Vol.2017-CE-139 No.23
2017/3/12

Hanabi:プログラミング教育改善のための横断的フィルタ機能を有するダッシュボード

田中 良樹[1,a)]　松澤 芳昭[2,b)]　木谷 友哉[1,c)]　酒井 三四郎[3,d)]

概要：プログラミング教育現場で取得される学習記録分析指標 (コーディングメトリクス) の分析を行うことで，教育者が学習環境を改善することを目指した情報ダッシュボード提示システム「Hanabi」を開発した．Hanabi の特徴は，1) 学習環境の分析に有用なコーディングメトリクスを採用していること，2) データに横断的なフィルタをかけ，多様な分析対象を指定できること，の 2 点である．文科系の大学生を対象としたプログラミング入門授業における 4 年分のデータを利用して，Hanabi を用いた 5 年目の授業の改善を試みた．本研究では，Hanabi を用いて議論された 7 例について，それぞれ Hanabi の使われ方と導かれた議論の質的分析を行った．その結果，Hanabi の支援によって授業内容，教示方法，学習者の属性による傾向の違いなど，広範囲に渡る授業状況の理解と改善が促進されることが明らかになった．

キーワード：コーディングメトリクス，ビジュアルプログラミング言語，コンパイルエラー，プログラミング教育，Learning Analytics

Hanabi: An Information Dashboard for Programming Class with Cross Filtering for Teachers to Improvement their Classroom

Abstract: In this paper, we reported the development of "Hanabi" an information dashboard system which provides coding metrics to visualize the students' engagement of their assignments. The system designed for teachers to improve their classroom: to help continuous evidence-based action research in education. We designed the dashboard based on our previous research which demonstrated analyzing coding metrics, and for teachers to facilitate the analysis. The system was equipped with cross-filter functionality for exploring the metrics, hence teachers can easily conduct temporal analysis or across-year comparison. We examined the system for the improvement of the 5th year trial using the latest 4 years' data, on our non-CS introductory class in university. The qualitative analysis was examined using the discourse between teachers and teaching assistants with the dashboard. The result shows that the system succeeded to promote the discourse which includes the clear understanding of the class and its improvement, such as teaching method, assignments, or property of students.

Keywords: Coding Metrics, Visual Programming Language, Compile Error, Programming Education, Learning Analytics

1. はじめに

プログラミング教育では，プログラムの読み書きを習得するだけでなく，現実社会における問題について，形式的な解決方法をデザインする能力も求められている [1]．そのような能力を育むプログラミング教育の形成的評価と改善のために，Educational Data Mining や Learning Analytics と呼ばれるデータ解析技術が近年注目されている [2]．

プログラミング教育は学習者が実際にプログラムを記述

[1] 静岡大学 総合科学技術研究科 情報学専攻
Department of Informatics, Graduate School of Integrated Science and Technology, Shizuoka University
[2] 青山学院大学 社会情報学部
School of Social Informatics, Aoyama Gakuin University
[3] 静岡大学 情報学部
Faculty of Informatics, Shizuoka University
[a)] tanaka@sakailab.info
[b)] matsuzawa@si.aoyama.ac.jp
[c)] t-kitani@inf.shizuoka.ac.jp
[d)] sakai@inf.shizuoka.ac.jp

ⓒ 2017 Information Processing Society of Japan

して解答する演習形式で行われるのが一般的である．こうした演習内で躓いている学習者を検出する目的で学習者の学習過程データを収集し活用する研究は数多く存在する [3][4][5][6][7]．しかし，これらの学習者の学習過程データを，カリキュラムや指導法の改善に継続的に利用するという試みは存在しない．学習者に対するテストやアンケート結果と，これらのデータによる分析結果，および教師の経験から得られた知見を組み合わせることで，更なる授業改善が期待できる．

我々は，コーディングメトリクスを用いた学習者の学習過程データの分析 [8] を行なってきた．分析の結果，Java を用いてプログラミングを行なった学生は概算で 20%程度コンパイルエラー修正に費やしているとことや，コンパイルエラーで躓き，ブロック言語を利用して学習を継続した学生が 2%程度存在することなどが明らかになった．

そこで本研究では，教育現場における教師が利用することを想定した，学習過程の分析システム “Hanabi” を提案する．提案システムの特徴はコーディングメトリクスをダッシュボード形式で可視化すること，横断的なフィルタ機能を有することである．

2. 先行研究

プログラミング教育においても，Learning Analytics 分野の研究が推進されている．Toll ら [9] は，学習者における，コンパイル，テキストの編集，アクティブな使用，時刻，といったような粒度のログを収集し，分析を行なっている．同レベルの粒度のログによる研究として，Jadud の研究 [10] や，松澤ら [11] の研究のような，コンパイルエラーの分析や，ClockIt[12] や Retina[13] などの分析が存在する．ClockIt は学習者のために，設計されており，Retina は学習者と教師の両方の利用を想定して設計されている．しかし，これらの研究のような，学習者 1 人 1 人の詳細な学習ログを分析するというアプローチで，クラス全体の改善を行うのは限界がある．

Helimen[14] や PPV[15] はタイピングレベルの細かな粒度のログデータを収集し，プログラミング過程を再現するツールである．しかし，これらのツールは個々の生徒の詳細な分析を目的としているため，本研究の目的とは異なっている．

