様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　３０　年　　６　月　２１　日現在

機関番号：　３２６８９
研究種目：　基盤研究(B)（一般）
研究期間：　2014～2017
課題番号：　２６２８０１２９

研究課題名（和文）プログラム言語に非依存なプログラミング教育環境の構築と評価

研究課題名（英文）Design and Evaluation of Language Independent Programming Environment for Introductory Programming Education

研究代表者

筧　捷彦（Kakehi, Katsuhiko）

早稲田大学・理工学術院・名誉教授

研究者番号：２００６２６７２

交付決定額（研究期間全体）：（直接経費）　　12,200,000円

研究成果の概要（和文）：本研究では，多プログラミング言語間の翻訳・解釈プラットフォームを開発して，プログラミング言語に非依存なプログラミング教育環境基盤の構築を行い，教育現場での実践に基づく評価を行った．　主要な成果は，１）言語翻訳システムの開発とそれを利用して学習者自身が自由にプログラミング言語を選択してプログラミングを学習できる教育環境の開発，２）学習者がプログラミング取り組む際の細かなログデータの収集とメトリクス測定・視覚化システムの開発などである．本研究成果は，大学情報入試の必要性について調査を行なう研究，プログラミング教育の社会的意義について調査を行なう研究など，様々な研究活動へも繋がった．

研究成果の概要（英文）：In this research, we developed the multi-programming language translation system which enables a translation between any two programming languages described as a module for this system. We developed a development environment for introductory programming education as an application of the language translation system and evaluated by the empirical study in classrooms. Significant achievements are 1) the development of the language translation system and the development of an educational environment where learners can freely select programming languages to learn programming, 2) the development of learners' log collection system which collects fine-grained logs of learners' work on programming, and the development of its visualization system and etc.. These research results also led to further researches such as research to investigate the necessity of informatics in university entrance examination, research to investigate the social significance of programming education.

研究分野：　プログラミング教育

キーワード：　プログラミング教育　プログラミング言語　多プログラミング言語翻訳　抽象構文木　コーディングメトリクス

１．研究開始当初の背景

近年国内外でプログラミング教育が再評価されている．英国では，操作偏重教育への反省からプログラミング教育が再評価され，プログラミング教育の復活が議論されている[1]．米国では，Computational Thinking[2](以下 CT)と名付けられた「世の中の様々な問題について，抽象化，自動化の観点からモデルを組立て，解決するスキル」の概念が提唱され，21 世紀の知識社会に生きるすべての人に必要なスキルとして議論が展開している．

CT を，問題解決能力育成を指向したプログラミング教育と解釈すれば，その学習環境の研究は S.Papert の LOGO 研究[3]以来 30 年以上行われてきた．近年は計算機の速度向上により，プログラミング学習のためのグラフィカルな開発環境が実用速度で動作する時代になり，特に 2000 年以降種々の環境が提案され，実際の教育現場で使われるようになった．A.Kay の Squeak[4]をはじめとする Scratch[5] などのビジュアルプログラミング言語環境，言霊[7]などの日本語プログラミング言語環境，ドリトル[7]などの初学者用オブジェクト指向プログラミング言語，PEN[8]などプログラムの動作を可視化する環境などが挙げられる．

その一方で，既存の教育環境（または開発環境）は言語に強く依存しており，言語を横断して運用できないという課題がある．例えば，PEN に実装されたビジュアルデバッガは有用であるが，言語が xDNCL に制約されるため，選択の機会が限定される．

教育に利用される言語は教師の嗜好にも制約を受けるため，学習者にとって最適な言語が選択されていない可能性もある．最新のプログラミング教育研究成果では，教育段階によって学習者が必要な言語が異なることが明らかになっている．松澤らは Java と Block(ビジュアル型)言語の相互変換システムを開発し，学習者に利用しやすい言語を選択させた所，Block から Java へシームレスに移行していくことを明らかにしている[9].

