ASPEN: 自動データ収集機能を備えた Webベースプログラミング学習システム

菅谷 みどり‡　　若森 拓馬†　　倉光 君郎‡

概要

情報科学の教育の一つであるプログラミング演習において，受講者のコーディングの実態は十分把握されていない．本研究では，学生のコーディングの実態の把握と，自習支援のためのプラットフォームとしてASPENを提案する．ASPENは，Webベースで開発を支援し，自動的に開発データを収集する．論文ではASPENの仕組みと，実際にASPENを授業に導入し，データを収集した結果の一部を報告する．また，得られたデータに基づき学生の開発工程を調査した結果について報告する．

ASPEN: A Web-based Programming Learning with Automatic Data Collecting System

Midori Sugaya‡　Takuma Wakamori†　Kimio Kuramitsu‡

Abstract

Within a programming exercise course, which is one of the educations of computer science, the actual coding process was not fully figured out. In this study, we propose ASPEN, which provides Web-based self-studying environment for students. We try to understand the actual coding process of students through this system, since ASPEN collects the data from students who use this development environment automatically. In this paper, we show the mechanism of ASPEN, and some results that are collected from students who attends the beginner's Java exercise course. Based on the data, we analyze the tendency of the development style of these students.

1. はじめに

多様な情報技術が社会に浸透する中，実用的なプログラミング能力を持つ人材の育成が急務とされている．こうした人材の育成のために，大学や専門学校などの教育機関ではプログラミング演習のカリキュラムを設け，専門教育を行っている.しかしコーディングの工程については，教員の理解と学生の現実に差があることが報告されている[1] 1.

教員は，学習者が提出するソースコード以外に,その開発力を測る手段がなく,学習者が具体的にどのように開発を進めているかの詳細を把握することは困難である．Jamesらはプログラミング初学者が実際に行っている開発の情報は少なく，検証が十分に行われていないこと問題とし，データ収集のためのプラットフォームを提案した[1].

開発環境からデータを収集し，初学者のプログラミングの進捗の把握や，教育用途に用いる試みとしては，Jamesらの統合開発環境へのロガーの組込みや[2]，Hackystatなど様々な手法が提案されている[3]．しかし,これらはEclipseなどの開発環境へのプラグインとして提供されていたり，Java VMの環境整備が必要となるなど，初学者がPCなどにインストールするには敷居が高い問題がある．その結果，実際に初学者が演習課題を行う場所が，演習室に限られてしまい,自習に制限を加えてしまう問題もある．

こうした問題に対処するために，我々はWebベースのプログラミング環境であるASPENを提案する．ASPENは,ブラウザ以外の特別な開発環境を必要とせず，インターネットに接続された環境であれば，どこでも利用することが可能なプログラムの開発環境であり,初学者も気軽に利用できる.また，ASPENでは学習者がどのようにプログラミングを行っているか実態を把握するため，プログラミングに関するログを自動的に記録し，データを提示できるものとした．データは，プログラミングの開発に関する情報である，コードの作成行数，コンパイルのエラー，警告,手戻り（コードの削除），コードのコピーなどの情報とした．これらの情報は，ソフト

† 横浜国立大学 Yokohama National Universit
‡ 横浜国立大学，科学技術振興機構/CREST. Science and Technology Agency/CREST

1 学生は，徹底的にテストを行い，頻繁にコンパイルを行い，段階的にコードを開発し，問題となる部分を修正したと主張しているのに対して,教員は，学生はこのような開発を行っていないと考えている[1].

ウェア工学で利用されている開発者のためのプロセス改善に用いられる PSP(Personal Software Process) を一部参考にして取り入れた.

また,学習者の自習を支援するための仕組みとして, ASPEN は, 教員が示したサンプルプログラムや, 他人のソースコードを閲覧できる機能を提供する. プログラミングの自習においては参考となるサンプルコードが重要であるという調査結果は, [7]示されていたことによる 2.

本論文では, これらの特徴をもつ ASPEN の設計, 実装について述べ, 実際に授業に適用した際の実践の結果について報告する. 報告結果では, ASPEN から収集したデータにより, 学習者の開発の実態を把握することに効果があったことを示した.

