情報処理学会論文誌　Vol. 52　No. 12　3135–3149 (Dec. 2011)

プログラミング入門教育を対象としたリアルタイム授業支援システム

長谷川　伸[†1]　松田承一[†2]
高野辰之[†2]　宮川治[†3]

本稿では，プログラミング入門教育を対象として，学習者の提出履歴などの課題取り組み状況を収集・分析可能なシステムを提案し，それを実際のプログラミング入門科目へ実施導入した結果と，それに関する考察を述べる．本システムにおいては，学習者の提出した課題の解答を評価のカテゴリ別に収集・分析し，適切な方法で教授者へ通知することで，授業中における教授者を積極的に支援する．学習者の詳細な提出履歴から学習者の課題に対する試行錯誤の一端を測ることができたほか，学習者の能動的な提出行動から収集される解答は，初回提出においてその 2 割がなんらかの誤りを犯しており，従来では見過ごされていた可能性があることが示唆された．このことから，本手法が教授者支援に関して有効であるといえる．

A Real-time Instruction Support System For Introduction to Computer Programming Education

Shin Hasegawa,[†1] Shoichi Matsuda,[†2]
Tatsuyuki Takano[†2] and Osamu Miyakawa[†3]

In this paper we propose a system to collect and analyze history of learner's submittal. This proposed system is intended for introduction to computer programming education, and in this system, an instructor in lesson is supported by collecting and analyzing source-code that is submitted by learner. And instructor also can understand those situations in real-time according to the proper method. We evaluated the appropriateness of the system for instructors by the application of this system to an actual introduction course to computer programming in university.

1. はじめに

プログラミングを理解するためには，自分自身で実践し，失敗する，あるいは間違いを犯す経験が必要であるとされている[1)]．特に，プログラミングを学び初めたばかりのいわゆる「初学者」にとって，この過程を経験することは非常に重要であると考えられる．また，学習中においては，初学者自身もその多くがそれを無意識のうちに実践している．しかし，難解なコンパイル時・実行時のエラーメッセージからプログラムのミスを把握することは，初学者にとってきわめて困難な行為であり，実践と失敗を繰り返すという過程を経ることに対し敷居が高いのもまた事実である．

近年，大学などの教育機関においてプログラミング教育がさかんに行われている．その多くは授業において，教員の講義後に演習という形式で学習者にプログラムを記述させるスタイルをとっており，学習者には，講義の内容を受けて演習時間内にプログラムを作成させている．演習時間は当然限られているために，プログラム作成に手間どっている学習者に対し，その理解を延々と待ち続けるわけにはいかない．そういった制約内で授業を成立させるためには，学習者の理解を促すための，教員あるいは授業を補佐する指導員（TA: Teaching Assistant）からの学習者に対する適切な助言が不可欠であることはいうまでもない．

しかし，先に述べたような実践と失敗という過程に授業中に対応するには，多くの学習者を少数の指導員で担当するという人的リソースの問題がある．発見の困難なエラーの対処やアルゴリズム記述のミス発見など，指導員に繊細さが要求される場合も多く，ときには 100 人単位で行われる授業中においては，その指導に限界があった．

そこで，我々は統合リアルタイム授業支援システム「IDISS: Integrated and Dynamic Instruction Support System」を開発した．本システムでは，授業中リアルタイムに課題を提示し，それに対し提出された学習者のソースコードに関して，コンパイルやインデントなどを自動的に評価し通知することができる．また，学習者の提出履歴を収集し，分析を行うことができる．これによって人的リソースの問題を解消されるほか，リアルタイムで課題を作成しながら理解度に応じた授業進行も可能となる．

†1 ニフティ株式会社
NIFTY Corporation
†2 東京電機大学大学院
Graduate School of Tokyo Denki University
†3 東京電機大学
Tokyo Denki University

© 2011 Information Processing Society of Japan

プログラミング教育において，このような教育支援システムは数多く提案されてきているが，我々が開発したシステムとはそのコンセプトを異にしており，また教員や TA が一体となって活用できるような教授者支援のシステムは見当らない．

本稿では，開発した統合的な授業支援システムについて，その有用性を検証し考察を述べる．

2. リアルタイム授業支援システムの仕様

2.1 支援の対象と使用の想定

本システムが想定する支援の対象は，主にコンピュータプログラミングの初学者である．また，システムは大学における授業中に利用し，授業中にいくつかの課題を学習者に提示し，解答させる．次に述べるような仕様に基づいたシステムを利用することで，主に教授者を支援する．

2.2 基本的な仕様

本システムは，学習者のプログラムを動的に解析し，教授者ならびに TA に対し，その結果をその場で適切に通知することで，授業進行をリアルタイムに支援することを主な目的とする．この目的を達成するために，本システムでは次の 4 点を基本的な仕様として定めている．

2.2.1 評 価 項 目

学習者が逐次提出するプログラムに対し，本システムでは次の 4 点の項目で評価し，その情報を収集する．

(1) **コンパイル**
提出されたプログラムがコンパイル可能であるかどうかを評価する．コンパイルの可否は，学習者が正しいプログラムを記述できているかどうかの一指標となりうる．

(2) **インデント**
プログラミングを行う際，インデント（字下げ）はプログラムのブロック構造を理解するうえで重要な要素であるとされる[2]．また，適切にインデントされたプログラムは，多くの場合可読性が高く，このようなプログラムを記述できるようになることを教育目標に掲げることは妥当である．しかし，多くの汎用言語（C 言語や Java 言語など）は，構文にフリーフォーマット形式を採用しており，またインデントはコンパイル時に検査されず，学習者はこれを見過ごしてしまう．こうしたことから，インデントを明示的に評価し，間違いがあれば学習者に提示できるようにする．

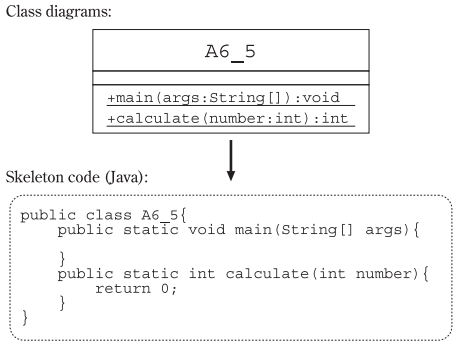

図 1　クラスの形式的記述の例
Fig. 1　Example of generating skeleton code.

(3) **クラスの形式的記述**
クラスの形式的記述とは，与えられた UML のクラス図からクラス名やメソッドおよびその引数，返却値の型など，プログラムの骨格（スケルトンコード）を形式的に導出し，最低限コンパイルが通る状態まで記述することをいう[*1]．クラスの形式的記述の例を**図 1** に示す．クラスの形式的記述を行わせることで，コンパイルエラーが発生した際には，クラスやメソッドの宣言が不正なのか，実装部分が間違っているのかの区別が明確になる．また，UML で示された仕様を満たしているのかどうかを検査することもできる．

(4) **ユニットテスト**
プログラム中に宣言された各メソッドが，与えられた仕様どおりの正しい振舞いをするかどうかを検査する．引数に対して適切に処理を行えているか，不正な出力命令が含まれていないかなどを検査し，プログラムの正当性を評価する．

2.2.2 収　　集

評価された項目は，学習者の情報とともに逐一，情報として収集し，保存する．収集した情報は分析の対象となる．

---

*1 なお，本システムにおいてはクラス図からソースコードを記述する作業を「機械的導出」と呼称しているが，本稿では．意味を分かりやすくするために「クラスの形式的記述」としている．

© 2011 Information Processing Society of Japan

2.2.3 分　　析

収集した情報からさまざまな分析が可能である．現在時刻における学習者の状況や，授業終了までの連続的な学習者の提出状況の変化など，リアルタイム，あるいは事後分析が可能である．具体的な分析の内容については 4 章および 5 章で述べる．