プログラミング演習中の学習者の進行状況をリアルタイムで把握するツールは数多く存在する．加藤ら [3] は，クラス全体における学習者の作業進度や発生させているエラーの種類をリアルタイムに教員に提示するシステムの提案を行っている．市村ら [5] は，つまづいている学生の早期発見と，学生が共通にかかえる問題を発見することを目的として，学習者のログを収集し，教員に提示するシステムを提案している．倉澤ら [16] はプログラムの動作過程をアニメーションによって表示する機能とコンパイルエラー発生状況の 2 点に着目し，学習者が理解困難な行やエラーの推定を推定し，教員に提示するシステムの提案を行っている．Alammary[6] や，c3pv[4] は，学習者の進行状況をシートマップを用いて可視化し，教員に提示するシステムを提案している．しかし，これらのシステムは，学習環境やカリキュラムの改善を目的に開発されておらず，学習環境の改善にどの程度寄与しているのかは不明である．

proGrep[17] は，学習者の困難解決の補助と，学習状況の把握と対応を目的として，学習者の学習履歴の収集，解析，活用を行うシステムである．収集した学習履歴を用いることで，多くの学習者に共通して見られた事例に対する補習や，練習問題の作成などの授業改善を行っている．しかし，学習者の直面した困難の解決を元に授業改善が行われているため，学習者全体の学習状況の分析や，授業改善による効果の分析は行われていない．

田口ら [18] はプログラミング演習における学習内容の改善のために，学習者の学習記録を収集，活用している．学習者ごとの各演習課題の達成度を推測し，豊富に用意された演習課題の中から最適な演習課題を選出する．しかし，提案システムは学習者にとって最も最適な課題を選択する手法の提案であり，学習環境の改善が目的ではない．

藤原ら [7] は，理解に行き詰まっている学習者を発見することを目的として，取組状況を示すデータ項目と呼ばれ 5 つの項目の時系列データの分析を行う手法の提案を行なっている．折れ線グラフを用いて，データの分析を行い，理解に行き詰まっている学習者の検出を行っている．定量的なデータをグラフを用いて可視化し，分析を行うという方向性は，本研究と同様である．

Johnson ら [19] は，医療現場における ICU(Intensive Care Unit) に例えて，ソフトウェアをソフトウェアエンジニアリングメトリクスを元に視覚化するシステムを提案している．提案システムはダッシュボード形式を採用しており，本研究の手法と類似している．

Heig ら [20] は，学習管理システムの利用法を示す視覚化ツールを提案し，学習システムへのアクセスを元に，学生の行動パターンを検出しようと試みてる．この研究で用いられるヒートマップによる視覚化は，本研究の手法と類似している．

ペーパーテストを用いて大学生向けのプログラミング入門教育における，学習者のプログラミング能力を測定するという試みも存在する．Lister ら [21] は，いくつかの地域の大学における学生のプログラミングを読む能力と，トレースする能力を報告している．Ford[22] は，認知科学的なアプローチから，学生のプログラミング学習における達成度を評価している．その結果，学生の 50%程度しか学習内容を理解できていないことが明らかになり，改善を試みている．

ⓒ 2017 Information Processing Society of Japan

図 1　Hanabi の外観

3. Hanabi の提案

本研究の目的は，現場の教師が学習者の学習記録を分析し，学習環境の改善を試みるシステムを提案することである．

3.1 Hanabi の目的

Hanabi は，実際の教育現場における教師が学習環境を分析することを支援するシステムである．Hanabi は，リアルタイムにプログラミング演習中における学習者の態を把握し改善を行うといったような「短期的」な改善サイクルではなく，前年度までの学習者の傾向から実態を分析し，改善案を議論し，改善を実施するといったような「長期的」な改善サイクルの支援を目的としている．

Hanabi は学習環境における実態の把握を容易にするために，データをグラフ形式にし，一覧表示するダッシュボード形式を採用する．分析対象を柔軟に指定できるようにすることで，学習者の抱える問題や授業内のより詳細な実態の把握が可能になるように，データに横断的なフィルタをかける機能を採用する．

授業改善のための授業分析に用いる定量的な指標として，コーディングメトリクス [8] を利用する．コーディングメトリクスは以下に示した 7 つである．

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

3.2 実装

Hanabi は Hanabi は JavaScript を用い，Web クライアントアプリケーションとして実装した．インストール不要であるため，即座に教育現場で分析可能である．

使用ライブラリは，D3.js，crossfilter.js，dc.js である．D3.js はグラフの描画を行う．crossfilter.js はデータにフィルタをかける．dc.js はこれらのライブラリの連携を担う．

