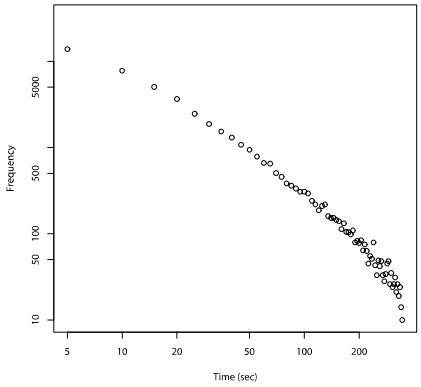

図６．エラー修正時間と頻度

ログラミング課題の平均は 49.43(分)であり，比率（分析時間/課題時間）は 35%となった．小さくないオーバーヘッドであるが，大学の授業で実現可能な範囲であった．

履修学生を対象とする質問紙調査の結果は，この試みが学生に受容可能であったかを問う質問について，分布の中心は「そう思う」であり，「そう思わない」以下の評価は 1 割程度であった．PSP を試行した先行研究における学生の評価が，非常に低評価であったのと比較して，良好なスコアと評価できる．提案ツールの使いやすさを問う質問に関しても，分布の中心は「そう思う」であった．分析過程の支援目的に一定の使用性が認められたと評価できる．

(2) プログラミング導入教育におけるコンパイルエラー修正プロセスの分析フレームワークの提案

本研究ではコンパイルエラー修正時間に着目し，プログラミング初学者のコンパイルエラー修正時間の増減速度を分析することでその傾向を明らかにすることを目的とした．増減速度を測定することで学習プロセスの傾向を明らかにした．

エラー数と修正時間の基礎データは，プログラミング学習者の学習記録を用い，その学習記録から自動で修正時間を算出するため

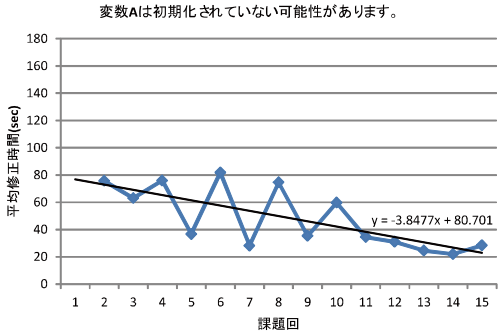

図７．修正時間の減少と速度計算の例

に，複合エラーに対応する修正時間の定義を行った．

エディタの操作履歴データ取得の対象は文科系の大学 1 年生約 100 名である．これらのデータは学生が半期間の Java プログラミング講義 15 回分で課された課題を解く過程で得られたものである．データ前処理後のコンパイルエラー総数は 67,910 個，コンパイルエラーの種類は 53 種類で，そのうち 27 種類が意味解析エラー，構文解析エラーが 26 種類であった，コンパイルエラー1 個の平均修正時間は 30.6 秒であった．

コンパイルエラー53 種に対しエラー数と平均修正時間の順位を算出し，エラー数の順位を横軸，平均修正時間の順位を縦軸にプロットしたものを図５に示す．図５では順位が高いほどエラー数が多い．本図よりエラー数と平均修正時間の相関がないことが分かる．例えば，学習者にとって修正が難しいエラーは右上に分布するエラーである．この位置に分布するエラーは学習者が目にする機会が多いにもかかわらず修正時間が長い．目にする機会が多いエラーであれば目にすることの少ないエラーよりも修正に慣れて修正時間が短くなると予想できるが，その予想に反している．他のエラーよりも修正が難しいことが予測され，教授者が注意して指導すべきことを示唆している．

エラー修正時間（横軸）と頻度（縦軸）を両対数グラフ上にプロットしたものを図６に示す．エラー修正時間はべき乗法則に従っていることが分かった．

次に，学習者のコンパイルエラー修正時間の変化を捉えるために，修正時間の増減速度を示す新しい指標 Correction Time Slope（以下，CTS）を考案した．エラーの平均修正時間について，各課題回毎の時系列データを求め，その線形近似式の傾きを CTS 値と定義した．CTS の算出例を図７に示す．グラフの縦軸にエラーの平均修正時間，横軸に課題回をとる．このグラフから線形近似式を求めて得られた傾き-3.8 が図７の対象エラー「変数Ａ

