HAL open science

A Demonstration of Evidence-Based Action Research Using Information Dashboard in Introductory Programming Education

Yoshiaki Matsuzawa, Yoshiki Tanaka, Tomoya Kitani, Sanshiro Sakai

To cite this version:

Yoshiaki Matsuzawa, Yoshiki Tanaka, Tomoya Kitani, Sanshiro Sakai. A Demonstration of Evidence-Based Action Research Using Information Dashboard in Introductory Programming Education. 11th IFIP World Conference on Computers in Education (WCCE), Jul 2017, Dublin, Ireland. pp.619-629, 10.1007/978-3-319-74310-3_62 . hal-01762842

HAL Id: hal-01762842

https://inria.hal.science/hal-01762842v1

Submitted on 10 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A Demonstration of Evidence-Based Action Research using Information Dashboard in Introductory Programming Education

Yoshiaki Matsuzawa, Yoshiki Tanaka, Tomoya Kitani, Sanshiro Sakai

Aoyama Gakuin University, Japan,
matsuzawa@si.aoyama.ac.jp
Shizuoka University, Japan

Abstract. In this paper, we demonstrated an evidence-based action research in an introductory programming class with the use of an information dashboard which provides coding metrics to visualize students' engagement of their assignments. The information dashboard was designed for teachers to improve their classroom teaching using the same coding metrics which was verified in our previous research [9]. The system was equipped with a cross-filter functionality for exploring the entire classroom metrics. Accordingly, teachers can easily conduct a temporal analysis, an across-year comparison, and a cross metrics analysis. We examined the system for the improvement of the 5th year course using a dataset from the past four years from a non-CS introductory programming course at a university. Qualitative analysis was conducted using the discourse between teachers and teaching assistants with the proposed dashboard. The results showed that the system succeeded in promoting discourse, which included a clearer understanding of the class and its improvement, such as teaching method, assignments, or of students' behavior.

Keywords. Programming education, information dashboard, learning analytics, action research.

1. Introduction

Twenty-first century knowledge societies and the commonality of information and communication technologies in societies have been pushing for a fundamental reform in education [1]. This request will greatly affect learning management in the informatics field, with a shift from teaching how to use applications to "computing" - creating and designing a new problem solution with computational thinking [2]. This new assessment method for the programming education is significant for research in this field [3, 4].

The issue of definition and its assessment has been a long quest [5]. Recent technologies are enabling us to collect fine-grained massive logs in educational situations by an automated way. Using the terms of educational data mining or learning analytics, a new assessment method using the collected data has been anticipated even in programming education [6]. The first generation of research in the 2000s was limited to simple analysis such as compilation error occurrences profiles analysis (e.g. [7]), although recently fine-grained log analysis has increasingly

revealed the actual efforts for student assignments [8], or measuring the impact of block-based language [9].

Prior studies have highlighted the difficulty of reproducing/replicating research results in this field [6]. There is general difficulty in human subject studies, as controversial discussions are still ongoing in programming education research [10, 11]. We should continue the pursuit of research toward the goal of the elucidating the nature of programming education. Simultaneously, the environment for analysis using the collected dataset should be developed in order to improve actual classroom learning.

Towards this end, we proposed an information dashboard for teachers to improve their introductory programming classroom. The tool was designed to help teachers' action research to improve the teaching/learning environment, especially in the case of classes that are repeatedly conducted over a relatively long period of time (imagine the 2nd year course of the subject will be improved by the data of the 1st year course).

The academic contributions of this paper contain three aspects. First is the design of the dashboard, which is based on our previous research, and enable teachers to effectively ascertain the unique learning issues in their respective classrooms. In addition, as we use the latest web technology, teachers can easily access the dashboard and explore it using a filtering functionality. The second contribution is the application of the dashboard to actual classroom environments. Teachers tried to make improvements for 5th year course using the dashboard which shows the former4 years' data-set. The third contribution is our study methodology. All of the sessions containing the detailed discussions of the teachers and assistants with the dashboard were recorded and in-depth qualitative analyses for each session were carried out.