本論文の構成は以下のとおりである. 第 2 節では, ASPEN の設計方針とその仕組みについて述べる. 第 3 節では, 実際に ASPEN を授業に適用し, データを収集した内容の分析と, 議論を示し, 第 4 節にて関連研究, 第 5 節で本論文を総括する.

2. ASPEN の提案

2.1 設計方針

ASPEN は,プログラミング初学者のソースコード記述の実態を正確に把握することを目的とし Web ベースのプログラミング環境を提供する. 主な設計方針を以下の 3 点にまとめた.

1. 開発環境のインストールなどが不要で, 場所を問わず利用可能であるものとする.
2. 利用者の開発の実態の把握のため,利用者の開発履歴を収集する.
3. ソースコードの検索機能を設けることで, サンプルとして提示したソースコードや, 他人のソースコードを閲覧可能とする.

1. については初学者にとって、高度な開発環境をセットアップすることは困難であることから, ASPEN は Web 環境として提供することで対応するものとした.また, コンパイルの過程を,Web ブラウザと Web サーバ側に分けた. これは, サーバ側でコード変換のオーバーヘッドを減らし, ブラウザ側で実行を行うためである 2.については, 開発履歴データベースを構築し, 利用者のデータを逐一保存できるものとした. 3. については, 利用者がアカウントごとに作成したファイルは全てデータベース化するものとした. 利用者は,検索キーワードを入力して, データベースにアクセスすることで, 他人が作成したソースコードを閲覧可能となる.

2.2 システム構成

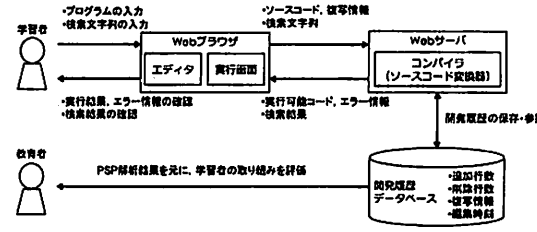

図 1 ASPEN のアーキテクチャ

図 1 に ASPEN のアーキテクチャを示した. 主なコンポーネントの詳細を以下に示す.

- Web ブラウザ

学習がプログラミングを行うための環境を提供する. 学習者は, インターネットに接続された Web ブラウザを起動し, その上で編集可能なテキストエディタを用いて, プログラムの作成, 更新, 保存を行う. Web ブラウザからは, Web サーバに対して, ソースコードや, 複写情報（ソースコードのコピー・ペーストに関する情報）, 検索文字列を送信する. なお複写情報があるかどうかの判断手法としては, テキストエディタ上で, テキストの変更イベントを取得し, 一度の変更に対して 2 文字以上の文字の変化があった際に, これを複写とみなすものとした.

- Web サーバ

ブラウザからソースコードを受け取り,ソースコードをパースし, 実行コードを生成するためのプログラムの変換や, 保存を行う.サーバは, クライアント側から受信したソースコードをブラウザで実行可能なコードに変換するコンパイラ（ソースコード変換器）をもつ. コンパイラは, クライアント側から送られてきたソースコードを, サーバ上で実行可能コードへと変換する. 変換したコードおよびコンパイル結果のエラー情報を, クライアント側の Web ブラウザに送信する. 同時に, クライアント側から受信した情報を開発履歴データベースへ保存する.

- 開発履歴データベース

学習者のプログラムの開発履歴を保存する. 学習者の記述したすべてのソースコードは, 開発履歴として, サーバ側の開発履歴データベースに保存される. ASPEN では, 他人のコードが参照可能であるが, これは他人のコー

2 サンプルプログラムの重要性は, 2012 年に実施した Java プログラミング入門アンケート結果でも, 80% 以上の学生が指摘した

ドがコピー可能であることも意味する．我々は，実際にどのようなコードがコピーされ，開発に利用されるのかを観測するために，コピー情報も開発履歴の一部として保存できるものとした．

ASPEN の基本構成はサーバクライアント形式とした．理由としては(1) ユーザがどのような環境でプログラムを編集しても，ソースコードや作成時間などのデータを一元管理ができる．(2) 講義資料や例題などを表示する Web サーバとの連携がとりやすい，といった理由である．