3.3 設計

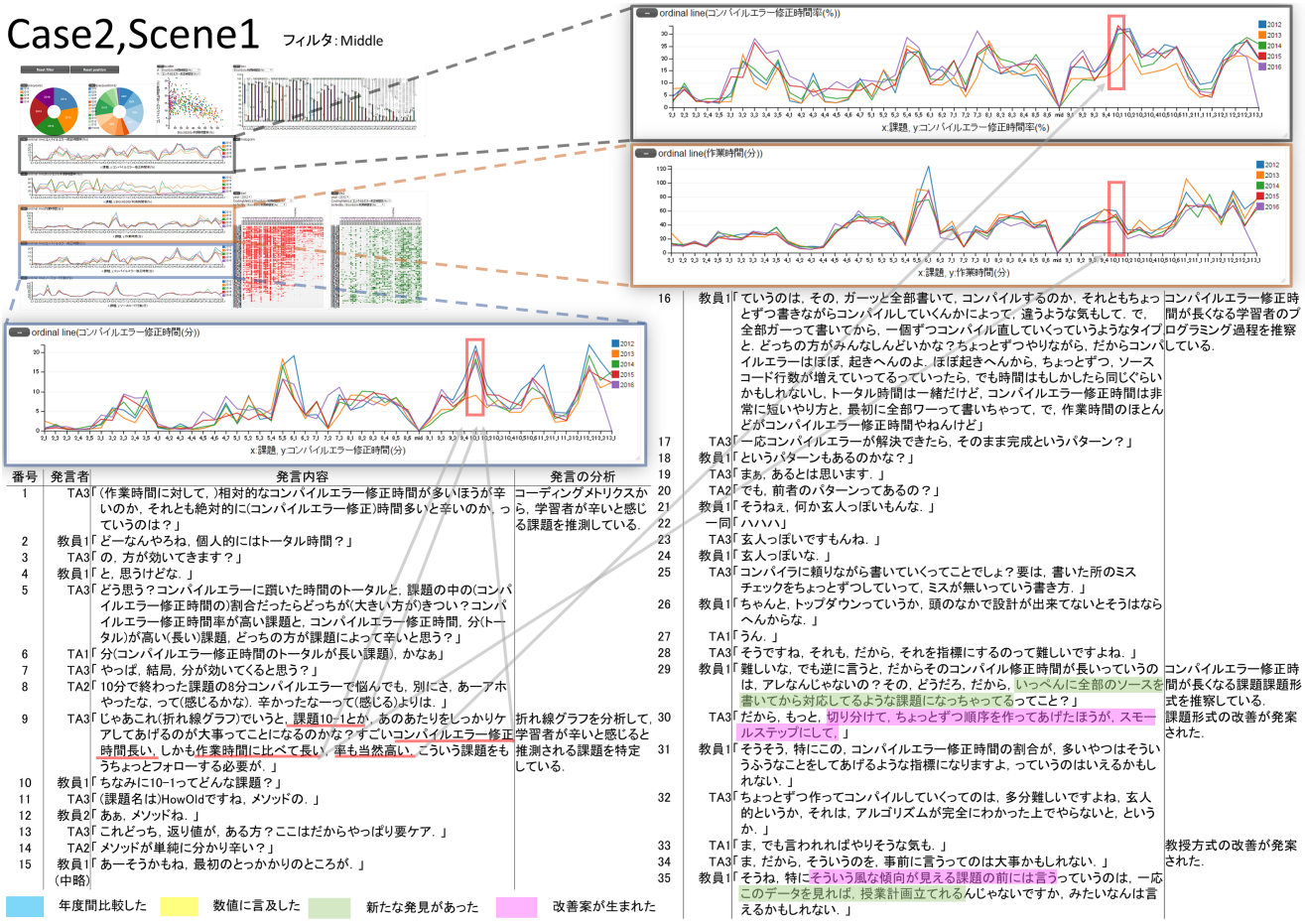
Hanabi の外観を図 1 に示す．HD(1920 × 1080 ピクセル) ディスプレイで全てのグラフが鳥瞰可能な設計としている．Hanabi に採用したグラフは以下の 6 つである．