2.2.4 リアルタイム

事後分析を除くこれらの行動をすべて，授業中にリアルタイムで行う．すなわち，提出・評価・収集・分析から閲覧までである．

2.3 教授者側の仕様

教授者は，授業を進行させながらリアルタイムにシステムを操作できる必要がある．本システムにおいて，教授者が利用できる機能の仕様は次のとおりである．

- **課題の登録と提示**
  教授者は，カリキュラムの進行や授業の進行度合いによって，その場で課題をシステムに登録できることが必要となる．
- **課題の提出状況の閲覧**
  教授者は，学習者が提出している課題の提出状況を，課題別に閲覧できる．提出状況は，システムが評価した評価項目に基づき，正解・不正解・未提出の各学習者のリストと人数を表示するとともに，評価項目がすべて正解である学習者の百分率を表示する．
- **学習者の課題進度の閲覧**
  学習者の課題に対する進行度を閲覧できる．進行度とは，授業に出席しているすべての学習者の相対的な順位をある一定のアルゴリズムによって算出したものである．この順位は，学習者が課題を提出するたびに更新されてゆく．順位付けには，評価項目に対する正解・不正解，提出時間などを加味する．学習者に順位付けを行うことで，授業中の課題の取り組み状況を教授者が俯瞰的に把握することができる．ただし，この順位付けは仮想的なものであり，課題進度による学習者の選別などに用いることとする．このアルゴリズムに関しては，4 章において述べる．
- **座席情報と個別解答の閲覧**
  授業に出席している学習者の座席情報を閲覧できる．また，各学習者の課題に対して提出された個別解答を閲覧することができる．学習者の座席位置を閲覧できることによって，進度の遅い，あるいは不正解の多いなどの学習者に対し，教授者あるいは TA が適切に指導を行うことができるようになる．

このように，教授者側においてはさまざまな要素を閲覧することができるが，一方で情報量が多いことから，授業の進行を妨げないためにも，一べつして授業の進度や提出状況，座席位置などを閲覧できることが望ましい．したがって，教授者が閲覧に利用するクライアントアプリケーションは，このような仕様を満たしているとともに，グラフィカルに分かりやすく情報を表示できる．

2.4 学習者側の仕様

本システムでは学習者に関して，個人所有のコンピュータ上でプログラミングを行えるよう，さまざまなオペレーティングシステムやプラットフォームを対象としている．したがって，教授者側で何らかの調整した環境を利用させる，あるいは Web アプリケーション上でプログラミングをさせるのではなく，マルチプラットフォーム上で動作するクライアントアプリケーションを学習者にダウンロードさせ，利用させるのが適していると考える．また，クライアントアプリケーションは，機能拡充を目的に頻繁にアップデートされる可能性があるため，学習者側にインストールされたクライアントアプリケーションも自動的にアップデートされる環境が望ましい．このような要求から，実際のシステムでは，学習者側のクライアントアプリケーションの実行環境として，JavaVM 上で稼働するアプリケーションとする．Java の実行環境（ランタイム）は，昨今のオペレーティングシステムには標準で搭載されており，同じアプリケーションをさまざまなプラットフォーム上で動作させることができる．

また，学習者の着席位置が自由であることを考慮して，学習者に自身の着席位置を入力させる．これは本システムの範ちゅうではないものの，自身の位置を偽って入力する学習者も想定されるため，正しい位置情報を入力することで，適切なサポートを受けられるメリットを学習者に認知させる必要があると考えられる．

課題の提出方法は，学習者の学習意欲を削がないよう，簡単に行える必要がある．ただし，先に述べたように，学習者のプログラミング環境に対し何らかの変更を加えることはできない．また，実践的な教育という観点から，学習者が特殊な環境でのプログラミングにのみ適応してしまうことも避けなければならない．このことから，学習者が利用するクライアントアプリケーションは，インタラクティブなインタフェーイスを備えることとする．また，学習者のソースコードは，学習者の能動的な提出行為によって収集することとする．

学習者の課題取り組み状況をリアルタイムに把握するにあたっては，このような収集方法を採用する際に，次のような側面が存在する．すなわち，学習者は課題を提出しないという選択もでき，逐一の収集を行えないという点である．本システムにおいては，「逐一，すべてを収集する」という側面よりも「学習者が正しいと思っている解答を収集する」という

© 2011 Information Processing Society of Japan

コンセプトをとり，実践と失敗という過程を学習者主観の観点から分析できると考えた．また，「提出していない」ことが情報として取得できるため，進行度の参考となるほか，提出できないほど極端に進度の遅い学習者の発見にもつながる．

3. 関連研究

プログラミング入門教育において，学習者もしくは教授者を支援するシステムはこれまで数多く研究されてきている．

西田らは，初学者教育用プログラミング環境 PEN と呼ばれる学習環境を構築し，評価を行っている．このシステムでは，日本語をベースとした xDNCL と呼ばれる手順記述言語を用いて，プログラミングを分かりやすく学習者に教授しようと試みている[3]．

斐品らは，アルゴリズム学習支援システムを提案している．このシステムでは，プログラミング言語によらずにアルゴリズムの理解を促すために，JPADet と呼ばれるツールを用い，抽象化されたアルゴリズムを図式的に学習者に解答させることができる[4]．

知見らは，学習者に内省（問題解決の過程を振り返る行為）を行わせることで，プログラミングの知識を定着させる学習環境を構築している．学習者のコンパイルエラー，実行エラー，あるいは学習者自身が入力する論理エラーや，これらのエラーから学習者が入力した内省から，学習者の学習プロセスの把握を試みており，一定の成果をあげている[5]．

これらの環境やシステムは，主に学習者の支援を目的としており，我々が開発した IDISS とはその目的が異なる．知見らのシステムは，失敗と実践を繰り返させる，あるいは教授者が学習者の学習プロセスを把握できるという点では IDISS と同様のコンセプトを持つが，学習者各自の個別学習を対象としている点が異なる．

一方で，教授者支援のためのシステムも提案されている．

小西らは，プログラミング教育における教育目標を，プログラミング言語に依存した目標と，プログラミング言語によらない目標に分類している．そのうち，プログラミング言語に依存した目標に関して，学習者のプログラムが，教授者が陽に PAD 表現で記述した「標準アルゴリズム」に対応するかどうかを解析し，教育目標を満たしているか検査する．このとき，小西らは教育支援システムにおけるプログラムの評価は教授者が想定した教育目標を満たしているか否かを基準に行われるべきであり，計算結果の等価性のみに着目してプログラムを評価すべきではないとしている[6]．我々も小西らと基本的には同様の立場をとるが，しかし，その裾野をプログラムの表層構造にまで広げた場合，学習者の思考の差異から「正しくプログラムを記述できていない」と評価されてしまうことがありうる．特に，評価対象のプログラムがアルゴリズム記述に限らない場合（クラスの形式的記述など），リアルタイムに評価を下す場合，プログラム記述にある程度の余裕を持たせることが必要であると考える．そういった観点から，本システム IDISS では構造の理解という教育目標に対し，インデントを検査するという方法で評価を行っている．また，教授者側が取得できるのは学習者の最終解答とその分析結果のみであり，解決へのプロセスを把握することはできない．リアルタイムな授業支援を目的としていない点も異なる．

学習者の解決へのプロセスを把握する観点から，Spacco らは AutoCVS という統合開発環境 Eclipse のプラグインを用いて学習者のプログラムを収集するシステムを提案している[7]．また，Jadud らはプログラミング用エディタ BlueJ を用いて学習者のスナップショットを記録する方法を試みている[8]．しかし，プログラムの自動評価や分析を目的としておらず，リアルタイムな授業支援に用いることを想定していない．

リアルタイムに学習者の学習状況を収集し蓄積できるシステムとして，玉田らは，アシスタントロボットを用いた教育支援システムを構築している[9]．このシステムは，学習者，教授者双方に対し，講義中の支援を提供しており，学習者のコンパイル行動をリアルタイムで把握し，エラーメッセージを分析，収集したのちに，その統計値を教授者に提示することができる．