言語の相互運用の難しさに起因する問題は，研究分野の発展も妨げている．プログラミング教育環境の研究では言語による記述能力や開発環境の相違が大きいという理由で比較研究が難しい．近年のプログラミング教育環境改善は提案が行われるものの簡易な評価しか行われておらず，学術上の進化が滞っている．相互運用環境の整備による精緻な比較研究がのぞまれる．

本研究では，これらの問題を解決するために，多プログラミング言語(以下，「多言語」と略す)間の翻訳システムを開発して，プログラミング言語に非依存なプログラミング教育環境基盤の構築を目指す．この基盤技術として，複数のプログラミング言語に対応したコード処理フレームワーク UNICOEN[10]の成果を活用する．提案基盤は，教育及び解析環境から言語を分離するアーキテクチャを採用して，言語に非依存な環境の開発を支援する開発基盤も提供する．

1.に関する引用文献)

[1] School, C. A.: Computing At School, http://www.computingatschool.org.uk/ (2015.05.16 に参照).
[2] Wing, J.: Computational Thinking, Communications of the ACM, Vol. 49, No. 3, pp.33–35 (2006).
[3] Papert, S.: Mindstorms: children, computers, and powerful ideas, Basic Books, Inc., New York, NY, USA(1980).
[4] Kay, A.: Squeak Etoys authoring & media, Viewpoints Research Institute Research Note, (online), available from ⟨http://www.squeakland.org/resources/articles/article.jsp?id=1008⟩ (2005).
[5] Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B. and Resnick, M.: Scratch: a sneak preview, Proceedings Second International Conference on Creating Connecting and Collaborating through Computing 2004, pp. 104–109 (2004).
[6]大岩元:識字教育としてのプログラミング，情報処理 学会論文誌 教育とコンピュータ，Vol. 1, No. 2, pp. 1–6 (2015).
[7] 兼宗 進，中谷多哉子，御手洗理英，福井眞吾，久野　靖:初中等教育におけるオブジェクト指向プログラミングの実践と評価，情報処理学会論文誌:プログラミング，Vol. 44, No. SIG 13(PRO 18), pp. 58–71 (2003).
[8] 西田知博，原田章，中村亮太，宮本友介，松浦敏雄::初 学者用プログラミング学習環境 PEN の実装と評価，情報 処理学会論文誌，Vol. 48, No. 8, pp. 2736–2747 (2007).
[9] 松澤芳昭，保井元，杉浦学，酒井三四郎:ヒビジュアル-Java 相互変換によるシームレスな言語移行を指向したプログラミング学習環境の提案と評価，情報処理学会論文誌，Vol. 55, No. 1, pp. 57–71 (2014).
[10] 坂本一憲，大橋昭，太田大地，鷲崎弘宜，深澤良 彰:UNICOEN:複数プログラミング言語対応のソース コード処理フレームワーク，情報処理学会論文誌，Vol. 54, No. 2, pp. 945–960 (2013).

２．研究の目的

本研究のテーマは，多プログラミング言語間の翻訳・解釈プラットフォームを開発して，プログラミング言語に非依存なプログラミング教育環境基盤の構築を行うことである．提案プラットフォームによる問題解決の様子を図示し，図１に示す．提案プラットフォームは，教育及び解析環境から言語を分離するアーキテクチャを採用して，言語に非依存な環境の開発を支援する開発基盤を提供する．

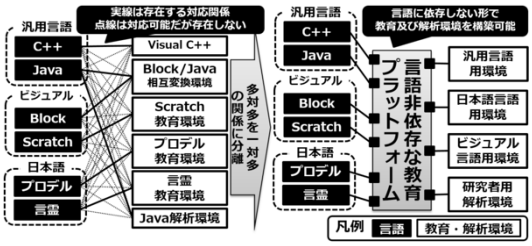

図１．提案プラットフォームによる問題解決

本研究には，２つの大きな目標がある．１つめの目標は，教育者・学習者に対するもので，言語と教育・解析環境の分離により，教育で使用する言語選択の制約を取り除くとともに，学習者と教育者は目的や学習段階に応じて，使用する言語の選択を可能にすることである．同僚の学習者が異なる言語を利用しても良いし，学習者が記述する言語と，教師が評価する言語が異なっても良い．プログラミング入門の教師はフローチャートによる表現を好む．その時，説明のためのフローチャートをプログラムから自動生成することもできるし，学習者がフローチャートで記述したものをそのまま動作ターゲットの言語に変換して動作を確認することもできる．現在は C 言語を利用しているが，ブロックエディタのような新しい環境も段階的に導入したいと考えている教育者などのニーズに応えることもできる．プログラミング教育の文脈，例えば Web システム，組込みシステム，サーバーサイドなど，の選択は学習者の動機付けを得るため等に重要であるが，文脈と言語を分離して，最良の文脈と言語を選択できる可能性も広がる．本環境では，プログラムが表現されている言語は単なる一時的な View となる．従って，プログラミング教育からアルゴリズムを分離して，「書きたい言語で書き，読みたい言語で読む」環境を学習者に提供することが可能になる．