2.3 ASPEN の学習機能

ASPEN を利用したプログラム開発の手順およびインターフェイスの設計と実装を以下に述べる．

1） ログイン
教員は，学習者の ASPEN アカウントを予め作成しておく．学習者は，教員から教わった ID とパスワードを用いて，ASPEN へログインを行う．

2） コードの管理
ログイン後は，学習者ホーム画面に移動する（図 2）．学習者（ここでは，d11gd186 とする）はホーム画面から，プログラムの新規作成，削除，編集するプログラムの選択などの操作を行うことができる．コード検索，演習資料の表示などの操作は，画面上部のナビゲーションバーを利用して行うことができる．

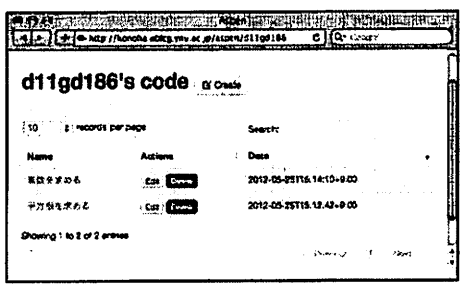

図 2 学習者ホーム画面

3） プログラムの編集，実行
学習者は，画面左側のエディタにソースコードを入力し，「Save」ボタンを押すことで，ソースコードの編集，保存を行うことができる．画面右側は，実行結果を表示する画面となっており，「Run」ボタンを押すと，ソースコードが実行され，結果が表示される．

学習者は，ソースコードと実行結果を照らし合わせながら，コーディングを行うことができる．また，「Check」ボタンを押すと，プログラムが型検査され，型エラーのある行がハイライトされ，エラー情報が表示される．学習者は，エラー情報を確認しながら，インクリメンタルにプログラムを開発することができる．

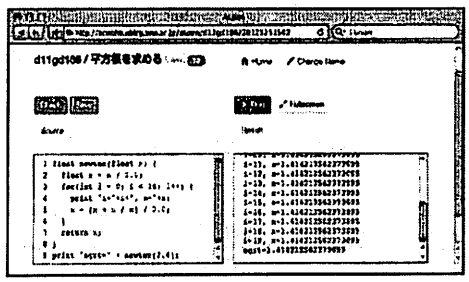

図 3 コード編集，実行画面

4） 他の学習者のプログラムの検索
学習者は，理解に行き詰まった際に，ASPEN の検索機能を利用して，他人の作成したプログラムを検索し，その内容を閲覧することができる．今回は，学習者がどのような共有方法を望んでいるかを調査する目的も兼ね，検索キーワードを学習者に入力してもらう形式をとった．学習者は，ソースコード名，ID，ソースコードの全文検索の結果から，友人のコードを探すことができる．

以上の手順により，学習者は Web ブラウザのインストールされた環境から，他人のプログラムのソースコードを参考にして，プログラミング演習に取り組むことができる．

また教員は，学習者と同じ手順で，学習者向けにサンプルプログラムを公開することができる．

3. ASPEN を用いたデータ収集

本節では，特に ASPEN を用いたデータの収集と，開発実態の把握，今後の支援の方向について述べる．

3.1 開発履歴の収集

開発履歴の収集にあたり，ASPEN では PSP (Personal Software Process) のデータを参考にした．PSP は個人のプログラミングのプロセスを向上させることを目的として，Humphrey が提唱した技法である[4]．本来は，個人が生産性やエラー率などを記録し，その作業を追跡することで，高品質のソフトウェアの開発に役立てるためのプロセス改善の方

法である.