① 円グラフ
　各年度における学習者の割合と課題の割合をそれぞれ示す．

② 折れ線グラフ
　各年度における課題ごとの 5 つのコーディングメトリクスの平均の推移を示す．

ⓒ 2017 Information Processing Society of Japan

Vol.2017-CE-139 No.23
2017/3/12



番号	発言者	発言内容	発言の分析
1	TA3	「(作業時間に対して,)相対的なコンパイルエラー修正時間が多いほうが辛いのか, それとも絶対的に(コンパイルエラー修正)時間多いと辛いのか, っていうのは？」	コーディングメトリクスから, 学習者が辛いと感じる課題を推測している.
2	教員1	「どーなんやろね, 個人的にはトータル時間？」	
3	TA3	「の, 方が効いてきます？」	
4	教員1	「と, 思うけどな. 」	
5	TA3	「どう思う？コンパイルエラーに躓いた時間のトータルと, 課題の中の(コンパイルエラー修正時間の)割合だったらどっちが(大きい方が)きつい？コンパイルエラー修正時間率が高い課題と, コンパイルエラー修正時間, 分(トータル)が高い(長い)課題, どっちの方が課題によって辛いと思う？」	
6	TA1	「分(コンパイルエラー修正時間のトータルが長い課題), かなぁ」	
7	TA3	「やっぱ, 結局, 分が効いてくると思う？」	
8	TA2	「10分で終わった課題の8分コンパイルエラーで悩んでも, 別にさ, あーアホやったな, って(感じるかな). 辛かったなーって(感じる)よりは. 」	
9	TA3	「じゃあこれ(折れ線グラフ)でいうと, 課題10-1とか, あのあたりをしっかりケアしてあげるのが大事ってことになるのかな？すごいコンパイルエラー修正時間長い, しかも作業時間に比べて長い, 率も当然高い, こういう課題をもうちょっとフォローする必要が. 」	折れ線グラフを分析して学習者が辛いと感じると推測される課題を特定している.
10	教員1	「ちなみに10-1ってどんな課題？」	
11	TA3	「(課題名は)HowOldですね, メソッドの. 」	
12	教員2	「あぁ, メソッドね. 」	
13	TA3	「これどっち, 返り値が, ある方？ここはだからやっぱり要ケア. 」	
14	TA2	「メソッドが単純に分かり辛い？」	
15	教員1	「あーそうかもね, 最初のとっかかりのところが. 」	
		(中略)	
16	教員1	「というのは, その, ガーッと全部書いて, コンパイルするのか, それともちょっとずつ書きながらコンパイルしていくんかによって, 違うような気もして. で, 全部ガーって書いてから, 一個ずつコンパイル直していくっていうようなタイプと. どっちの方がみんなしんどいかな？ちょっとずつやりながら, だからコンパイルエラーはほぼ, 起きへんのよ. ほぼ起きへんから, ちょっとずつ, ソースコード行数が増えていってるっていったら, でも時間はもしかしたら同じぐらいかもしれないし, トータル時間は一緒だけど, コンパイルエラー修正時間は非常に短いやり方と, 最初に全部ワーって書いちゃって, で, 作業時間のほとんどがコンパイルエラー修正時間やねんけど」	コンパイルエラー修正時間が長くなる学習者のプログラミング過程を推察している.
17	TA3	「一応コンパイルエラーが解決できたら, そのまま完成というパターン？」	
18	教員1	「というパターンもあるのかな？」	
19	TA3	「まぁ, あるとは思います. 」	
20	TA2	「でも, 前者のパターンってあるの？」	
21	教員1	「そうねぇ, 何か玄人っぽいもんな. 」	
22		一同「ハハハ」	
23	TA3	「玄人っぽいですもんね. 」	
24	教員1	「玄人っぽいな. 」	
25	TA3	「コンパイラに頼りながら書いていくってことでしょ？要は, 書いた所のミスチェックをちょっとずつしていって, ミスが無いっていう書き方. 」	
26	教員1	「ちゃんと, トップダウンっていうか, 頭のなかで設計が出来てないとそうはならへんからな. 」	
27	TA1	「うん. 」	
28	TA3	「そうですね, それも, だから, それを指標にするのって難しいですよね. 」	
29	教員1	「難しいな, でも逆に言うと, だからそのコンパイル修正時間が長いっていうのは, アレなんじゃないの？その, どうだろ, だから, いっぺんに全部のソースを書いてから対応してるような課題になっちゃってるってこと？」	コンパイルエラー修正時間が長くなる課題課題形式を推察している.
30	TA3	「だから, もっと, 切り分けて, ちょっとずつ順序を作ってあげたほうが, スモールステップにして. 」	課題形式の改善が発案された.
31	教員1	「そうそう, 特にこの, コンパイルエラー修正時間の割合が, 多いやつはそういうふうなことをしてあげるような指標になりますよ, っていうのはいえるかもしれない. 」	
32	TA3	「ちょっとずつ作ってコンパイルしていくってのは, 多分難しいですよね, 玄人的というか, それは, アルゴリズムが完全にわかった上でやらないと, というか. 」	
33	TA1	「ま, でも言われればやりそうな気も. 」	教授方式の改善が発案された.
34	TA3	「ま, だから, そういうのを, 事前に言うってのは大事かもしれない. 」	
35	教員1	「そうね, 特にそういう風な傾向が見える課題の前には言うっていうのは, 一応このデータを見れば, 授業計画立てれるんじゃないですか, みたいなんは言えるかもしれない. 」	

年度間比較した　数値に言及した　新たな発見があった　改善案が生まれた

図 2　談話の質的分析の 1 例 (Case2)

③ 散布図

選択した 2 つのコーディングメトリクスにおける学習者の分布を示す.

④ 箱ひげ図

選択したコーディングメトリクスにおける，各課題ごとのコーディングメトリクスにおける学習者の分布を示す.

⑤ ヒストグラム

1 つの課題における学習者の分布を示す.

⑥ タイル図

選択したコーディングメトリクスを，学習者 1 人の課題 1 つごとに色の濃淡で示す.

グラフはそれぞれ任意の位置に移動したり，拡大縮小したり，非表示にしたりすることが可能である．そのため，着目したいグラフのみを抽出したり，並べたりして分析を行うことも可能である.

4. 実験方法

4.1 目的

実験の目的は，以下の仮説を検証することである.

仮説： 教師による Hanabi を用いた分析は，学習環境の改善に貢献する.

4.2 対象

対象とする学習環境は，文科系学部 1 年生を対象としたプログラミング入門科目である．この学習環境では，ブロック言語と Java の併用環境 [23] が利用されている．当該授業では，授業終了後に授業実施者 (教員と Teaching Assistant) によって，今回の授業の反省と，次回授業の注意点を議論するミーティングが行われる.