２つめの目標は，教育開発環境の開発者に対するもので，開発者は少ない実装コストで新しい教育環境や対応言語，学習状況の解析機能の追加を可能にすることである．プログラミング教育では，言語だけではなく，プログラミング教育用の IDE(Integrated Development Environment)の開発やビジュアルデバッガの開発によって学習を支援する試みがされてきた．しかし，それらのツールは特定の言語に強く依存しており，複数の言語への対応が高コストであることから，移植は試みられてこなかった．近年ではプログラミング教育版 Learning Analytics と呼ぶべき，学習データ分析ツールが開発されてきており，今後も高度な学習記録分析が行われていくことが期待される．しかし，これらも特定の言語に依存しているため，移植が難しく，普及が阻害されていた．本プラットフォームの開発目標は，言語変換に対応する言語の追加，および IDE，ビジュアルデバッガ，Learning Analytics ツールへの容易な拡張性を持つことである．

３．研究の方法

本研究で提案するプログラミング教育環境プラットフォームの全体像を図２に示す．

プラットフォームは大まかに，「多言語間言語翻訳システム」および，「教育解析環境の開発基盤」からなる．「言語翻訳システム部」開発の基礎技術として，派生版 UNICOEN を開発する．UNICOEN の長所は既に多くの言語に対応するプロセッサを備えており，幅広い能力を持つ中間言語へマッピング可能になっている．しかし，追加の言語対応に関しては，プログラミング言語毎に実装する必要がある点と，言語翻訳システムに利用する場合，言語によって異なる記述能力の違いをどのように吸収し，中間言語を設計するかという点が課題となる．オリジナルの UNICOEN は言語の意味解析については考慮していないため，各文法の和集合(Union)をとって解析結果を処理している．しかし，本フレームワークでは翻訳が必要なため，基本的には，基礎文法部分についての共通部分(Intersection)をとって解析処理を行う方針で新しい抽象 AST を設計する．しかし，静的型の処理など，共通部分とはならないが必要と考えられる文法事項については抽象 AST に含み，静的型付けをしない言語ではアノテーションなどをつけるようにして変換する．

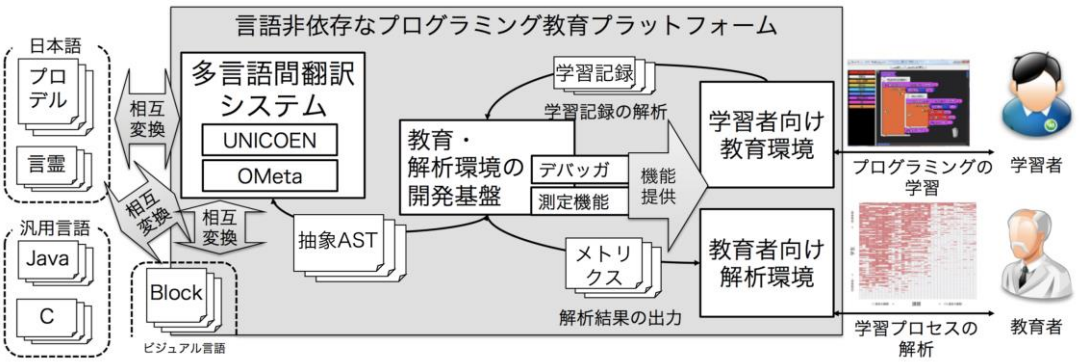

図２．提案プラットフォームのアーキテクチャ

「教育解析環境の開発基盤」については，言語によって異なるコンパイラ処理系，デバッガなどを抽象化し，プラグイン化し低コストで組み込めるように設計した上で，なんらかの VM(Virtual Machine)上で様々な言語から変換されたプログラムを動作させるように設計している．本環境は，多言語を並行的に学習者が利用できる API を提供する．初学者用ビジュアルデバッガ，研究者に必要な学習者の操作ログの取得機能について，本フレームワーク用に移植する．メトリクス測定に関しては，UNICOEN によって既に実装されているので，メトリクスの履歴取得と可視化環境を提供する．