2. Related Work

As discussed in the previous section, the recorded data in the actual programming session and proposed method of analyzing the recorded data are a common research approach within this field. For example, Toll et al., 2016 [8] proposed four categories to be classified by the granularity of the recorded logs. The categories include Compilations, Text change, Active Use, and Time in Tool. Jadud's compilation analysis [7] is categorized as the Compilation level, and Matsuzawa's compile error visualization [12] can be categorized at the level as well.

ClockIt [13] and Retina [14] are categorized as the Text change level. Comparing this approach through focusing on teacher usage of the visualizations, ClockIt is designed for a single student. Retina is designed for both students and teachers; however, the teacher's view shows the detailed errors for each student. There are limitations in the ability for conducting action research to improve an entire class.

Helminen et al, 2013 [15] and PPV [16] are tools to display the replay of programming sessions using fine-grained typing logs. The purpose of these tools is to perform an in-depth analysis of each individual student. Alammary et al., 2012 [17] advocated a "Smart Lab". A dashboard is a system for teachers to visualize the students "seats map" and shows the status of each student. This kind of tool is considered useful for ascertaining the progress of student assignments in real-time. But its use is questionable toward the improvement of an entire class or curriculum.

In the field of software engineering, Johnson et al, 2012 [18] proposed a dashboard system. They used the metaphor of a medical ICU (Intensive Care Unit). The proposed system shows a commit (code) telemetry which visualizes the current status using the metrics used in the software engineering field.

Heig et al, 2013 [19] advocated a suite of visualization tools to show the usage of the learning management system and attempted to detect the student behavior patterns. The data they used were the access logs of a learning management system. Although differing from our research, the visualization of a heat map for each student has some resemblance to our tool.

Student performance in undergraduate programming classes has been studied using paper exams. Lister et al, 2004 [20] reported on reading and tracing skills for multi-regional university students. Ford, 2010 [21] tried to assess the achievement of their classes incorporating tests used in cognitive studies in programming [10]. Results showed that only 50% of students understood class assignments, so they attempted to improve this aspect of their classroom. This is quite a remarkable result as a form of evidence-based action research within programming education.

3. Information Dashboard

This section will provide a brief description of the information dashboard which is designed for teachers to improve an introductory programming class, by providing coding metrics [9] to show the engagement of students in their assignments. The whole view of the dashboard is shown below in Fig 1. According to Few [22], an information dashboard can be defined as follows:

> *A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance.*

The main objective of this study is to improve an introductory-level programming classroom. All the visualization graphs are laid out in an HD (1920 x 1080 pixels) screen, which provides teachers with a glance view of all the coding metrics. The purpose of the dashboard is not for real-time “monitoring”, but to promote teacher “exploration” of the data over a relatively long period of time (weekly, monthly, or yearly). Hence, the dashboard design resembles an “accumulated” view rather than a general telemetry system, which promotes understanding of the summary in a short time.

The logs of student computer operations were collected by the previously proposed framework [16], and the system computes the coding metrics: working time, LOC (lines of code), compile error correction time, and block editor usage ratio. The metrics are computed for each unit of assignment x student. Brief explanations of each graph in Fig 1 are described as follows:

1. Pie graphs show the years and weeks. A user can filter by clicking this part.
2. A scatter plot shows the relationship between two selected coding metrics.
3. Five line charts are shown for each of the 5 coding metrics, which are arranged with the assignments ordered chronologically. Each colored line shows a different year.
4. A box plot shows the distribution of students for a selected coding metric.

5. A histogram shows the distribution of a particular selected assignment.
6. Tiled representations provide a graduation analysis environment as used in our previous research [9].

Fig. 1. Proposed information dashboard

The system was implemented on the web with Javascript, with the user being able to use a browser to see the dashboard. The libraries of d3.js, dc.js, and crossfilter.js were used in the implementation; accordingly, the system is equipped with cross-filter functionality for exploring the entire classroom metrics. Teachers can easily conduct a temporal analysis or cross-year comparison.

4. Research Method

4.1 Research questions

We conducted an evaluation of the dashboard in an actual class setting. The research questions of the evaluation generally focused on whether or not we would be able to achieve the objectives of the dashboard design:

RQ1: Can the dashboard facilitate among teachers productive discussions for the goal of improving the quality of introductory programming classes? Can this be entirely performed by using coding metrics visualized on the dashboard?