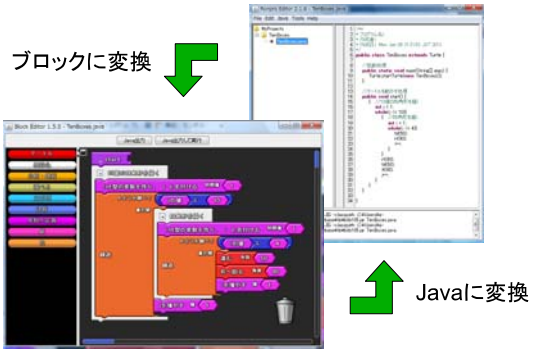

図８．BlockEditor のインタフェース

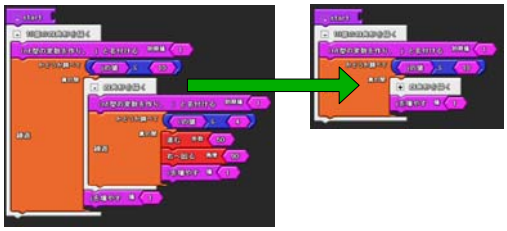

図９．抽象化ブロックと折りたたみ例

は初期化されていない可能性があります」の CTS 値となる．

負の CTS 値はカリキュラムが進むにつれて学習者のエラー修正時間が短くなることを示す．学習が起これば言語仕様を理解していくはずであるため，多くのコンパイルエラーについて CTS 値は負の値となると考えられる．正の CTS 値はカリキュラムが進むにつれて学習者のエラー修正時間が長くなることを示す．学習が進んでいると仮定すれば学習内容の難易度の上昇が，学習による修正時間の短縮を上回っていることをが予想できる．このようにして，提案する指標を利用してプログラミングカリキュラムの評価が可能になることが期待できる．

(3) ブロックエディタ方式によるプログラミング構造化教育支援システムの開発と評価

教育現場でブロック型言語を用いたプログラミング教育や，ブロック型言語の開発が行われている．ブロック型言語はテキスト記述型言語と違い文法エラーが無いので，プログラミング初学者にとって使いやすい言語となっている．テキスト記述型言語で文法エラーを起こりにくくしたり，入力支援をすることによってエラーの問題を回避して，テキスト型言語のプロセスが入門で身につくことが長所である．その一方で，プログラミング初学者にとってブロック型言語と直接比較しているデータは無く，これらの精緻な評価は当該分野の課題である．

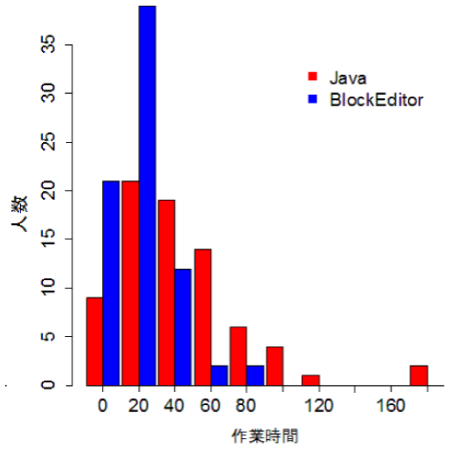

図１０．環境毎の課題回答時間分布

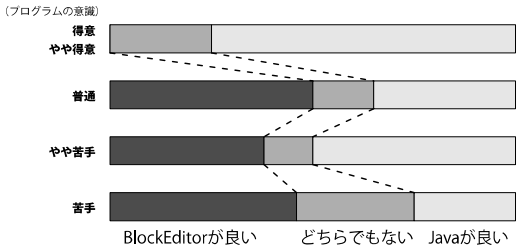

図１１．学生の意識と BlockEditor の評価

ビジュアルプログラミング言語による初等プログラミング教育の環境の提案と改善が行われている．しかしながら，既存のツールは構造化技法をサポートしておらず，C や Java などのテキスト記述型言語への移行も考慮されていないという問題がある．そこで我々は，OpenBlocks フレームワークを利用して，構造化技法をサポートし，Java 言語と相互変換ができるプログラミング教育環境を開発した．

開発した「BlockEditor」の外観を図８に示す．図８の画像が BlockEditor を使用している様子である．プログラムを作るときは，左のウィンドウにある各カテゴリからブロックを取り出し，作業ウィンドウでブロックを組み立てる．Java に変換するときは，エディタの上部にある「Java 出力」というボタンを押すと，BlockEditor で組み立てたプログラムが Java に変換され，初学者向けに開発された Java 開発環境（以下，Java エディタ）に出力される．

BlockEditor では，OpenBlocks フレームワークを拡張し，ブロック型言語においてプログラムを構造的なまとまりごとに分けることができるブロック「抽象化ブロック」を提供している．抽象化ブロックとその折りたたみ例を図９に示す．ブロック上のバーに自然言語でブロックの中に構築したプログラムの目的を記述することができる．また，ブロックを折りたたみ，中に構築したブロックを

隠すことが出来る.

文科系の大学生1年生107名に本システムの使用実験を実施し効果の検証を行った．実験はプログラミング入門科目の第8週の授業で行った．本実験では抽象化ブロックは受講者の任意で利用できる事を説明した.