４．研究成果

本章では，２つの主要な研究成果を抜粋し，それらの成果の概要を報告する．

（１）多言語間プログラミング言語翻訳システムの設計と実装

多言語間プログラミング言語翻訳システム JUNICOEN の設計と実装を行った．本システムの設計を図３に示す．

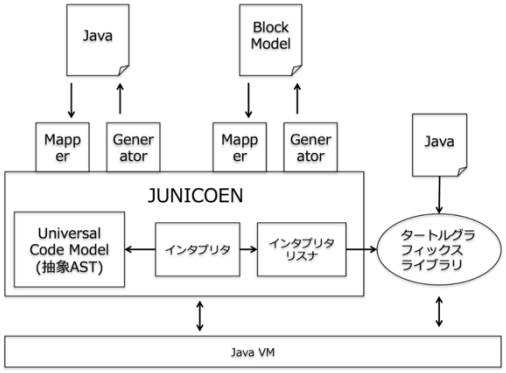

図３．JUNICOEN の設計

UNICOEN 部で扱うコードは抽象コードモデルであるが，実際に動作するソフトウェアの設計においては,UNICOEN 自体を動かすホスト言語の選択が必要である．本プロトタイプでは我々はホスト言語に Java を用い，ターゲットマシンとして Java Virtual Machine を想定した．再現の対象となるシステムがすべて Java で書かれていたため，資産の再利用がしやすいことと，JVM がマルチプラットフォーム上で安定して高速に動くことを考慮しての選択結果である．ただし，Java はセキュリティ上の問題が深刻になってきており，Web ブラウザで動く JavaScript エンジン上で動作することが，教育現場でも必須の条件となってきている．そのため，今後はインタプリタ部の JavaScript 版を開発し，ライブラリを開発して Web 上で直接動作する実装も検討している．オリジナルの UNICOEN は C＃で実装されており，今回の Java 実装版は JUNICOEN と名付けられた．

JUNICOEN は，クラスライブラリとして定義された Universal Code Model(以下，Uni-Model)と，インタプリタ部からなる．Uni-Model は現在，当該カリキュラムで利用する Java の文法に対応した，概して Java 文法のサブセットとして設計されている．インタプリタは Uni-Model を直接，解釈実行するモジュールである．今回はホスト言語として Java を利用しているため，Java に変換し，Java コンパイラでコンパイルしたバイトコードを動かすことも可能である．しかし，その方法では，抽象レイヤで動作するビジュアルデバッガが開発できないため，直接インタプリタで実行するモジュールが設計されている．インタプリタリスナは，抽象実行部とネイティブ言語で直接記述されたライブラリを接続し，ライブラリを利用するプログラムを動作させるモジュールである．この設計によって，ユーザプログラムを抽象レイヤで実行しつつ，タートルグラフィックスのようなネイティブライブラリコールを伴うプログラムが実現できる．

本実装では，Java およびブロック言語へのそれぞれ Mapper と Generator の両方が実装されている．Mapper はゲスト言語から Uni-Model への変換，Generator はその逆変換モジュールを指す．この部分に関しては，対応する言語毎に個別実装をする必要があるが，できる限り安価なコストで実装できるように設計されている．

JUNICOEN―Java 相互変換システム

本節では Java のソースコードと Uni-Model 表現の相互変換を実現するシステム（Java 変換システム）について説明する．図４に Java 変換システムの設計を示す．

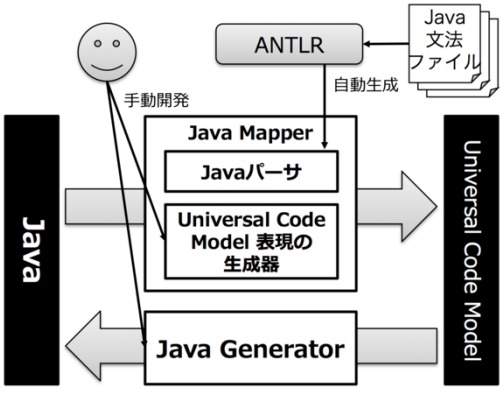

図４．Java Mapper と Generator の設計

Java 変換システムは Java Mapper と Java Generator から構成され，さらに，Java Mapper は Java パーサと Uni-Model 表現の生成器から構成されている．

Java パーサは ANLTR を用いて自動生成して

おり，Uni-Model 表現生成器と JavaGenerator は我々が手作業で開発した.ANLTR はパーサジェネレータであり，GitHub 上に公開されている Java 言語用の文法ファイルを入力して，Java パーサを生成した.

Uni-Model 表現生成器は，ANTLR が生成した Java パーサが出力する具象構文木から JUNICOEN の抽象構文木（Uni-Model 表現）を生成する.ANTLR が生成するパーサが出力する具象構文木は,Visitor パターンで操作できるように設計されている．したがって，Uni-Model 表現生成器は ANTLR が提供する Visitor クラスを拡張することで，出力される具象構文木の構造を読み取り，対応する JUNICOEN の Uni-Model 表現を構築する.