RQ2: While RQ1 is supported, how does it work? What can be facilitated as discoveries or actual ideas by users for improvements in their classrooms?

4.2 Education environment descriptions

The introductory programming course was designed for liberal arts students, as opposed to computer science students. Therefore, the main objective of the course

was to develop a better understanding of task-oriented programming. The objective was independent from any programming language, although Java language was used in the actual environment. Approximately 100 students participated annually in the course, which was administered by two teachers and six teaching assistants.

4.3 Use of the dashboard and analyzed discourse

We evaluated the dashboard at teaching staff meetings which were held weekly for the purpose of classroom management. Two teachers and six teaching assistants participated in the meetings, and their discussions focused on reflecting on the previous week's class and management methods for the next class. The evaluation was conducted in a 2016 course management meeting using the proposed dashboard on a shared projection screen, which showed the prior four years' data plus additional data up until the previous meeting.

All discussion sessions were recorded using a camcorder, and seven cases were selected (as described on section 5.1) in which the discussants used the dashboard in their discussions to conduct in-depth qualitative analysis.

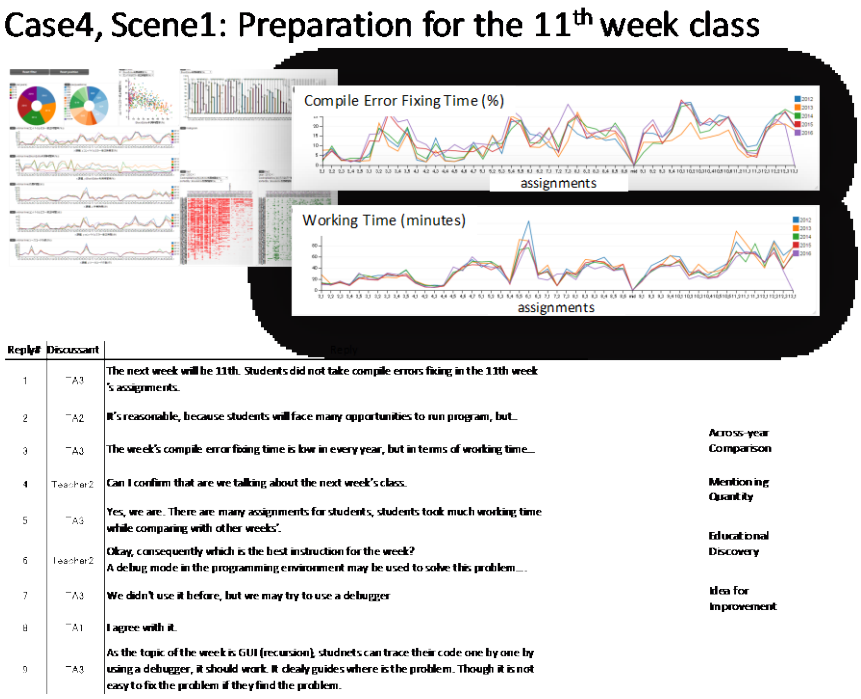

Fig. 2. An example of the coding method used in the qualitative analysis of the discourse using the proposed dashboard

4.4 Coding method

We conducted two types of analyses for the seven discourse cases. An example of the two analyses is shown in Fig 2. As for RQ1: to confirm the dashboard properly works,

we inspected the relationships between the discourse and usage of the dashboard. As shown in Fig 2, the arrows reveal the relationships, each indicating an underlined sentence where an arrow starts from and mentions the part of the dashboard the arrow points to.

Table 1. The coding scheme for RQ2: how does the dashboard promote productive discussions for teachers?