学生が課題に要した時間を分析し，ヒストグラムに集計した結果の一部を図9に示す．図中の横軸の単位は「分」である．BlockEditor群の方がJava群よりも時間が短いほうに分布している．学習効果による影響も考察してして，BlockEditorの効果を確認できた.

提出されたプログラムをPPVによって分析して抽象化ブロックの利用率，およびコメントの利用率を測定した，その結果，BlockEditor で抽象化ブロックを利用している受講者は Java でコメントを記述している受講者よりも 7%増加していることが分かった．本実験では，抽象化ブロックを受講者に強制的に使用するよう指示していない．抽象化ブロックを利用している受講生は，プログラムの目的毎にまとめるために抽象化ブロックを利用している事が確認できた.

受講者のプログラミングに対する意識とBlokEditorの評価をクロス集計した．プログラミングに対する意識別に対する回答の積み上げ棒グラフを図１０に示す．上から順に，「プログラムが得意，やや得意（4 名：少人数のため統合した)」,「普通（21 名)」「プログラムがやや苦手（16名)」,「苦手（28名)」のデータを示す(無回答は2名)．図８より，受講者の全体的なプログラムの理解度について，「BlockEditor」もしくは「ややBlockEditor」の方が理解しやすいと答えた受講者の数と,「Java」もしくは「やや Java」の方が理解しやすいと答えた受講者の数はほぼ同数であった．プログラミングが「得意」もしくは「やや得意」だと答え,「Java」もしくは「やや Java」の方が理解しにくいと答えた受講者は3名(4.2%)であった.

JavaとBlockEditorのどちらの方がプログラムを理解しやすいかという質問に対して，どちらもほぼ同じぐらいの人数支持者がいる事が分かった．(中西ら，2012)もPEN環境においてとテキスト形式とフローチャートの双方で教育を実施し，同様の報告をしている．このことから，テキスト記述形式，ビジュアルプログラミング形式のどちらか一方が教育現場で利用しやすい，ということは無く，学習者が選択できる環境が必要であるということが明らかになった.

５．主な発表論文等

〔学会発表〕(計 9件)

①榊原康友，松澤芳昭，酒井三四郎(2012)，″プログラミング導入教育におけるコンパイルエラー修正時間の分析″，情報処理学会第74回全国大会，2012.3.7,（名古屋工業大学，愛知県）

②保井元，松澤芳昭，酒井三四郎(2012)，″ブロックエディタ方式によるプログラム構造化教育支援システム″，情報処理学会研究報告 2011-CE-113(11)，pp.1-8，2012.02.05,（三重大学，三重県）.

③松澤芳昭，児玉公信，塩見彰睦，酒井三四郎(2011),″PCA と MVC の複合アーキテクチャスタイルを用いた組込みモデリング技法の提案″，ESS2011，2011.10.20,(代々木オリンピックセンター，東京都)

④Yoshiaki Matsuzawa, Jun Oshima, Ritsuko Oshima, Sanshiro Sakai(2011), ″Experience of applying KBDeX as a Self-Assessment Tool in Collaborative Learning″, COINs11, 2011.09.09,（Hyperwerk, Academy of Art and Design, Basel, Switzerland）

⑤Yoshiaki Matsuzawa, Yasuhiro Noguchi, Takao Mori, Satoshi Shima, Akichika Shiomi (2010),″ESAD: An Intensive Retreat Program for Embedded System Architect Developing″, 17th APSEC, pp.90-97, 2010.12.01,（Hilton, Sydney, Australia）

⑥Yoshiaki Matsuzawa, Jun Oshima, Ritsuko Oshima, Yusuke Niihara, Sanshiro Sakai, (2010),″KBDeX: A Platform for Exploring Discourse in Collaborative Learning″, Coins2010, Web出版，2010.10.09,（College of Art and Design, Savannah, US）

⑦保井元，酒井三四郎，松澤芳昭(2010)，“コメントがプログラムコード実装時間と修正時間に及ぼす影響”，教育システム情報学会第35回全国大会，2010.08.27,（北海道大学，札幌市）

⑧Yoshiaki Matsuzawa, Akichika Shiomi, Tomohiro Haraikawa, Sanshiro Sakai(2009),″Two Challenges to Promote EVM on PBL in Software Engineering Education″, IRSPBL'09, pp.1-10, 2009.12.4,（Victoria University, Melbourne, Australia）

⑨松澤芳昭，大岩元(2009)，“情報系学生を対象としたオブジェクト指向までのプログラミング入門教育の実践と課題”，情報教育シンポジウム（SSS2009），pp199-206, 2009.8.21,（国民宿舎虹の松原ホテル，佐賀県）

６．研究組織

(1)研究代表者

松澤 芳昭（YOSHIAKI MATSUZAWA）

静岡大学・情報学部・助教

研究者番号：40517017

(2)研究分担者

(3)連携研究者