2016 年度は，同様の学習環境における，5 年目の授業であった．そのため，Hanabi には予め，2012 2015 年度の同授業で収集したデータと，2016 年度のミーティングが行われる前週までの，課題で収集したデータを入力した．そのため，ミーティングを行う際には，Hanabi を用いたこれらのデータの分析が可能な状態とした.

第 6〜10 回，第 12 回，第 14 回の，計 6 回の授業後のミーティングにおける授業改善の議論の場面で，Hanabi による分析も行なった．計 6 回のミーティングから，Hanabi を用いて議論を行っている様子を 7 例，確認した．7 例を Case1〜7 と名付け，分析対象とした.

ⓒ 2017 Information Processing Society of Japan

表 1　コーディング基準と発言の例

コード	説明	発言の例
年度間比較した	グラフの重なりを確認したり，データにフィルタをかけることで，数値の年度間の比較を行っている．	(グラフの形が) 今年は逆ですね/(中間課題のソースコード行数の)2013 年までの最高が 3000(行) ちょっとぐらい
数値に言及した	グラフの分析から，時間，行数，人数，割合などの数値に言及している．	20 %だから，4 人に 1 人か 5 人に 1 人ぐらいは，(BlockEditor を) まぁまぁ使ってた/Middle(中間課題) の外れ値がさ，12000 行超えてたりなんかする
新たな発見があった	新たな学習者の実態が明らかになっている．前年度までの授業内の観察による知見がデータで裏付けられている．	好きな図形描くために頑張り倒して 2000 行書いた/(作業時間とコンパイルエラー修正時間には，) 一応正の相関はね，そらそうやろね
改善案が生まれた	学習者の実態から，課題の形式や指導方法などの改善案が生まれている．	(課題を) 切り分けて，ちょっとずつ順序を作ってあげたほうが，スモールステップにして/どうやってインストラクションしたら良いんだ？デバッグモードでやったほうがいいのか？

4.3　ミーティングにおける議論の分析

議論の内容を記録しておき，議論の内容について質的な分析を行なった．分析の例を図 2 に示す．発言内容の項目は，議論における発言者の発言を示している．発言の分析の項目は，分析者が発言の内容をどのように解釈したのかを示している．発言内容における赤い下線は，Hanabi を見て発言したと考えられる部分に引かれており，赤い下線から伸びている灰色の矢印は，その発言をした時点で，注目されていたと考えられるグラフの箇所を指し示している．

議論の中で，「年度間比較をした」，「数値に言及した」，「新たな発見があった」，「改善案が生まれた」と解釈できる発言をコーディングした．コーディングの基準と，発言の例を表 1 に示す．コーディングを行った箇所は，それぞれ，色分けをして網掛けを行なった．

4.4　ヒートマップの作成

ミーティングにおける，グラフ利用の網羅率を視覚化するため，ヒートマップを作成した．

1 つの Case について，議論の最中に着目するグラフが変わった時点を境界として，いくつかの Scene に分割した．1 つの Scene ごとに，グラフに言及しながら発言されたと考えられる箇所 (図 2 における矢印) を，「利用された」と定義した．Scene ごとに利用された回数を加算し，網羅率を測定した．

5.　結果

5.1　ミーティングにおける議論の分析結果

Case1〜7 における議論の概要を表 2 に示す．Case1 では，実際に課題の順序が変更され，変更による効果を定量的に分析する様子が観察された．Case2 や Case5 では，定量的な分析を元に，課題の特性を議論し，改善案が導かれた．Case3 や Case5 では，新たに，学習者の実態が明らかになり，それによって議論が発展した様子が確認された．Case6 では，コーディングメトリクス間の相関が分析され，授業実施者の経験との相違があることが明らかになった．Case7 では，課題の形式による影響が定量的なデータを元に議論された．

各 Case ごとのコーディング結果を表 3 に示す．結果から，新たな発見は全ての Case で確認され，合計 19 件であった．数値に言及した発言も 21 件見られ，定量的な分析を元に議論が行われていると判断できる．年度間比較は 7 件確認され，改善案も 3 件確認された．

5.2　ヒートマップ

作成されたヒートマップを図 3 に示す．ヒートマップから，7 つの Case において，6 種類，全てのグラフが利用されていることが分かった．7 つの Case のうち，6 つの Case で円グラフを利用して，フィルタをかけて分析が行われていた．最も多く利用されたのは，作業時間の折れ線グラフで，最も利用されなかったのは，ソースコード行数の折れ線グラフだった．

6.　考察

6.1　Hanabi のデザインの評価

Case1〜7 の分析において，全てのグラフが利用され，全ての Case で新たな発見があった．これらの結果から，Hanabi で採用した 6 種類のグラフは全て分析に有用に作用したと考えられる．

議論の中には，折れ線グラフに着目し議論が行われ，更に詳細な分析を行うために，タイル図や箱ひげ図を利用するという様子が観察された．複数のグラフを比較して分析が行わえる様子も観察された．こうした結果から，複数のグラフが表示されているダッシュボード形式は有用であったと考えられる．

7 つの Case のうち，6 つの Case データにフィルタをかけて分析が行われていることが確認された．この結果から，フィルタ機能は分析に必要であると考えられる．特に課題ごとや年度ごとの分析に利用されていたころから，フィル