Coding Category	Brief Definition	Representative Example(s)
Across-year Comparison	A discussant is talking about an across-year comparison using particular parts of the dashboard, by comparing tw lines, or using filtering functionality.	– This year's shape of line is opposite to the previous years' – The maximum of LOC in middle assignment was around 300 until 2013, but this year...
Mentioning Quantity	A discussant is mentioning to a quantitative data which indicated in a particular part of the dashboard.	– This 20% means approximately 1 of 5 students used block editor.... – To my surprise, there is a student who wrote 12000+ lines in t assignment...
Educational Discovery	A discussant (or whole group) found the new fact or ide for educational situation including students behaviors. Tacit knowledge which is observed by teaching staff in classroom was clearly supported or unsupported by data which is shared between teaching staff as explicit knowledge.	– This student accomplished really hard work to draw his favor figure. – We can confirm the negative correlation between working tin and compile error fixing time, though its obvious.
Idea for Improvement	A discussant suggested or proposed an idea for improvement of educational situation, such as difficulty assignment, method of instruction, teaching.	– This assignment can be split into two or three small steps because.... – A debug mode 15in the programming environment may be us to solve this problem....

As for RQ2: how the dashboard promotes productive discussions for teachers, we created a coding scheme as shown in Table 1. The coding categories are comprised of Across-year Comparison, Mentioning Quality, Educational Discovery, and Idea for Improvement. The four categories are colored as shown in the legend in Fig 2, with the coded parts in the discourse highlighted in each color.

The analysis for entire discourse was performed by a single rater, who also served as a teaching assistant and participated in the teaching staff meetings. The second rater performed independently for 4 of 7 cases (57%) of the discourse. The interrater reliability was 73%, conflicts are resolved by a raters' discussion.

5. Results

5.1 Short descriptions of each case

Case 1: Effects of reordering assignments

The teaching staff discussed what the obstacles were for why students in previous courses required a longer time in completing their assignments. The teachers hypothesized that the problem was caused by the ordering of assignments, so that the teachers tried to reorder the assignments for the current year. A week later, the teachers could confirm there was improvement, as the dashboard indicated the actual working time was reduced by nearly 10 minutes. They also discovered that there were

some students who exhibited a longer compile error correction time, particularly in the target assignment, later confirming one of the reasons was their low usage of BlockEditor.

Case 2: Compile error correction time and its improvement

One teacher had felt it was difficult to maintain student motivation in some assignments in which the average compile error correction time was indicated as being longer. The teaching team isolated the type of assignments, and then realized that students tended to compile once after writing the whole program. A few suggestions were given for how to improve the situation, including dividing the assignment into some smaller steps, or teaching students to compile progressively one by one for each small part.

Case 3: Working time and maximum lines of codes

Midterm assignment: a task to create their own GUI contents (Game or other interactive contents) was discussed. The average working time was approximately five hours, with a maximum time of over 25 hours. The result was unexpectedly long in a positive way. The maximum code size was 16,000+. It was negatively evaluated because the teachers considered it was caused by copying without thinking of the abstraction of the algorithm.

Case 4: Characteristics of the assignment and improvement of instruction

The dashboard indicated a particular week's characteristics: students worked long, but took a short time to compile error corrections. The topic of the week was recursion. The team concluded to promote student use of a debugger to trace their code.

Case 5: Working time outside of classroom

The team tried to estimate how long students worked outside of classroom time. As the working time of each assignment was 40 minutes, the working time outside was estimated to be one hour. The team discussed the estimation's validity for education.

Case 6: Correlation between several coding metrics

Correlations between several coding metrics were analyzed in the discussion. For example, the fact there was no correlation between working time and lines of code was an unexpected result. After the discussion about the reason for the result, teachers reached a consensus that the assigned common tasks were well-structured, and the code size could be estimated within a common solution.

Case 7: BlockEditor usage of the 2016 year's course

Similar analyses to those of BlockEditor's research [9] were discussed. During the current year there were a few compulsory assignments using BlockEditor, although the results indicated a low BlockEditor usage ratio even during the early weeks of the course. Whereas during the prior years the usage ratios of BlockEditor were shown to also be high during the latter weeks of the course. The team also confirmed several other facts: students selected BlockEditor to reduce compile error corrections, or there were no significant differences in working time and LOC between BlockEditor users and non-users.

5.2 Results of qualitative analysis

Based on the results of the qualitative analysis of seven cases, we created a heat-map to visualize the coverage of used graphs on the dashboard. The heat-map is shown in Fig 3. In the figure, the used graphs in the discussion of each case are highlighted in transparent red.