今回 ASPEN でのデータ収集の目的は，プログラミング演習における，学習者の演習（自習）の実態をできるだけ正確に把握することである．実態の把握のために必要なデータは，ソフトウェア開発の現場で，実際にエンジニアの開発工程の把握に利用されている項目が望ましい．中でも最終的に改善手法とも結びつけが期待される PSP を用いることで演習の支援環境についての示唆もある．これらの理由により PSP を参考にデータ収集項目を定めた．

表 1 ASPEN が収集する開発履歴

アクション	保存する情報
サインイン	学習者名(ID)
サインアウト	学習者名(ID)
コード閲覧	ID，コード名
コード検索	ID，検索文字列
コード実行	ID，コード名
コード編集	ID，コード名，コード本体，複写情報
コード作成	ID，コード名
コード削除	ID，コード名

表 1 に PSP に従い，ASPEN で収集したデータの一覧を示した．学生は,利用者はサインインしないと，自分のホームページでファイルの作成やプログラムの実行などができないため，サインインの値は利用者の利用回数を示すことができる．また，利用者のコード閲覧回数，検索回数，実行，編集，作成，削除については，その都度 ASPEN 上のデータベースに記録されるものとした．

表 2 授業概要

科目名	Java プログラミング演習
対象学年，学生	早稲田大学理工学部電子生命学科，1年
目標	Java プログラミング技術の習得，オブジェクト指向言語理解
期間，回数	2012 年 4 月～，全 1 5 回
授業時間	木曜日 3 限，90 分
受講人数	35 人
授業形態	演習室のコンピュータを利用，授業の前半に講義，後半に演習を行う．演習課題を次回の授業開始時までに提出する．

3.2 実践報告

我々は，早稲田大学，および横浜国立大学の二つの大学で，ASPEN を利用した授業を行った．今回の実践報告では主に，早稲田大学での実践内容を報告する.授業の概要を表 2 に示した．特に今回は実践報告として，プログラミング演習の授業のうち 2 回の演習を対象に，開発履歴の収集と解析を行った結果を報告する．2 回の演習のうち 1 回は条件分岐（if-else，等価/関係/論理演算子，case 文），他方は繰り返し(for，while 文，インクリメンタル演算子，多重ループ)の授業である．

授業の進め方では，前半で基礎的な概念の説明を行い，後半に演習を行うものとした．演習では，教員が例題プログラムを学生の前で実際に記述し，コンパイル，実行までの実演を行う．その後,学生にも同様の手順でプログラムを作成するように指示をする形で進めた．

このとき学生は，教員のコードを閲覧し，コピーして利用することもできる．提出課題の多くは例題プログラムを応用したものであるが，そのままでは正しい回答とはならず，例題のプログラムの動作を理解し,改変することが必要である．演習中に回答できなかった課題は，次の演習まで自習により対応するものとした．今回論文で対象とする 2 回の演習では，計 11 問の課題を出した．

3.2.1 収集データ項目

演習課題は ASPEN 上で提出とした．提出した課題については，下記の項目を集計した．

- コードの作成行数
- コンパイルのエラー，警告数
- 手戻り（コードの削除）数
- コードのコピー数

PSP でも，コードの生産性（作成行数/時間，LOC） や，エラー警告数，手戻り数などを記録するが，ASPEN ではこれらの値を自動的に収集することが特徴である．

尚，コードのコピーはいつ，誰がどのようなコピーを行ったかの情報は全て開発履歴データベースに保存するが，このことは事前に学生に伝え，丸写しは評価されないことは周知を行っている．

3.2.2 グループ分類での傾向

演習課題の採点結果に従って学習者のグループを上から順番に 6 つに分けた．さらに，6 つに分けたグループごとに，PSP のデータ収集項目＋コピー数に従い，1 回答あたりの結果を図 4 にまとめた．

図 4 のグラフから最も成績が良かった（全

問正解）のグループAにおいては,コピー数が最も少なく，2番目に結果が良かった B と，最も結果が悪い E, F のグループではコピーが他よりも多かったが，特に E, F のグループではこの傾向が強く，1 回答あたり 40-50 行のコピーが行われていた．1回答あたりのコード行数，エラー数，手戻り行数には，成績上位グループと下位グループに数的には大きな差は無かった．