ⓒ 2017 Information Processing Society of Japan

表 2 各 Case ごとの議論の概要

	議論の概要
Case1	Hanabi を用いて，昨年度までの学習者の傾向を分析したところ，課題 7-1 は作業時間が長くなっていた．課題の解答に必要な知識は，既習範囲の知識と，新たに学習する知識に分類可能だと考えられるが，学習者がどちらに躓いているのかは不明であるという議論が起こった．そこで，本年度は，新たに学習する内容である，「丸め誤差」についての知識を必要としない課題 7-2 を先に解答させるという変更を行なった．変更の結果を分析したところ，「丸め誤差」について悩んでいる時間は 10 分程度だと推測された．これらの課題では，コンパイルエラー修正時間率が大きい学習者が存在することや，本年度の BlockEditor 利用時間率が前年度までと比べて小さいことなども確認された．
Case2	授業実施者の経験から，学習者が苦痛に感じる課題は，コンパイルエラーの修正にかかる時間が長い課題だと推測された．Hanabi を用いて，コンパイルエラー修正時間が長い課題を確認した．コンパイルエラーの修正にかかる時間が長くなるのは，一度にまとめてコンパイルをしていることが原因だと考えた．そのため，こうした課題に対しては，課題をスモールステップに分割することや，こまめにコンパイルするように指導する，といった改善が発案された．
Case3	中間課題は自由に GUI 作品を作成するという解答形式であるが，平均の作業時間は 300 分程度であった．分布を確認したところ，最大値が 1600 分を超えており，ソースコード行数の最大値は 12000 行を超えていることが明らかになった．これは，授業実施者にとって，想定以上であった．中間課題はメソッドを学習する前の課題であるため，プログラムの大半はコピー&ペーストであるか，もしくは外部ライブラリを利用して作成されたのではないかと推測された．
Case4	再帰メソッドについて学習する課題は，コンパイルエラー修正時間は短いが，作業時間が長くなるという傾向が見られた．こうした課題はデバッガを利用してインストラクションを行なった方が良いという改善案が生まれた．
Case5	1 課題あたりの作業時間は 40 分程度であることから，学習者が授業外にどの程度プログラミングを行っているのかを推測した．その結果，平均で 1 時間程度であることが推測され，これが適当であるかどうかの議論が起こった．
Case6	コーディングメトリクス間の相関についての分析が行われた．作業時間とソースコード行数に相関がないというのは，予想外の結果であり，原因が考察された．考察の結果，解答形式が決まっている課題の場合，課題の最終的なソースコード行数は，およそ決まった形になることが原因だと推測された．
Case7	BlockEditor を利用して解答するように指定されている問題 (BLOCK 問題) の数が少ない年度は，BlockEditor の利用率が課題の序盤から下落しており，多い年度は，課題の中盤まで 4 割程度，終盤でも 3 割程度，利用されていることが確認されている．BlockEditor を利用する動機については，コンパイルエラーを回避する目的だと考察された．BlockEditor を利用している学習者は継続して利用し続ける傾向を確認し，こうした学習者の作業時間とソースコード行数は他の学習者と大差がないことも確認している．

表 3 各 Case におけるコーディング結果

	年度間比較した	数値に言及した	新たな発見があった	改善案が生まれた
Case1:課題の順序変更の効果についての議論	3	2	3	0
Case2:コンパイルエラー修正時間に関する議論と改善案	0	0	2	2
Case3:中間課題における取組状況の議論	3	13	4	0
Case4:課題の特性とインストラクションの改善案	0	0	2	1
Case5:授業外におけるプログラミング時間の推測と議論	0	2	1	0
Case6:コーディングメトリクス間の相関の議論	0	0	2	0
Case7:BlockEditor の利用に対する議論	1	4	5	0
計	7	21	19	3

タ機能によって，詳細な分析が可能になっていると考えられる．

6.2 Hanabi を用いた分析による効果

試用実験における議論の分析の結果から，新たな発見は全ての Cese で確認され， 7 つの Case において，19 件であった．この結果から，Hanabi は学習環境における実態を明らかにすることに貢献していると考えられる．「新たな発見があった」というコードは，「新たな学習者の実態が明らかになっている」ことに加え，「全年度までの授業内の観察による知見が数値で裏付けられている」と，定義した．授業内の観察による知見は，ミーティング内で議論されない限りは，暗黙知として各授業実施者内に蓄積される．しかし，Hanabi を用いて分析し，これらの暗黙知がデータで裏付けられるかを議論することで，各授業実施者内の暗黙知が学習者の実態と整合しているかが確認され，データを元に形式知に変換され，授業実施者内で共有されるという利点がある．特に暗黙知がデータによって裏付けられなかった場合は，授業実施者は誤った指導をしている可能性がある．Hanabi を用いることで，こうした授業実施者の学習者に対する認識の改善が期待できる．