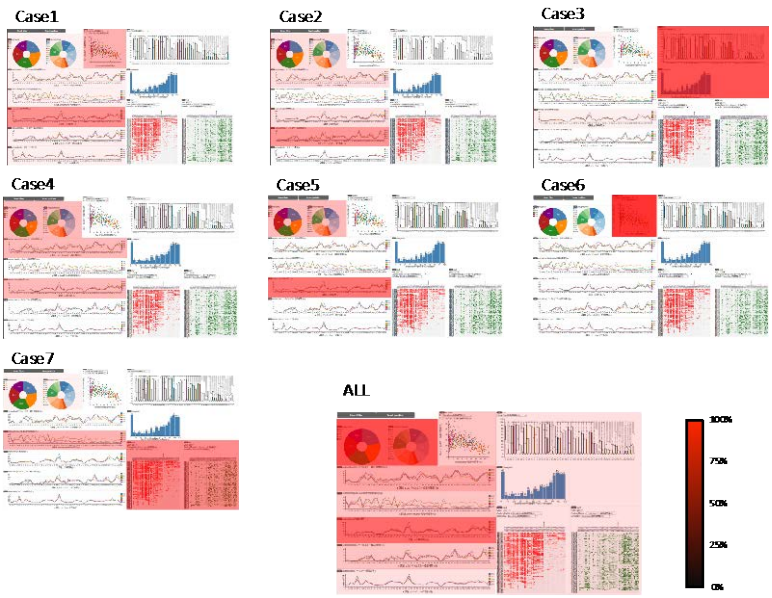

Fig. 3. A heat-map representing the coverage of graph-usage in each case. ALL shows the coverage by summing up all seven cases

Table 2. The results of analysis for RQ2: how does the dashboard promote productive discussion for teachers?

	Across-year Comparison	Mentioning Quantity	Educational Discovery	Idea for Improvement
Case1: Effects of reordering assignments	3	2	3	0
Case2: Compile error fixing time in the class and its improvement	0	0	2	2
Case3: Correlation between working time and maximum lines of codes	3	13	4	0
Case4: Characteristics of particular assinment and improvement of its instruction	0	0	2	1
Case5: Working time on outside classroom	0	2	1	0
Case6: Correlation between several coding metrics	0	0	2	0
Case7: Block-editor usage of this year's class	1	4	5	0
Total	7	21	19	3

The depth of transparency red in Fig 3 shows the percentage of usage, which is calculated by the frequencies of arrows in the discourse (Fig. 2). The heat-map of ALL indicates the graphs used in all seven cases, which is calculated by the number of cases in which the graph was used.

The results of the analysis using the described coding scheme (as shown in Table 1) are shown in Table 2. We could confirm that the four coding categories can be seen in most cases broadly. Essentially, many educational discoveries were identified using the dashboard, and the phenomenon is supported by the descriptions of each case described in section 5. 1. Although the number of Ideas for Improvement was not particularly high if compared with the other categories, these were ideas supported by the facts discovered in the Educational Discovery category.

6. Discussion

RQ1 asked "Can the dashboard facilitate teachers' productive discussion to improve introductory programming classroom? And is it able to be performed by using coding metrics visualized in the dashboard?" The results were generally positive: we could examine 100% of the coverage by use of the graphs in the seven cases by the heat-map, which were effectively used in the discourse on improving the classes. The filtering functionality was also used in 6 out of 7 cases. To be specific in each case, the use of line charts of coding metrics was popular, and the other boxplot or tile representations were used when users needed to use specific data to explore deeper facts in the classroom.

RQ2 asked "While RQ1 is supported, how does it work? And what can be facilitated as discoveries or actual ideas by users for improvements of their classroom?" A surprising result revealed by the qualitative analysis was that 19 Educational Discoveries were coded in all of the seven cases. We define Educational Discovery not only as finding a new fact or idea in an educational situation, but by whether or not the tacit knowledge observed in the classroom was clearly supported. The results indicate that evidence-based facts can be shared between teaching staff as explicit knowledge. Furthermore, misunderstandings of the classroom environment can be fixed through this process. We strongly believe that a progressive, iterative, and continuous improvement of classrooms can be attained through this process.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers 25730203, 26280129.