また，倉澤らは，プログラミングの一斉授業を対象として，学習状況把握支援システムを開発している[10]．このシステムでは，アニメーションを用いて学習者に対しプログラムの動作理解を促すことができるほか，リアルタイムで学習者のプログラムのコンパイルエラーを収集・分析し，その傾向を教授者に提示することができる．リアルタイムでこの動作が行われることで，ダイナミックに学習者全体の傾向の把握が可能となるとしている．

これらのシステムは，リアルタイムに学習者のエラーを収集・分析し，教授者に提示できるという点は IDISS と共通するが，収集するエラーがコンパイルエラーのみであることが異なる．また，学習者は特殊な環境（管理者調整のネットブート OS を利用させる，あるいは Web インタフェーイス上でプログラミングを行わせる，特定のエディタの利用を強制する，など）でなければシステムを利用することができず，個人所有のコンピュータ上でプログラミングを行ううえで大きな障害となるほか，学習者に特定のインタフェーイス上でプログラミングをさせることは，コンパイルコマンドやコマンドラインの利用など，プログラミングに付随する基礎的知識を教授するという観点からも，あまり実践的ではない．IDISS では JavaVM が搭載されたコンピュータであれば，クライアントアプリケーションを起動させるだけで利用できるため，学習者の環境に左右されずにシステムを利用することができ

© 2011 Information Processing Society of Japan

る．また，コンパイル環境が整っていれば，学習者は任意のエディタやツールなどを利用することができる．

そして，これらあげたシステムの多くは，学習者支援あるいは教授者支援に限らず，プログラミングにおけるアルゴリズム学習を主眼としており，プログラミング入門教育において重要となる，プログラムの形式的記述などを主な教育目標とするカリキュラムに対応可能なシステムとはいい難い．また，より実践的な教育を行うという観点から，独自のプログラミング言語を用いるという選択肢も棄却される．

なお，プログラムの自動評価という観点から，Edwards は学習者にプログラムを提出させ，それを自動評価し，その結果を即座に学習者に提示する Web-CAT というシステムを提案し，構築している[11),12)]．Web-CAT は学習者に対し自動的に順位付けを行うほか，テスト駆動開発（TDD: Test Driven Development）アプローチなど重視し，評価にはプログラム開発手法のテストファーストを取り入れている．これらのコンセプトは我々のシステムと共通しているが，プログラミング初学者に対しテスト作成をも求めるのはハードルが高いため，IDISS とは利用対象者が異なると考えられる．

4. 実 装 環 境

4.1 システム構成

本章では，2 章で述べた仕様に基づいて開発したシステムの構成について述べる．

本システムは，フロントエンドとして教授者側クライアント・学習者側クライアント，そしてそれらの統一的なバックエンドで構成される．これらシステムの構成を**図 2** に示すとともに，次節からその詳細を述べる．

4.2 バックエンド

バックエンドは，学習者がソースコードを提出し，またそれを解析する「課題提出・評価部」，学習者の座席情報を収集する「出欠・位置確認部」，情報を適時加工し，閲覧するための API を規定する「インタフェーイス定義部」，それらの情報を一括して蓄積しているデータベースで構成される．

4.2.1 基 本 構 成

本システムのバックエンドは，3 つの部から構成される．以下にその詳細について述べる．

出欠・位置確認部　学習者の座席位置指定インタフェーイスから送信される座席情報を受信する．受信した情報は部門間共有のデータベースに保存・管理される．

課題提出・評価部　学習者が提出したソースコードを受け付け，解析・蓄積を行う．課題提出・評価部がソースコードを受付けると，即座にそのコンパイル可否の検査・インデント検査・クラスの形式的記述の検査・ユニットテストが行われる．ただし，現状ではユニットテストをシステムに実装していない[⋆1]．これらの情報は即座に学習者側クライアントアプリケーションに通知するとともに，データベースに保存される．バックエンド，学習者側クライアントアプリケーション双方に詳細なエラー情報が通知されているが，学習者にその情報をどの程度詳細に見せるかはクライアントアプリケーションの実装により調整可能である．現実装では，各評価項目に対する “○” と “×” のみを学習者に対し表示している．

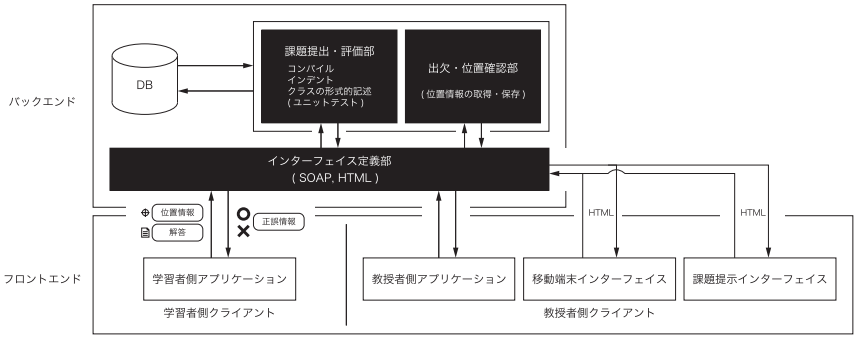

図 2　リアルタイム授業支援システムの構成
Fig. 2　Constructure of real-time instruction support system.

インタフェーイス定義部　データベースに蓄積された情報を適時加工し，閲覧できるようにするための API（Application Programming Interface）を規定する．また，位置情報や課題提出に関する API も含んでいる．これらは，HTTP によるウェブサービスとして実装されており，学習者側クライアントアプリケーション，教員側クライアントアプリケーションともにこの API を利用して情報のやりとりを行う．また，サービスとして稼働させることで，クライアントの追加や別のシステムとの連携を行うことができるようになる．現状では次のようなサービスが稼働している．

- 学習者の認証

⋆1 ユニットテストは，学習者のソースコードを実行させるという性質上，サンドボックス上での実行が要求されるほか，無限ループなどにも対応させる必要がある．現状はこの機構について設計途上であり，実装は今後行っていく．

© 2011 Information Processing Society of Japan

- 座席情報をともなった学習者の認証
- 学習者の位置情報の取得
- 課題の作成
- 当日に出題された問題以外の問題を取得
- 当日に出題されている問題を取得
- 教授者が受け持つ科目の一覧を取得
- 学習者の提出物の一覧を取得
- 授業の現在時刻の進行度を取得
- 特定の時刻における学習者の進行度を取得
- ソースコードを提出

これらのサービスは，SOAP（Simple Object Access Protocol）を利用しており，クライアントの実装を問わない．

4.2.2　進行度の算出

学習者の進行度に関しては，次のようなアルゴリズムにおいて算出する．

進行度は，算出しようとしている時点において，授業に出席しているすべての学習者を対象として相対的に算出される．すべての学習者に対し，以下に述べるような評価の優先度に基づいて順位付けを行う．学習者数が変動したときの影響を少なくするため，最終的な進行度は，この順位を 1 から 100 の間で正規化した値とする．

まず，現在出題されている問題（複数出題されている場合には問題群）に対しての完全正解数が大きい学習者を上位とする．完全正解数とは，学習者のソースコードが，コンパイル・インデント・クラスの形式的記述・ユニットテストのすべての評価項目に対して正解している問題数である．たとえば，算出時点において 3 問の問題が出題されているとき，2 問完全正解，1 問インデントの間違いがあった場合，完全正解数は 2 である．

完全正解数が複数の学習者間で同一であった場合，どの問題に対しての解答かにかかわらず，学習者が最後に正解の解答を提出した時間が早い学習者を上位とする．たとえば，完全正解数が 1 の学習者が 2 名いたとして，どの問題に対しての解答かにかかわらず，学習者 A は 15 時 50 分，学習者 B は 15 時 51 分に正解の解答を提出した場合，学習者 A が上位となる．

また，未提出の学習者がいた場合は，その学習者の順位はすべて同一となる．未提出とは，出題された問題に対していっさい解答をしていない状態を指す．複数の問題が出題されている場合，すべての問題に対しいっさい解答していない状態である．たとえば，学習者数

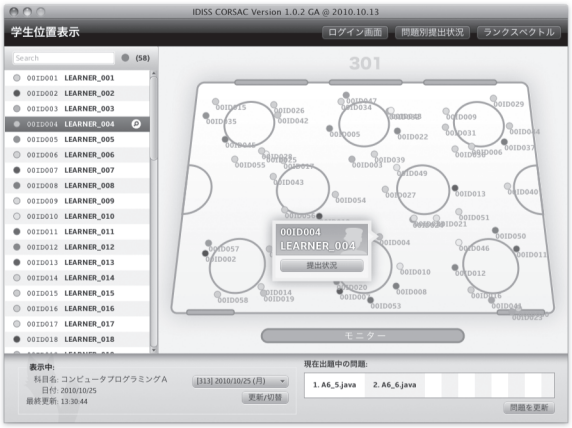

図 3　教授者側クライアントアプリケーションの外観
Fig. 3　Screenshot of instructor side client application.