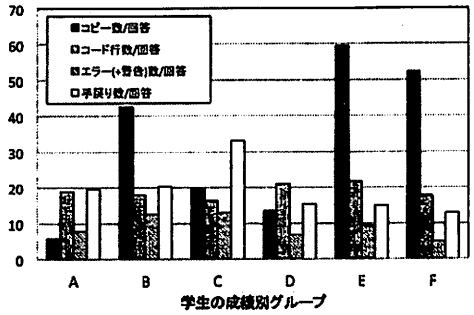

図 4 演習問題の採点結果

3.2.3 個人ごとのデータ傾向

PSP では，個人ごとの開発工程を詳細に分析する．これにならい，A グループ，F グループの学習者を 2 名抽出し，開発履歴のデータから,開発の経緯をもとに分析を行った.対象のプログラムは階乗計算を for ループにより解く課題であり，その結果を時間ごとのデータを示した(図 5, 図 6). 図 5 の A グループの学生は, 15:36 に演習を開始し，約 1 時間かけ課題を解いた.コピーは行っておらず,行数は，10 分あたり 13 行程度でほぼ線形に増大している．コンパイル，コードの削除は逐次行われており，最終的には数十行程度のコードとなった.コード生産性(コード行数/時間) は高くはないが，コード記述とコンパイル，削除を繰り返しながら正しい答えを導いている様子が見て取れる．A グループの学生の全体の傾向も，同様に線形にコード量が増え，手戻りは多くとも，エラーも多くなく，収束する傾向にあった．

次に正答率の低い F グループの学生の開発履歴を図 6 に示した.この学生はコードを 10 行程度コピーした後，コード量は増えず，手戻り行数やエラーが幾つか出た後正答に至らずに 25 分程度で終了している．コピーし，改変してもエラーが収束せず，あきらめている様子が見てとれる．F の同じ学生で正答した場合の経緯を図 7 に示した.コピー後に，コードを改変し，20 行前後記述を追加し正解に至っている．この場合も手戻りやエラーが無いことから,理解しないまま動作させている可能性もあると考えられる．全体として，不正解者のコードは，コピーを頻繁に行い，改変後のエラーが収束せず終了していたり，そもそも全体としてコードの改変などの試行錯誤が少ないことが指摘できる．

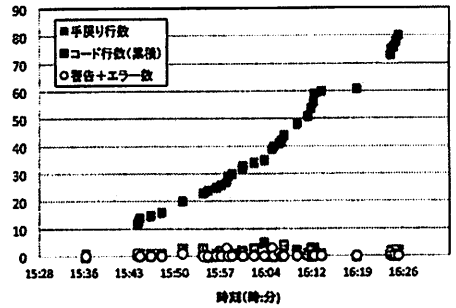

図 5 グループ A （正答）開発履歴

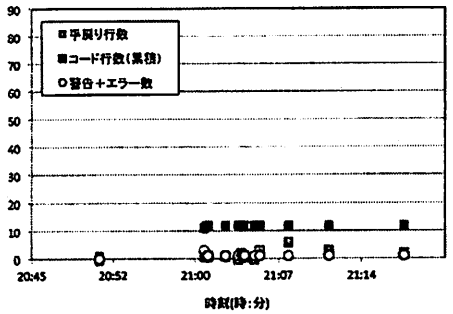

図 6 グループ F （不正解）開発履歴

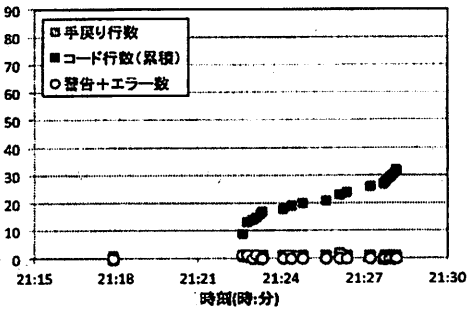

図 7 グループ F （不正解）開発履歴

3.2.4 議論

今回，正答率が高いグループの傾向として，コード行数が線形に伸び，エラー率が低いもしくは収束する傾向が見られた．また，正答

率が低いグループでは，コピーが行われているが，その後理解ができない場合はコード行数が延びず，エラーが収束しない，もしくは行数が延びずにあまり活発に修正がなされていないなどの傾向が読み取れた．今回は演習数も少なかったため，これらを一般化することは時期尚早であるが，これらのことから，正答率が高い学生と，低い学生には異なる傾向があることが分かった．また，こうした開発の傾向について PSP で取得するデータにより，ある程度把握できることが分かった．