References

1. Griffin, P., McGaw, B., & Care, E. (Eds.). Assessment and teaching of 21st century skills. New York, NY: Springer(2012)
2. J, Wing. Computational thinking and thinking about computing, Philosophical Transactions of The Royal Society A, 366, pp.3717-3725 (2008)

3. Brennan, K., Resnick, M. New frameworks for studying and assessing the development of computational thinking. Annual American Educational Research Association Meeting, 2012.
4. Lye, S., Koh, J. Review on teaching and learning of computational thinking through programming: What is next for K-12?, Computers in Human Behavior, 41, pp.51-61(2014).
5. Tedre, M., Denning, P. The long quest for computational thinking, Proc of the 16th Koli Calling Conference on Computing Education Research, pp.120-129 (2016)
6. 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., Toll, D. Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies, ITiCSE WGR'16, pp.41-63 (2015)
7. Jadud, M. Methods and Tools for Exploring Novice Compilation Behaviour, In proceedings of the Second International Workshop on Computing Education Research ICER '06, pp. 73.84, (2006)
8. Toll, D., Olsson, T., Ericsson, M., Wingkvist, A. Fine-Grained Recording of Student Programming Sessions to Improve Teaching and Time Estimations, The International Journal of Engineering Education, 32(3A), 1069-1077 (2016).
9. Matsuzawa, Y., Tanaka, Y., Sakai, S.: Measuring an impact of block-based language in introductory programming. ifip-tc3 conference SaITE2016, (2016)
10. Dehnadi, S. A cognitive study of learning to program in introductory programming courses, Ph.D thesis, Middlesex University, (2009)
11. Lung, J., Aranda J., Easterbrook, S., Wilson, G. On the difficulty of replicating human subject studies in software engineering, ICSE'08, pp.191-(2008)
12. Matsuzawa, Y., Hirao, M., Sakai, S., Compile Error Collection Viewer: Visualization of Compile Error Correction History for Self-assessment in Programming Education, The International Journal of Engineering Education, 32(3A), 1117-1127 (2016).
13. Norris, C., Barry, F., Fenwick, J., Reid, K., Rountree, J. ClockIt: Collecting Quantitative Data on How Beginning Software Developers Really Work, ITiCSE'08, pp.37-41 (2008).
14. Murphy, C., Kaiser, G., Loveland, K, Hasan, S., Retina: Helping Students and Instructors Based on Observed Programming Activities, SIGCSE'09. pp.178-182 (2009).
15. Helminen, J., Ihantola, P, Karavirta, V., Recording and Analyzing In-Browser Programming Sessions, Koli Calling '13, pp.13-22, (2013).
16. Matsuzawa, Y., Okada, K., Sakai, S., Programming Process Visualizer: A proposal of the Tool for Students to Observer Their Programming Process, Proc ITiCSE'13, pp.46-51 (2013)
17. Alammary, A., Carbone, A., Sheard. Implementation of a smart lab for teachers of novice programmers, Proceeding ACE '12 Proceedings of the Fourteenth Australasian Computing Education Conference, pp.121-130 (2012)
18. Johnson, P., Zhang, S. We need more coverage, stat! Classroom experience with the Software ICU, 3rd International Symposium on Empirical Software Engineering and Management, pp.168-178 (2009)
19. Haig, T., Falkner, K., Falkner, N. Visualisation of Learning Management System Usage for Detecting Student Behaviour Patterns, Proceedings of the Fifteenth Australasian Computing Education Conference (ACE2013), pp.107-115 (2013)
20. Lister, R., Adams, E., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Mostrom, J., Sanders, K., Seppala, O., Simon, B., Thomas, L. A multi-national Study of reading and tracing skills in novice programmers, SIGCSE'04, pp.119-150 (2004).
21. Ford, M., Venema, S. Assessing the Success of an Introductory Programming Course, Journal of Information Technology Education, Vol.9, pp.133-145 (2010).
22. Few, S., Information Dashboard Design: Displaying Data for At-a-Glance Monitoring, O'Reilly Media, 2006.