30 名の授業において，10 名が未提出であった場合．未提出の学習者は一律で 21 という順位を与えられる．

4.3　フロントエンド

ここでは，教授者および学習者が利用するフロントエンドについて述べる．双方ともにスタンドアローンのクライアントアプリケーションとして実装される．クライアントアプリケーションは，Java Web Start Architecture[13] を用い，クロスプラットフォーム上で動作する．実装には JavaFX[14] を用い，従来よりも洗練され学習者・教授者双方に利用しやすいインタフェーイスを提供する．

4.3.1　教　授　者

教授者側は，クライアントアプリケーション，またはウェブ・ブラウザを利用して学習者の情報を閲覧することができる．

授業中において，教員は主にクライアントアプリケーションを利用して情報を閲覧できる．

クライアントアプリケーションの外観を**図 3** に示す．教員側のクライアントアプリケーションでは，学習者自身が入力した座席位置を一覧できる．また，学習者ごとの個別解答やその正誤を瞬時に閲覧できる．学習者の位置は教室の俯瞰図上に「円」として表示される．この円は，経過時間や学習者の課題提出状況，達成度などの情報をもとに，進行度としてその色がリアルタイムで変化する．この色の変化を教員が観察することで，教室全体における

© 2011 Information Processing Society of Japan

図 4　進行度の分析画面
Fig. 4　Analysis screen of progress-rank.

課題の取り組み状況や理解度などを視覚的に，かつ瞬時に把握することができ，授業進行を支援する．進行度は，バックエンドにおける「インタフェーイス定義部」において算出される値を利用する．また，授業開始から現在時刻までの進行度の推移をタイムラインで確認することもできるほか，任意の授業における任意時間の進行度を表示することもできる．進行度の分析画面を**図 4** に示す．

また，課題の提示は，ウェブブラウザによって行う．これは，教員ではない教授者（TA など）も課題を提示できるようにするためである．課題の提示は，解答となるソースコードを入力し，送信するだけで完了する．

TA などの授業をサポートする教授者に対しては，「常時教室内を歩き回り指導を行う」という職務の性質上，移動端末による情報確認が望ましいと考えられるため，Apple 社の iPod touch に搭載されるブラウザに最適化した情報確認画面を提供する．その外観を**図 5** に示す．この確認画面には，提示された課題に対し「未提出」あるいは「不正解」の学習者の名前と位置，その間違いの詳細が表示される．

4.3.2 学　習　者

学習者は，クライアントアプリケーションを利用して，ログインを兼ねた自身の座席位置の送信と，課題の提出を行う．

アプリケーションは，起動時に学習者に座席位置の指定を促す．これを座席位置指定インタフェーイスと呼ぶ．ひと形のアイコンをドラッグすることで，座席位置を指定できる．座席指定の画像は，利用する教室の形態によって切替え可能な設計となっている．座席位置を入力できることで，授業中に自由座席で個人所有のコンピュータからシステムを利用することも可能になる．

図 5　iPod touch における情報確認画面
Fig. 5　Displays for iPod touch.

位置の指定と同時にログインを促し，位置情報の送信とログインが完了すると，クライアントアプリケーションは自動的に課題提出のインタフェーイスに切り替わる．クライアントアプリケーションの外観とその画面遷移を**図 6** に示す[⋆1]．

学習者はは，ソースコードのファイルを指定された枠にドラッグ&ドロップすることで課題を提出することができる．ファイルは自動的にバックエンドにおける「課題評価・提出部」に送信される．そして，コンパイル，インデント，クラスの形式的記述，ユニットテストの正誤を学習者に通知する．正誤は，評価項目ごとに，正解の場合は “○”，不正解の場合は “×” のみを通知している．これは，自身の間違いを自身の力で発見させ，実践と失敗の過程を経験させるためである．また，過去に出題された課題に対する解答も行えるようになっている．

4.4 利 用 方 法

学習者は，授業開始時もしくは教授者の指示によって学習者側クライアントアプリケー

⋆1 図 6 では，「クラスの形式的記述」を「機械的導出」としている．

© 2011 Information Processing Society of Japan

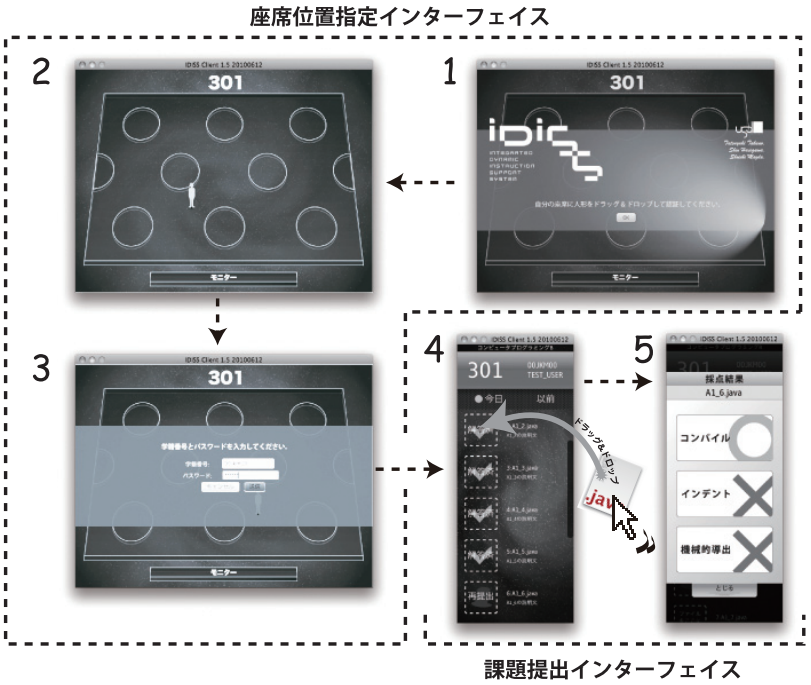

図 6　学習者用クライアントアプリケーションの画面遷移
Fig. 6　Transitions of learner side client application.

ションを起動する．アプリケーションは Web 上の URL にアクセスするだけで起動できるため，当該 URL を授業の Web コンテンツ上にあらかじめ配備しておく．

学習者がクライアントアプリケーションを起動した後，座席位置送信インタフェーイスを利用し，出欠のためのログインと自身の座席位置を送信させる．

ログインと位置送信が完了すると，アプリケーションのインタフェーイスは自動的に課題提出インタフェーイスに切り替わる．学習者は，教授者が授業前にあらかじめ登録した課題，あるいは授業進行中にリアルタイムで登録した課題を，教授者の指示を受けながら解答し，提出インタフェーイスより学習者任意のタイミングでソースコードを逐一提出する．

教授者は，教授者側クライアントアプリケーションを起動し，リアルタイムな提出状況や進行度の推移を観察する．

表 1　本システムを試験導入した科目
Table 1　Subjects applyied to IDISS.