JUNICOEN-Block 相互変換システム

BlockModel と Uni-Modle 表現の相互変換を実現するシステム（Block 変換システム）について説明する．図５に Block 変換システムの設計を示す.

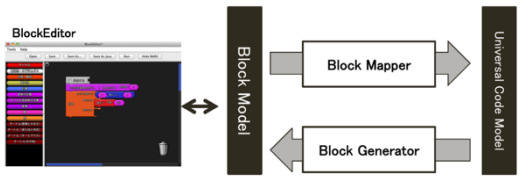

図５．Block Mapper と Generator の設計

BlockModel は，BlockEditor で入出力可能な Block コード形式（DOM）である．この入出力部分の大半は，オリジナルの OpenBlocks の機能を利用している．BlockMapper と BlockGenerator は我々が手作業で実装している.

BlockModel と Uni-Model 変換をするに当たり，BlockModel に若干の修正を加えている．例えば，変数宣言を表す BlockModel に型情報が含まれていなかったものに，型情報を持つように再設計している．これは Uni-Model の変数宣言が型情報を持つため,Uni-Model への変換に不足する情報が出てしまったためである．このため，フレームワークの提供する BlockModel の再設計をし，モデル間の情報の差をなくした BlockModel を Block 変換システムに利用している.

Java 変換システムとは若干手法が異なるものの，BlockModel を再帰的に走査し，対応する Uni-Model を生成するプログラムを作成することで Mapper を作成することができた．各 Uni-Model 表現に対して，対応する BlockMode を作成するプログラムを作成して Generator を作成することができた．従って，今後新たに Uni-Model との相互変換に対応した言語は，Block 型言語との相互変換をすることが可能である.

（２）コーディングメトリクスを用いたブロック言語使用による構文エラー回避効果の計測

Visual Programming Language(VPL)を用いたプログラミング入門教育は，構文エラーによる学習のオーバーヘッドを軽減できることが利点とされている．しかし，実際にどの程度オーバーヘッドを軽減し，学習環境の改善に貢献しているかを定量的に示した研究はない.そこで本研究では，7 つのコーディングメトリクスを定義し，入門教育における VPL 使用効果の定量的な計測を試みた.

本研究の分析対象は，文科系のプログラミングの授業 4 年分(2012～2015)の必修課題 48 題の，受講者，計 404 名分のデータである．本研究で対象とするブロック言語は先に開発した JUNICOEN を利用したブロック-Java 変換システムによる開発環境である.

本研究では，ブロック言語選択による効果を定量的に示すため,「コーディングメトリクス」を定義する．コーディングメトリクスは全て，1 人の学生の 1 課題あたりの数値を示す．以下に各コーディングメトリクスの定義を示す.

- WoT(作業時間)
- BWT(BlockEdtor 利用時間)
- BWT%(BlockEditor 利用時間率)
- ECT(コンパイルエラー修正時間)
- ECT%(コンパイルエラー修正時間率
- JECT%(Java におけるコンパイルエラー修正時間率)：
- LOC(ソースコード行数)：

構文エラー回避効果の分析

コーディングメトリクスを利用して，ブロック言語使用による構文エラー回避効果による，以下の 2 つの仮説を検証した.

仮説 1:「BlockEditor 利用時間率」が大きいほど「コンパイルエラー修正時間率」は小さくなる.

仮説 2:「BlockEditor 利用時間率」が大きいほど「作業時間」は小さくなる.

これらの仮説を検証するために，コーディングメトリクス間の相関係数の算出を行った.算出には pearson の積率相関係数を用いた.

図５に 4 年間のコーディングメトリクスの相関係数を示す.「BlockEditor 利用時間率」と「コンパイルエラー修正時間率」の相関係数は-0.45 となり,負の相関があった.しかし,「BlockEditor 利用時間率」と「作業時間」は 0.10 となり，相関が小さいことが分かった.以上の結果から，仮説 1 は支持されるが，仮説 2 は棄却された.