数値に言及した発言は 21 件確認された．これは，定量的な分析が行われていることの根拠になると考えられる．特に，議論の中では，具体的な数値ではなく，グラフの外

ⓒ 2017 Information Processing Society of Japan

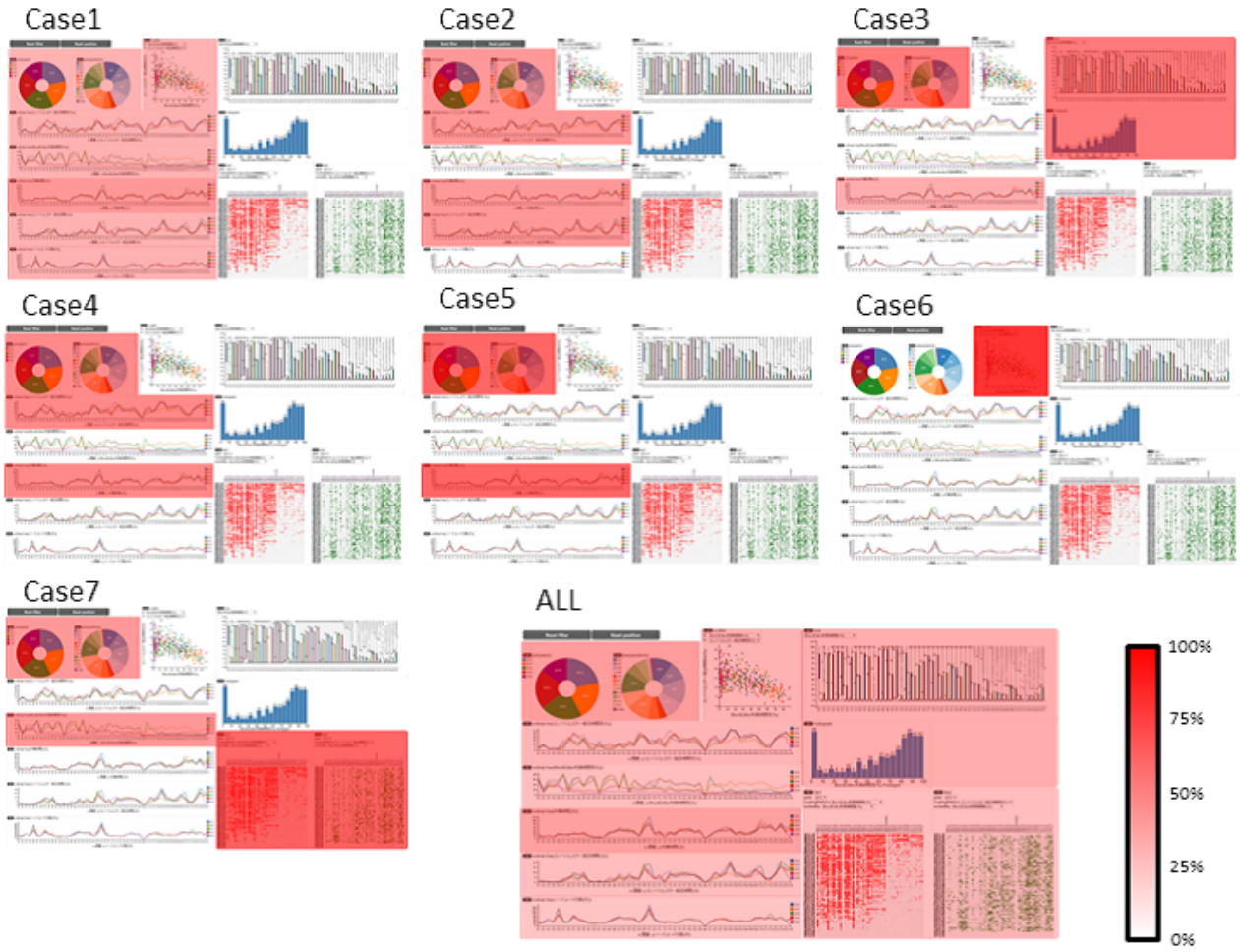


図 3 ヒートマップ

形に着目して分析する様子も観察されており，こうした分析も含めると，定量的な分析は，更に多く行われていると考えられる．学習者の実態を把握する手段として，学習者に対するアンケートや聞き取り調査や，授業実施者による授業内の観察が考えられる．しかし，学習者は客観的に自分の学習状況を判断することが難しく，個人の能力によって，授業に対する感想も違ってくると考えられ，授業改善に役立てるのは適当ではないと考えられる．授業実施者による授業内の観察は，数人の教員や TA で網羅するのには限界があることや，観察対象となる学習者の実力の偏りによって，授業実施者間で，現状に対する認識に差が出てしまうことが問題だと考えられる．ツールを用いて，授業内の学習者の進捗状況を確認するだけでは，課題ごとの学習状況の比較や，学習が進むにつれて変化する学習者の実態の把握が難しいという問題がある．Hanabi を用いることで，こうした問題へ対処できていると考えられる．

新たな発見は 19 件であったのに対し，改善案は 3 例であった．今回の実験では，15 回のうち，6 回のミーティングでの利用に留まっていたため，全てのミーティングで利用することによって，更に多くの発見や改善案が期待できると考えられる．しかしながら，Hanabi のデザインは新たな発見を支援する機能に留まっており，改善案を導く機能が不足しているという可能性もある．改善案を導くためには，新たな発見から，問題点を分析する必要があると考えられ，こうした問題点を更に抽出するようなデザインが必要であると考えられる．