科目名	コンピュータプログラミング A
略称	CPA
主な学習目標	構造プログラミング
履修人数	61 人
期間	2010 年 9 月 22 日〜12 月 22 日

5. 実施検証

2010 年 9 月 22 日から，同年 12 月 22 日にかけて，東京電機大学情報環境学部情報環境学科で開講された，プログラミング入門科目であるコンピュータプログラミング A（以下，CPA）において，本システムを試験的に導入し，その動作検証を行った．本章では，この検証により収集された各種の情報から，さまざまな分析を試み，その考察を行うことを目的とする．

試験導入した科目について**表 1** に示す．

CPA では，Java 言語を用いて．クラスやメソッドの宣言（クラスの形式的記述を含む），制御構造の使用方法，配列などを学習する．本科目では構造プログラミングの理解と習得を目標としている．

5.1　個人別提出履歴の追跡

本システムでは，学習者の課題提出を逐一収集しており，その履歴を任意に追跡することができる．この行為は授業中，あるいは授業終了後など，教授者任意の時間に行うことができる．

本節では，収集された学習者の提出履歴のうち，試行錯誤が顕著な学習者を抽出し，その過程を一覧する．抽出方法は，完全正解までにかかった提出回数が 5 回以上であり，同一カテゴリのエラーが 2 回以上連続している学習者を抽出し，試行錯誤が顕著かどうかを目視で判断した．抽出した学習者 A の提出履歴をまとめたものを**表 2** に示す．

また，問題 127 における提出動向を**表 3** に示す．

5.1.1　詳細追跡

抽出した学習者の提出履歴に対応する課題は，問題 ID 127 であった．以下，この問題を問題 127 とする．表中の “○” は正解を，“×” は不正解を表している．また，これら記号直下の行は，不正解の原因を表している．本来はシステムが自動的にメッセージを付与する

© 2011 Information Processing Society of Japan

表 2　抽出された学習者の提出履歴
Table 2　History of Learner's submittal.

学習者 A の提出履歴（2010 年 12 月 1 日/問題 ID127/CPA）

提出時間	コンパイル	インデント	クラスの形式的記述
14:52	× ファイル名：Janken2.java 未定義の変数の使用：janken	○ —	× getHandsign(int numnber) printlnputMessage()
15:03	○ —	○ —	× class Janken2 getHandsign(int number){} printlnputMessage(){}
15:05	× ファイル名：janken.java	○ —	× class Janken2 getHandsign(int number){} printlnputMessage(){}
15:07	○ —	○ —	× getHandsign(int number){}
15:08	○ —	○ —	× getHandsign(int number){}
15:09	× 未定義変数の使用：HandSigns	○ —	○ —
15:10	× 未定義変数の使用：HandSigns	○ —	○ —
15:12	○ —	○ —	○ —

○：正解，×：不正解，—：メッセージなし

表 3　問題 127 の提出動向
Table 3　Trend on Question 127.

提出人数（人）		55
平均提出回数（回）		1.54
エラー件数（件）		45
提出回数の内訳（人）	8 回	1
	4 回	2
	3 回	4
	2 回	9
	1 回	39
エラー内訳（件）	コンパイル	6
	インデント	5
	クラスの形式的記述	34

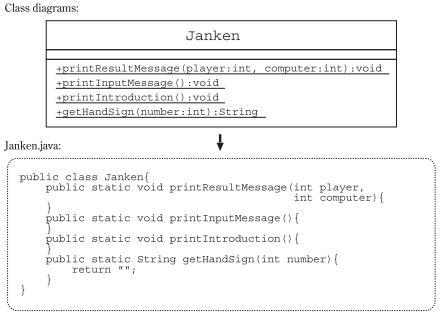

図 7　問題 127 の模範解答
Fig. 7　Correct answer of Question 127.

が，ここでは簡略化したものを示している．また，この「原因」は模範解答に対し，学習者のソースコードにおいてシステムが誤答と判断した箇所をそのまま抜き出したものである．

問題 127 では，学習者に UML のクラス図から，クラスの形式的記述を求めている．この問題の模範解答を**図 7** に示す．表 2 から，問題 127 に対し学習者 A は，完全正解までに 8 回の提出を行っていることが分かる．問題 127 では，図 3 から平均は 1.5 回となっており，この学習者は完全正解までに平均よりも多くの提出を行っている．

また，インデントは早期から正解となっているが，一方でコンパイル，クラスの形式的記述において，つづりのミス，大文字・小文字の区別などの試行錯誤を繰り返している様子がうかがえる．以下に，それぞれの提出における考察を述べる．

14:52　クラス名を Janken としながら，ファイル名は Janken2.java としてしまっている．Java 言語では，public クラス名とファイル名は一致させなければならないため，これはコンパイル・エラーとなる．この問題 127 は，前日に出題したクラス Janken に，メソッドを追加した状態で再度クラスの形式的記述をさせたものである．そのため，前日のファイルへの上書きを嫌って，ファイル名を変更したために起こったエラーであると推測される．また，クラスの形式的記述では，メソッドのつづりエラーを起こしている．特に printlnputMessage では，大文字の I と小文字の l を混同して記述している．教材はゴシック体を用いて記述されており，判別が付かなかった可能性がある．

15:03　クラス名とファイル名の不一致に関していえば， Janken2 で統一し解消されたためコンパイル可能となった．コンパイルエラーのメッセージを適切に読み取れていることが分かるほか，エラーを 1 つずつ解決して次へ進もうとしていることが読み取れる．

© 2011 Information Processing Society of Japan

提出までに 10 分強かかっていることからも，自身で考え，実践していると考えられる．

15:05　ファイル名・ファイル名に対しての試行錯誤の途中であると考えられる．

15:07-08　メソッド `getHandSign` における大文字・小文字がいかに区別し難いかが読み取れる．

15:09-10　未定義変数の使用は，実装部に `handSign` という変数を使用しており，メソッド名と混同し修正してしまったと考えられる．

15:12　このような試行錯誤を経て，学習者 A は最終的に正解へとたどり着いている．

5.1.2　出題問題への解答動向

表 3 は，この問題 127 に関する提出動向を示している．学習者 A の提出回数は，問題 127 における平均提出回数を大きく上回っている．また，提出回数が 4 回の学習者は 3 名，3 回の学習者は 4 名，2 回の学習者は 9 名であった．これらの学習者のうちの多く，特に提出回数が 3 回以上の学習者に関しては，クラスの形式的記述に関する間違いが多数を占めた．また，これらの学習者の多くは，初回の提出ではコンパイルとクラスの形式的記述を間違え，その後はクラスの形式的記述を何度も試行錯誤するという傾向がみられた．

5.2　初回提出解答の動向

本節では，学習者の初回提出解答からその動向を示す．初回提出解答とは，出題された問題に対し，学習者が提出した解答のうち，最初回のものをいう．したがって，1 回の授業において問題が 2 問出題された場合，各学習者の初回提出は 2 件となる．ただし，未提出の場合は，件数には含まれない．

CPA における初回提出解答の動向を**表 4** に示す．実施日とは 1 題でも問題が出題されている授業実施日を示している．また，「誤答人数」は，1 問目あるいは 2 問目において誤答と判断された学習者の人数である．「提出人数」は，その問題に解答を送信した学習者の人数である．「エラー件数」とは，その授業実施日に検出されたエラー件数をカテゴリ別に総計した件数である．また，表 4 におけるエラー件数の合計を割合として示したものを**図 8** に示す．

表 4 より，提出人数に対する誤答人数比が 0.238 となっている．これは，各問題の初回提出解答において，約 2 割の学習者が誤答と判断されていることを意味する．

図 8 より，初回提出解答においてはクラスの形式的記述に関するエラーが割合としては最も多い．また，学習者が自身のプログラミング環境において正誤を判断できるはずのコンパイルに関するエラーも，インデントのエラーと同程度の割合で発生していることが分かる．

表 4　CPA における初回提出解答の動向
Table 4　Trend in first submit on CPA.