今後は,開発の進捗と理解度の関係についてより調査を進めた上で，今回の E, F などの正答率が低い学生については傾向分析，パターン化による早期の自動検知や支援を検討する.さらに，今回 ASPEN ではコピーを是とした開発支援システムとして提案したが，正答率が低い学生に多く見られたことから，こうした学生に対する学習理解の促進に真に有効であるのかについて,今後より詳細な調査および議論が必要であると考える．

4. 関連研究

本節では，ソフトウェアの開発工程を収集するシステム，及び Web ベースのプログラミング環境についての関連研究を述べる．

Hackeystat[3]は，emacs, Junit, Ant などのエディタや開発ツールに対して，センサを配置し，ソフトウェアの開発工程を測量するシステムである．しかし，導入には既にある開発環境にインストールする必要があり，この作業は初学者には敷居が高い．ASPEN は，Web ブラウザ上にプログラミング環境を構築しているため，特別なソフトウェアを導入することなく，開発工程の収集を行える．

一方，Web ブラウザでプログラミング環境として利用する研究も提案されている．Scheme 言語では Web ブラウザで実行するシステムである WeScheme[5]がある．主に代数学や幾何学，単純なモデリングを学ぶ学習者を対象とした，数学的な実験基盤として提供されている．CodeSchool[6] では，Web アプリケーションの開発に関するコーディング技術向上のための教育プログラムが組まれている．ASPEN は，現状は開発の観点での学習者のデータ収集を行うことを目的としているため，これらとは目的が異なる．

また，PSP 技法を教育に活用する試みは,既に多数の研究が提案されている [1][2][3]．しかし，これらが提示している手法や計測のメトリクスが，大学などのプログラミング初学者の現場で有効であるか，まだ十分な結論は得られていない．

5. むすびに

本論文では，プログラミング演習の自習の支援および，開発の実態を把握するための支援環境である ASPEN を提案した．本論文では ASPEN の仕組みや利用方法，また実際に導入の実践を行った結果,を示した．また，データ収集にあたっては，PSP の指標を導入し，個人データの解析を行い，正答率が高い生徒と低い生徒に違いがみられたことを報告した．

今後は，ASPEN の収集した学習履歴を利用した，理解の遅い学生の検知や,データを学習者にフィードバックし，その効果を確認するなどの支援を検討している．また， ASPEN の外部公開を推進して行き，多くの方々からのフィードバックを受けて改善して行きたい．

謝辞 ASPEN の教育実践にご協力いただいた早稲田大学，および横浜国立大学の学生に感謝いたします．

参考文献

[1] James B. Fenwick, Jr., Cindy Norris, Frank E. Barry, Josh Rountree, Cole J. Spicer, and Scott D. Cheek. Another look at the behaviors of novice programmers. *SIGCSE Bull.*, Vol. 41, No. 1, pp. 296–300, March 2009.

[2] Gregory Dyke. Which aspects of novice programmers' usage of an ide predict learning outcomes. SIGCSE '11, pp. 505–510, New York, NY, USA, 2011. ACM.

[3] Philip M. Johnson, Hongbing Kou, Joy Agustin, Christopher Chan, Carleton Moore, Jitender Miglani, Shenyan Zhen, and William E. J. Doane. Beyond the personal software process: metrics collection and analysis for the differently disciplined. ICSE '03, pp. 641–646, Washington, DC, USA, 2003. IEEE Computer Society.

[4] Watts Humphrey. *Introduction to the personal software process(sm).* Addison-Wesley Professional, first edition, 1996.

[5] Danny Yoo, Emmanuel Schanzer, Shriram Krishnamurthi, and Kathi Fisler. Wescheme: the browser is your programming environment. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, ITiCSE '11, pp. 163–167, New York, NY, USA, 2011. ACM.

[6] Envy Labs. Code school. http://www.codeschool.com/.

[7] Essi Lahtinen, Kirsti Ala-Mutka, Hannu-Matti Järvinen: A study of the difficulties of novice programmers. ITiCSE 2005: 14-18.