今回の実験では，授業実施者の 1 人として，Hanabi の開発者がミーティングに参加した．Hanabi の開発者は，過去 4 年分の当該授業におけるデータを分析する研究に関わっていたため，他の授業実施者と比べて知識が豊富であると考えられる．こうした開発者がミーティングに参加することで，より詳細な分析が行われ，その結果多くの発見や改善案が示された可能性がある．しかし，新たな発見や改善案は，開発者からのみではなく，他の授業実施者からも多く生まれており，開発者が議論に参加していなくても，Hanabi による分析は新たな発見に貢献すると考えられる．

謝辞 本研究は JSPS 科研費 25730203，26280129 の助成を受けたものです．

ⓒ 2017 Information Processing Society of Japan

参考文献

[1] Wing, J.: Computational Thinking, *Communications of the ACM*, Vol. 49, No. 3, pp. 33–35 (2006).
[2] Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Borstler, J., Edwards, S., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M., Sheard, J., Skuupas, B., Spacco, J., Szabo, C. and Toll, D.: Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies, pp. 41–63 (2015).
[3] 加藤利康，石川　孝：プログラミング演習のための授業支援システムにおける学習状況把握機能の実現，情報処理学会論文誌，Vol. 55, No. 8, pp. 1918–1930 (2014).
[4] 井垣　宏，齋藤　俊，井上亮文，中村亮太，楠本真二：プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案，情報処理学会論文誌， Vol. 54, No. 1, pp. 330–339 (2013).
[5] 市村　哲，梶並知記，平野洋行：プログラミング演習授業における学習状況把握支援の試み，情報処理学会論文誌，Vol. 54, No. 12, pp. 2518–2527 (2013).
[6] Alammary, A., Carbone, A. and Sheard, J.: Implementation of a smart lab for teachers of novice programmers, *Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123*, Australian Computer Society, Inc., pp. 121–130 (2012).
[7] 藤原理也，田口　浩，島田幸廣，高田秀志，島川博光：ストリームデータによる学習者のプログラミング状況把握，電子情報通信学会第 18 回データ工学ワークショップ，D9-5 (2007).
[8] 田中良樹，松澤芳昭，酒井三四郎ほか：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，情報教育シンポジウム 2016 論文集， Vol. 2016, pp. 59–66 (2016).
[9] Toll, D., Olsson, T., Ericsson, M. and Wingkvist, A.: Fine-grained recording of student programming sessions to improve teaching and time estimations, *International Journal of Engineering, Science and Innovative Technology*, Vol. 32, No. 3, pp. 1069–1077 (2016).
[10] Jadud, M. C.: Methods and tools for exploring novice compilation behaviour, *Proceedings of the second international workshop on Computing education research*, ACM, pp. 73–84 (2006).
[11] Matsuzawa, Y., Hirao, M. and Sakai, S.: Compile Error Collection Viewer: Visualization of Compile Error Correction History for Self-assessment in Programming Education, *INTERNATIONAL JOURNAL OF ENGINEERING EDUCATION*, Vol. 32, No. 3, pp. 1117–1127 (2016).
[12] Norris, C., Barry, F., Fenwick Jr, J. B., Reid, K. and Rountree, J.: ClockIt: collecting quantitative data on how beginning software developers really work, *ACM SIGCSE Bulletin*, Vol. 40, No. 3, pp. 37–41 (2008).
[13] Murphy, C., Kaiser, G., Loveland, K. and Hasan, S.: Retina: helping students and instructors based on observed programming activities, *ACM SIGCSE Bulletin*, Vol. 41, No. 1, pp. 178–182 (2009).
[14] Helminen, J., Ihantola, P. and Karavirta, V.: Recording and analyzing in-browser programming sessions, *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, ACM, pp. 13–22 (2013).
[15] Matsuzawa, Y., Okada, K. and Sakai, S.: Programming process visualizer: a proposal of the tool for students to observe their programming process, *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, ACM, pp. 46–51 (2013).
[16] 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 (collaboration と agent 技術/一般)，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).
[17] 長　慎也，筧　捷彦：proGrep-プログラミング学習履歴検索システム，情報処理学会研究報告コンピュータと教育 (CE)， Vol. 2005, No. 15 (2004-CE-078), pp. 29–36 (2005).
[18] 田口　浩，糸賀裕弥，毛利公一，山本哲男，島川博光：個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援，情報処理学会論文誌，Vol. 48, No. 2, pp. 958–968 (2007).
[19] Johnson, P. and Zhang, S.: We need more coverage, stat! Classroom experience with the Software ICU, *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, IEEE, pp. 168–178 (2009).
[20] Haig, T., Falkner, K. and Falkner, N.: Visualisation of learning management system usage for detecting student behaviour patterns, *Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136*, Australian Computer Society, Inc., pp. 107–115 (2013).
[21] Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O. et al.: A multi-national study of reading and tracing skills in novice programmers, *ACM SIGCSE Bulletin*, Vol. 36, No. 4, ACM, pp. 119–150 (2004).
[22] Ford, M. and Venema, S.: Assessing the success of an introductory programming course, *Journal of Information Technology Education*, Vol. 9, No. 1, pp. 133–145 (2010).
[23] 松澤芳昭，保井　元，杉浦　学，酒井三四郎：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).

ⓒ 2017 Information Processing Society of Japan