実施日	誤答人数（人）		提出人数（人）		エラー件数（件）		
	1 問目	2 問目	1 問目	2 問目	コンパイル	インデント	クラスの形式的記述
10 月 4 日	14	—	59	—	1	13	2
10 月 6 日	18	—	62	—	14	14	17
10 月 13 日	13	13	60	60	8	5	22
10 月 18 日	6	3	58	58	1	1	9
10 月 20 日	12	8	59	59	7	4	12
10 月 25 日	20	11	57	58	9	11	17
10 月 27 日	19	—	54	—	10	11	10
11 月 8 日	16	10	55	57	3	1	22
11 月 10 日	13	—	55	—	2	3	11
11 月 15 日	6	—	54	—	1	1	4
11 月 24 日	15	—	53	—	3	2	12
11 月 29 日	20	—	48	—	4	3	18
12 月 1 日	21	—	55	—	3	3	18
12 月 6 日	18	—	54	—	7	4	11
合計	211	45	783	292	73	76	185
	256		1,075			334	
	提出人数に対する誤答人数比：0.238						

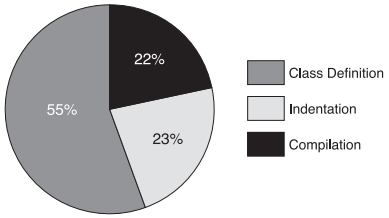

図 8　初回提出における学習者の誤り
Fig. 8　Mistakes of Learner at first submit.

5.3　提出回数の度数分布

本節では，学習者の提出解答が正解となるまでの回数から示される結果について論ずる．学習者は，提示された問題に対して，複数回提出を行うことができる．その解答が誤答である場合，学習者は自身のソースコードを見直し，再度提出することがほとんどである．また，提出を行っていない学習者に対しては，教授者が提出を促している．

CPA における平均提出回数を**表 5** に示す．平均提出回数とは，各実施日の問題ごとに学

© 2011 Information Processing Society of Japan

表 5　CPA における平均提出回数
Table 5　Average of times of submit at CPA.

実施日	問題数	平均提出回数 1 問目	2 問目
10 月 4 日	1	1.28	—
10 月 6 日	1	*1.38	—
10 月 13 日	2	1.34	1.33
10 月 18 日	2	1.22	1.05
10 月 20 日	2	1.16	1.25
10 月 25 日	2	1.29	*1.59
10 月 27 日	1	*1.46	—
11 月 8 日	2	*1.5	*1.43
11 月 10 日	1	1.32	—
11 月 15 日	1	1.11	—
11 月 24 日	1	*1.5	—
11 月 29 日	1	*1.58	—
12 月 1 日	1	*1.54	—
12 月 6 日	1	*1.47	—
総平均提出回数		1.36	

習者が提出した回数を，学習者の全体数で平均したものである．また，総平均提出回数とは，14 の実施日すべての提出回数を平均したものである．また，表中のアスタリスク（*）は，総平均提出回数より大きい平均提出回数を示している．

表 5 より，提示された問題に対して学習者が提出を行う回数は，平均して 1.36 回であることが示されている．この平均より顕著に大きい平均提出回数を示す問題についての度数分布表を作成し，それをグラフ化したものを**図 9**（10 月 25 日），**図 10**（10 月 27 日），**図 11**（11 月 8 日），**図 12**（11 月 24 日，11 月 29 日，12 月 1 日），**図 13**（12 月 6 日）に示す．度数分布表を作成した実施日と出題問題は**表 6** である．また，すべての問題の題意はクラスの形式的記述である．

これら度数分布表とグラフから，次のような結果が読み取れる．

10 月 25 日　1 問目と比較して，2 問目は，2 回以上の提出を行っている学習者が増えており，平均提出回数も大きい．また，5 回以上提出を試みている学習者も増えている．10 月 25 日は，主に引数と返却値についての授業であり，2 問目である A6_6.java は，`getMessage` というメソッドを含んでいる．この問題では，授業の資料上初めて返却値を持ち複数の引数をとるメソッドについて学んでいるため，混乱が大きかったと推測される．エラー内容としては，メソッドのつづりエラーをはじめ，`return` 指定の忘れ，

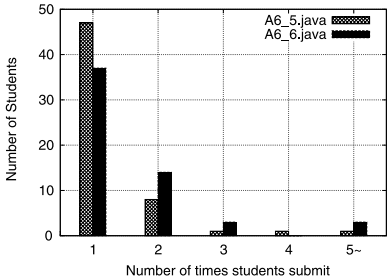

図 9　10 月 25 日の提出回数の度数分布
Fig. 9　Frequency of times of submit at 25 Oct.

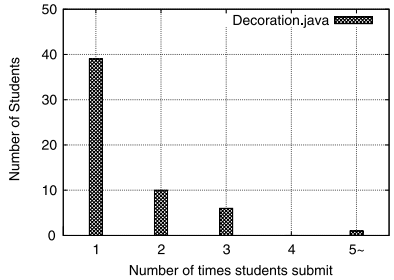

図 10　10 月 27 日の提出回数の度数分布
Fig. 10　Frequency of times of submit at 27 Oct.

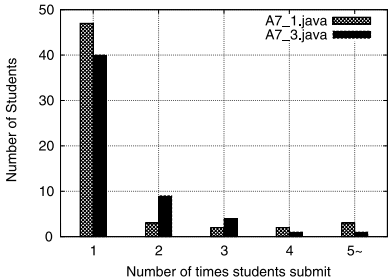

図 11　11 月 8 日の提出回数の度数分布
Fig. 11　Frequency of times of submit at 8 Nov.

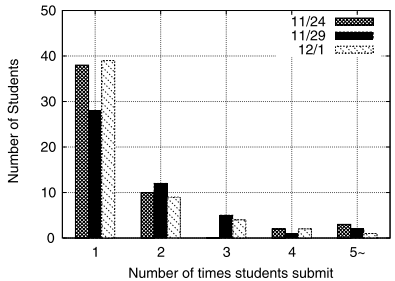

図 12　Janken.java の提出回数の度数分布
Fig. 12　Frequency of times of submit for Janken.java.

引数の順序エラーなど多岐にわたるが，多くは返却値，引数に関したものであったことからも，このことが裏付けられると思われる．

10 月 27 日　総平均提出回数と比較して，1.46 と比較的大きい平均提出回数を示している．Decoration.java は，授業の資料上初めてメソッドを 3 つ含むクラスのクラスの形式的記述を求めており，記述量の増加がエラーを誘発していると考えられる．また，複数の引数，返却値を各メソッドで扱うため，それに関する混乱も大きいと思われる．

11 月 8 日　1 問目，2 問目ともに平均提出回数が比較的大きい回数を示している．また，ばらつきはあるものの 2 回以上提出を試みている学習者が目立つ．この日提示された問題は，1 問目は `printWeatherInformation` メソッド，2 問目は `getWeatherInformation` メ

© 2011 Information Processing Society of Japan

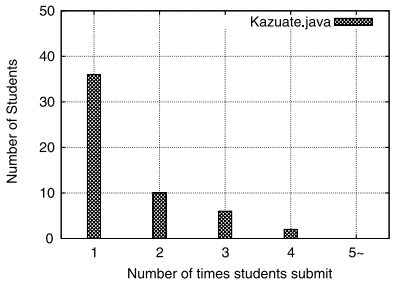

図 13　12 月 6 日の提出回数の度数分布
Fig. 13　Frequency of times of submit at 6 Dec.

表 6　実施日と出題問題
Table 6　Date and Questions.