タイル図を利用して，各指標の時系列分析も行った.「コンパイルエラー修正時間率」のタイル図を図 6 に示す．値が 0 のセルを「薄い緑」，値が 50 以上のセルを「濃い緑」，とした． タイル図の分析結果，学習が進むにつれて,全体的としては「BlockEditor 利用時間率」

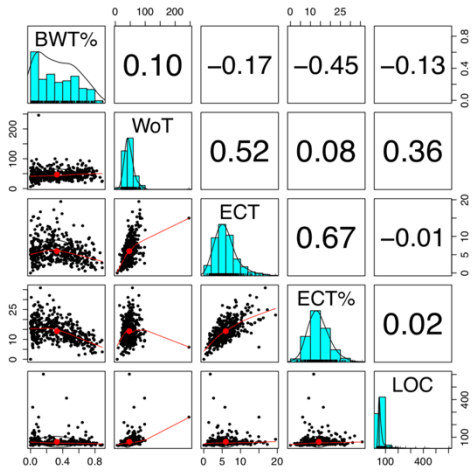

図５．コーディングメトリクス間の相関係数

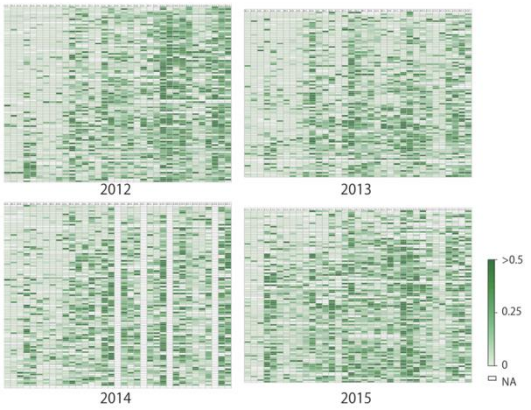

図６．コンパイルエラー修正時間率(ECT%)のタイル図

と対照的に，緩やかに色が濃くなっていく傾向が確認できた．これはブロック言語からJavaへの移行が進むことで，コンパイルエラーが発生し，修正しているためと考えられる．どの年度も色が濃いタイルは，タイル図の中央右よりに集中していた．タイル図の中央から右よりにかけて色が濃い要因として考えられるのは，課題の文法要素が増え，構文に対する理解が追いついていないことや，Javaへ移行した学生が増えてくること等である．

５．主な発表論文等

〔雑誌論文〕（計　13　件）

① Ryosuke Ishizue、Kazunori Sakamoto、Hironori Washizaki 、 Yoshiaki Fukazawa 、 PVC: Visualizing C Programs on Web Browsers for Novices、Proceedings of the 49th ACM Technical Symposium on Computer Science Education、査読有、2018、pp.245-250、https://doi.org/10.1145/3159450.3159566

② Yoshiaki Matsuzawa、Yoshiki Tanaka、Sanshiro Sakai、Measuring an Impact of Block-Based Language in Introductory Programming 、 IFIP Advances in Information and Communication Technology、査読有、Vol. 493、2017、https://doi.org/10.1007/978-3-319-54687-2_2

③ 筧捷彦、論文誌「教育とコンピュータ」の発展に期待する、情報処理学会論文誌教育とコンピュータ（TCE）、査読無、Vol. 1 、 No. 1 、 2015 、 pp.1-3 、 http://id.nii.ac.jp/1001/00112942/

〔学会発表〕（計　24　件）

〔図書〕（計　0　件）

〔産業財産権〕

○出願状況（計　0　件）

○取得状況（計　0　件）

〔その他〕

ホームページ等

(1) JUNICOEN ソースコードの公開
https://github.com/UnicoenProject/Junicoen

(2) JUNICOEN を利用した Java-ブロック相互変換によるプログラミング教育環境のソースコードの公開
https://github.com/macc704/CRiPS

(3)(2)の Javascript 版教育環境(一部未実装)のソースコードの公開
https://github.com/macc704/jsCRiPS

６．研究組織

(1)研究代表者

筧 捷彦（KAKEHI, Katsuhiko）
早稲田大学・理工学術院・名誉教授（2016 年度-現在）
早稲田大学・理工学術院・教授（2014, 2015 年度）
研究者番号：40517017

(2)研究分担者

松澤 芳昭（MATSUZAWA, Yoshiaki）
青山学院大学・社会情報学部・准教授（2017 年度-現在）
青山学院大学・社会情報学部・助教（2015, 2016 年度）
静岡大学・情報学部・講師（2014 年度）
研究者番号：40517017

坂本 一憲（SAKAMOTO, Kazunori）
国立情報学研究所・アーキテクチャ科学研究系・助教（2014 年度-2017 年度）
研究者番号：60609139

(3)連携研究者