実施日	出題問題	
	1 問目	2 問目
10 月 25 日	A6_5.java	A6_6.java
10 月 27 日	Decoration.java	—
11 月 8 日	A7_1.java	A7_3.java
11 月 24 日	Janken.java	—
11 月 29 日	Janken.java	—
12 月 1 日	Janken.java	—
12 月 6 日	Kazuate.java	—

ソッドのクラスの形式的記述を求めており，また双方のクラスには `temperature` という変数名を含んでいる．これらのつづりエラーが最も多く検出されていることからみても，つづりが複雑な，あるいは難読な英単語を含む問題は提出回数が増えると考えられる．

11 月 24 日，11 月 29 日，12 月 1 日　これらの実施日に提示された問題は共通して Janken.java であり，これは徐々にメソッドを追加していき，そのつどクラスの形式的記述を行わせ提出を求めたものである．最終的な解答は図 7 となる．この Janken.java は，ジャンケンゲームを表現するクラスで，これまで学んできたメソッド定義，引数，返却値などの項目がすべて含まれているいわば総まとめの役割を持つ問題である．平均提出回数は総平均提出回数を大幅に上回っている．また，図 12 からも 2 回以上提出している学習者が多いことがうかがえる．エラーの内訳としては，コンパイルエラーのほか，メソッドのつづりミスによるクラスの形式的記述のエラーが多いほか，引数の順序や指定のミスなどがみられた．

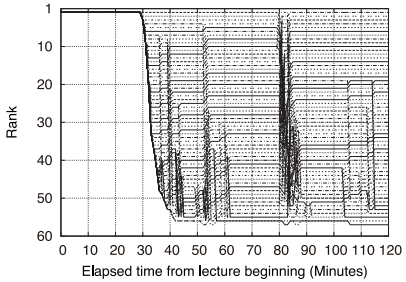

図 14　CPA の 10 月 25 日における進行度の推移グラフ
Fig. 14　Change graph of progress-rank at 25 Oct.

12 月 6 日　1.47（表 5）と，比較的大きい平均提出回数を示している．1 問目である Kazuate.java には，`getResultMessage` というメソッドが含まれており，このメソッドの定義ミスが多くみられた．このメソッドは `int` の引数を 2 つとるため，特に引数の指定とメソッド名のつづりミスが多数を占めている．

5.4　進行度の推移

本節では，進行度の推移について述べる．本システムでは，授業中に学習者の提出状況や解答状況などを視覚的に把握するために進行度を算出し利用している．**図 14** に示すのは，CPA の 10 月 25 日における進行度の推移を，**図 15** に示すのは，CPA の 10 月 18 日の授業における進行度の推移をそれぞれグラフ化したものである．

また，**図 16** に示すのは，CPA の 10 月 25 日における進行度に，途中点を加味した推移をグラフ化したものである．途中点とは，完全正解数の個数ではなく，コンパイル，インデント，クラスの形式的記述，ユニットテストの 4 つの評価項目のうち，一部正解している場合も加味した評価をさす．ここでは，4.2.2 項で述べた進行度のアルゴリズムに，完全正解数が等しい場合には，評価項目の正解項目数が多い学習者を上位に，さらにそれが等しい場合には提出時間の早い学習者を上位とするよう変更を加えたものを使用している．

縦軸 Rank は進行度の順位を示し，横軸は，授業を開始してからの経過時間を示している．なお，これらの図では，進行度は 1〜100 に正規化する以前の値を用いている．

© 2011 Information Processing Society of Japan

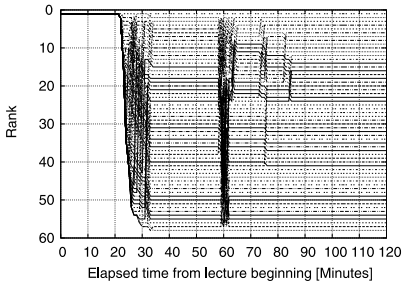

図 15　CPA の 10 月 18 日における進行度の推移グラフ
Fig. 15　Change graph of progress-rank at 18 Oct.

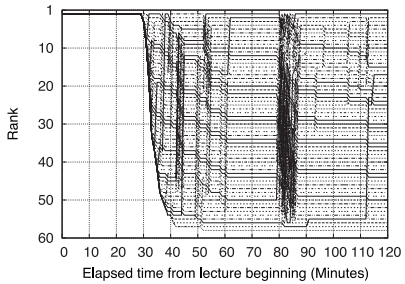

図 16　CPA の 10 月 25 日における進行度の推移グラフ（途中点を加味した場合）
Fig. 16　Change graph of progress-rank with partial points at 25 Oct.

図 14 において，授業開始後 30 分，および 80 分前後に大きな変動が確認できる．これで，30 分および 80 分前後にそれぞれ問題が提示されたことが分かる．また，推移の状況から，順位は問題提示後 10 分前後で安定してくることが分かる．多くの学習者は，課題を 10 分程度で解き，提出していると考えられる．

図 15 においても，同様に，30 分前後および 60 分前後に大きな変動が確認できる．10 月 18 日に関しては，表 5 より，平均提出回数が少なく，表 4 よりエラー件数も少ないことから，図 14 と比較して，全体的な進行度の変動がみられない．

図 16 では，進行度に途中点を付加することで，さらに推移を詳細に示している．途中点を加味することによって，図 14 よりもさらに詳細に進行度の変動を確認できる．30 分前後の変動から，提出は比較的早期に行うが解答は間違えており，その後徐々に順位が下がっていく学習者を確認することもできる．

6. 考　　察

先の検証から，本章では本システムの有用性について考察する．

6.1　個人提出履歴の追跡について

本システムでは，個人提出履歴が分析可能な状態で適切に収集されている．また，検証によって提出履歴から学習者の試行錯誤が詳細に読み取れることが確認できた．アンケート提出などの最終解答では得られなかった試行錯誤の一端を測るには十分であると考えられる．

また，出題された問題に関する提出動向と提出履歴を照らし合わせることで，提出回数が多い学習者における動向を測ることができた．特に，問題に対する平均提出回数と，提出された解答における評価項目別エラー履歴の傾向から，「試行錯誤しているがなかなか正解できない学習者」の前兆を予測できる可能性が示唆される．エラー履歴の傾向を蓄積し，平均提出回数を閾値として学習者の解答を分類することで，将来的には前述の学習者を早期に発見することができるようになると考えられる．

6.2　初回提出の動向について

初回提出は，学習者が出題された問題に対して初めて提出をする解答である．検証によって，この初回提出が誤答である学習者が 2 割程度存在することが確認された．また，表 4 における誤答人数とコンパイルエラー数から，初回提出が誤答である 2 割の学習者のうち，その 3 割の原因がコンパイルを含むものであった．

今回行った実施検証では，学習者に対し，最低限コンパイルを完了させた状態の解答を提出するようあらかじめ指示をしていた．それにもかかわらず，初回誤答のうち 3 割もの学習者が初回提出時にコンパイルが間違っていると判断されている．このことから，これらの学習者に対して教員の指導が行き届いていない可能性が示唆される．

また，残りの初回誤答の 7 割程度の学習者に関しては，初回提出ではコンパイルが可能な状態であった．すなわち，これらの学習者は，インデントやクラスの形式的記述といった，自身では正確な正誤を判断できない項目に関して何らかの間違いを犯していることが分かる．

初回提出において，コンパイルができたかできなかったかにかかわらず，何らかの間違いを犯しているこれら学習者に対しては，講義を進行させるにあたり，適切な支援が必要である．従来であればこれら学習者は見過ごされていた可能性が高く，また最終解答を綿密にチェックしたとしても，そのフィードバックは次回以降の授業に持ち越されてしまう．本システムによって，これら 2 割の学習者は位置情報をともなって適切に把握されているため，見過ごすことはより少なくなると考えられる．また，進行度にも反映されるため，リアルタイムで授業にフィードバックすることが可能であると考えられる．

ただし，学習者が，間違いが含まれていることを認識していながら，それを検証させるためにシステムに意図的に提出を行う「消極的提出」を行っている可能性は否定できない．特に，指示の行き届いていない学習者に関しては，その可能性もあると考えられる．しかし，消極的提出を行う学習者も，自身では検証できない項目（インデントやクラスの形式的記述）をシステムに検証させていることに変わりはない．そのため，「解いてから提出」している学習者も，消極的な提出を行っている学習者も，本システムの 1 つの目的である「試行錯誤を収集する」という観点からみれば，あえて区別をする必要はないと考えられる．同時

© 2011 Information Processing Society of Japan

に，本システムは成績付けや順位づけではなく，失敗を経験させることを目的としており，意図的であるかどうかにかかわらず，学習者がつねにフィードバックを受け取れる環境を提供するためのシステムであるため，誤答を提出することによるいかなるペナルティも設けていない．また，これらの学習者の多くは，その後の試行錯誤によって完全正解に到達していることが確認できており，本システムによって教授者が適切に指導を行えたものと考える．ただし，指導が行き届いていない学習者に対しては，講義中に適切な再指示が必要になるだろう．

6.3　提出回数の度数分布について

度数分布から，授業の内容によって提出回数が変動することが確認できた．特に，引数や返却値，あるいはそれらの複数使用など，新しい概念の出現にともなって学習者は多くの提出を行うことが確認できた．

間違いの傾向として，授業資料において複雑な英単語などの使用でつづりのエラーが多く確認できたことで，つづりの指導の徹底，あるいは授業資料の見直しなどのフィードバックが行えると考えられる．また，新しい概念の出現時には，問題をこまめに提示し，段階的な理解を促すなどの授業改善が可能になると考える．

6.4　進行度の推移について

検証によって，進行度が提出状況や解答状況を適切に表現できていることが確認できた．また，進行度に途中点を加味することで，さらに詳細な学習者の学習状況を把握することができることが確認できた．このことから，システムが収集した膨大な情報を単純な方法で一べつ可能であると考えられる．また，進行度の重み付けのアルゴリズムは柔軟に変更可能であることから，より多角的な分析を講義中に行うことができるようになると考えられる．

7. むすび

本稿では，プログラミング教育において，学習者の実践と失敗という過程を収集し，分析可能な授業支援システム IDISS を提案し，その仕様を述べた，そして，その仕様に基づいてシステムを開発し，実際の授業に導入することで有用性の検証を行った．

本システムにおいては，学習者個人の提出履歴が適切に収集されていた．また，学習者の初回提出の動向と提出回数の度数分布を検証することによって，従来は見過ごしていた可能性の高い学習者を把握することができることが確認できた．

そして，進行度の推移が学習者の取り組み状況を適切に反映していることが確認できたことで，授業中において，学習者の課題への取り組み状況が逐一把握でき，リアルタイムに授業へフィードバックすることができると考えられる．

本稿の検証によって，プログラミング教育において 2 割の学習者は初回提出において何らかの間違いを犯しており，それらの学習者に対しては迅速で適切な支援が必要であろうことが示唆された．

少数の教授者が多数の学習者に対し授業を行う一斉学習においては，すべての学習者に指導を行き届かせることは困難であり，また教授者もその事実を把握できない，あるいは人的リソースの問題，授業時間などの制約から把握していても黙認しているケースが多々あると考えられる．本システムの導入は，そういった学習者を把握し，適切に指導を行うことで全体の底上げを行う一助となりうるだろう．

今後は，未実装のユニットテストを含めた評価項目カテゴリの拡充を図り，より詳細な分析を行うことで，充実した教授者支援を行えるシステムとすることが望まれる．また，収集した情報から統計的解析を行い，授業改善の指標として活用していきたい．

また，本システムはクラウド環境的に利用することができるため，Web サービスとしての公開が比較的容易である．また，クライアントアプリケーションも，JavaVM を搭載してさえいれば多くの環境で稼働させることができる．インデントやクラスの形式的記述に関する評価部，クライアントアプリケーションなどは汎用的に利用できると考えられるので，プログラミング入門教育を行っている多くの教授者が利用できるよう，利用条件を含めた具体的な公開方法に関して積極的に検討してきたい．

参　考　文　献

1) Martin, R.C.: *Clean Code: A Handbook of Agile Software Craftsmanship* (*Robert C. Martin Series*), 1st edition, Prentice Hall (2008).
2) Miara, R.J., Musselman, J.A., Navarro, J.A. and Shneiderman, B.: Program indentation and comprehensibility, *Comm. ACM*, Vol.26, pp.861–867 (1983).
3) 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).
4) 斐品正照，徳岡健一，河村一樹：構造化チャートを用いたアルゴリズム学習支援システム，情報処理学会論文誌，Vol.45, No.10, pp.2454–2467 (2004).
5) 知見邦彦，櫨山淳雄，宮寺庸造：失敗知識を利用したプログラミング学習環境の構築，電子情報通信学会論文誌 D-I，情報・システム，I-情報処理，Vol.88, No.1, pp.66–75 (2005).
6) 小西達裕，鈴木浩之，伊東幸宏：プログラミング教育における教師支援のためのプログラム評価機構，電子情報通信学会論文誌 D-I，情報・システム，I-情報処理，Vol.83,

© 2011 Information Processing Society of Japan

No.6, pp.682–692 (2000).
7) Spacco, J., Hovemeyer, D. and Pugh, W.: An Eclipse-based course project snapshot and submission system, *Proc. 2004 OOPSLA Workshop on Eclipse Technology eXchange* (*eclipse '04*), pp.52–56, ACM, New York, NY, USA (2004).
8) Jadud, M.C.: Methods and tools for exploring novice compilation behaviour, *Proc. 2nd International Workshop on Computing Education Research* (*ICER '06*), pp.73–84, ACM, New York, NY, USA (2006).
9) 玉田春昭，荻野晃大，上田博唯：アシスタントロボットを用いたプログラミング教育支援システムの構築，映像情報メディア学会技術報告，Vol.34, No.25, pp.143–148 (2010).
10) 倉澤邦美，鈴木恵介，飯島正也，横山節雄，宮寺庸造：プログラミング演習における一斉指導のための学習状況把握支援システムの開発，電子情報通信学会技術研究報告 ET，教育工学，Vol.104, No.703, pp.19–24 (2005).
11) Edwards, S.H.: Improving student performance by evaluating how well students test their own programs, *J. Educ. Resour. Comput.*, Vol.3 (2003).
12) Edwards, S.: Web-CAT Wiki, Stephen Edwards（オンライン），入手先⟨http://web-cat.org/WCWiki/WebCatWiki⟩（参照 2011-07-03）.
13) Oracle: Java SE Desktop Technologies – Java Web Start Technology, Oracle（オンライン），入手先⟨http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136112.html⟩（参照 2011-03-25）.
14) Oracle: JavaFX Rich Internet Applications Development, Oracle（オンライン），入手先⟨http://javafx.com/⟩（参照 2011-07-03）.

(平成 23 年 3 月 29 日受付)
(平成 23 年 9 月 12 日採録)

長谷川　伸（正会員）
1987 年生．2009 年東京電機大学情報環境学部卒業．2011 年東京電機大学大学院情報環境学研究科修了．同年ニフティ株式会社に入社．ソフトウェアの設計・開発，Web サービス開発や情報アーキテクチャに興味を持つ．

松田　承一
1988 年生．2010 年東京電機大学情報環境学部卒業．現在，東京電機大学大学院情報環境学研究科修士課程在学中．ソフトウェア設計および開発に興味を持つ．

高野　辰之（学生会員）
1983 年生．2007 年東京電機大学情報環境学部情報環境デザイン学科卒業．2009 年東京電機大学大学院情報環境学研究科情報環境デザイン学専攻修了．現在，東京電機大学大学院先端科学技術研究科情報通信メディア工学専攻博士課程（後期）．ソフトウェアの設計・開発，教育工学に興味を持つ．電子情報通信学会，日本教育工学会各会員．

宮川　　治（正会員）
1964 年生．2001 年東京電機大学大学院工学研究科情報通信工学専攻博士後期課程修了．博士（工学）．現在，東京電機大学情報環境学部情報環境学科准教授．ソフトウェアの設計・信頼性，教育支援システムに興味を持つ．電子情報通信学会，日本教育心理学会各会員．

© 2011 Information Processing Society of Japan