Measurement Range Increment in a Method for Evaluating Panoramic Understanding of Programming

Dick Martinez Calderon, Kin Man, Hidenari Kiyomitsu, Kazuhiro Ohtsuki
Graduate School of Intercultural Studies
Kobe University
Kobe, Japan
137c127c@stu.kobe-u.ac.jp

Yukinobu Miyamoto, Yi Sun
Graduate School of Information Technology
Kobe Institute of Computing
Kobe, Japan

Masami Hirabayashi
Institute of Advanced Media Arts and Sciences (IAMAS)
Gifu, Japan

***Abstract*— In a prior study we introduced the *Programmed Visual Contents Comparison Method (PVCC)* for assessment of programming abilities related with *Panoramic Understanding of Programming (PUP)*. With this method, by comparing two or more output pictures produced by programming samples (a *question*), a student must decide which one of the programs producing them is more difficult to build with programming, or, if the difficulty is similar for all of them. This study reported also the results of a test to evaluate this method's validity performed with groups of students of diverse fields. Validity was verified by comparing the initial programming ability reported by programming professors of these groups with the test results; this confirmed that the PVCC method worked well to find programming abilities related with PUP. In this paper we propose an enhancement to the PVCC Method based on the preparation of New Questions where two or more samples displaying both input data and output pictures are shown. By adding input data to output pictures and focus strictly on the programming processes needed to obtain these pictures from the provided input data, we aim to broaden the range of discernible programming abilities related with PUP. The results of a test performed to verify the suitability of New Questions performed with professors of programming is also reported.**

Keywords*— Computer science education; Programming Training; Student Assessment; Graphic Design; Software Engineering

I. INTRODUCTION

During the last two decades, software development has changed drastically, more and more people not involved in professional software development have become able to do programming by using new resources [1], for example: code samples and tutorials used through copy-pasting; simplified libraries, and several visual software development tools and languages, where the programming code is hidden and it can be applied with just a click. As a concrete example of these changes we must refer to the inclusion of a significant number of disciplines previously considered non-related with computing, into curriculum guidelines [2].

Even though several studies have been proposing new systems oriented to reduce the gap between those multiple disciplines [3][4], an exhaustive search of the literature revealed few studies concerning the assessment of programming ability in this different range of fields, and particularly, of the multiple ways a student could apply programming skills according to his knowledge level or field; in other words, how a student applies a *Panoramic Understanding of Programming*.

In a prior study [5] we introduced the *Programmed Visual Contents Comparison Method (PVCC)* for assessment of programming abilities related with *Panoramic Understanding of Programming (PUP)*. With this method, by comparing two or more displayed pictures produced by programming samples (a Question), a student must decide which one of the programs producing those pictures is more difficult to build with programming, or, if the difficulty is similar for all of them.

The aforementioned study reported also the application of a test to evaluate the validity of the method to groups of students of: Graphic Design, Game Design, and IT. We examined the validity by comparing the initial programming ability reported by programming teachers of these groups with the results of the test, and we found out that the proposed method worked well to find programming abilities related with PUP.

In this paper we propose an enhancement to the PVCC Method, oriented to extend the range of identifiable programming abilities related with PUP. The main focus of this enhancement is the preparation of *New Questions* where two or more samples displaying both input data and output pictures are shown.

By adding input data to output pictures and focus strictly on the programming processes needed to obtain these pictures from the provided input data, we aim to broaden the range of identifiable programming abilities related with PUP.

Section II of this text gives an overview of the PVCC Method. Section III presents the characteristics of the enhancement to the PVCC method by proposing New Questions, and Section IV presents a discussion regarding the suitability of the New Questions based on the results from a test performed to professors of different universities in Japan.

II. Programmed Visual Contents Comparison Method Basic Definition and Previous Test to Student Groups

The PVCC method is based on the comparison of 2 displayed pictures produced by programming samples that, for our purposes we call a *Question*; if a *Question* is showed to the student taking the test, he is requested to decide which one of the samples is more difficult to build with programming than the other, or, if the difficulty is similar for both of them.

We built a Web Testing System for students of programming from different fields and with different expertise, where *Questions* including three types of samples: Static Pictures, Animated Graphics and Controlled by Mouse (sample objects can be moved or changed by hovering and clicking) were arranged.

Fig. 1 shows an example of how a *Question* was displayed on screen; in this case both samples were Animated Graphics, the sample on the left draws and erases a circle each frame, while the sample on the right draws circles each frame without erasing them.

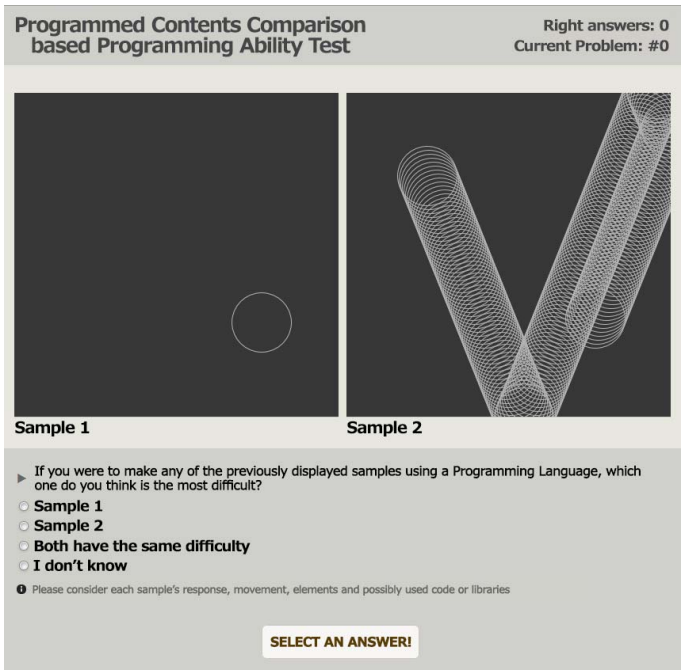

Fig. 1. Example of a Question as displayed on the Web Testing System

The person answering these *Questions* must use any experience and knowledge he could have on programming, regardless of the tools or programming languages he could know. The following Questions are a good illustration of this method.

Fig. 2 shows a *Question* where both samples are built by using the same code (an iteration process) changing only its parameters. Therefore the correct answer for this question was established to be: *the difficulty is similar.*

We would expect students who understand how the iteration process is applied on both samples to answer *the difficulty is similar*, since they would surely identify that both samples are built by using the same program only changing its parameters.

In the other hand, those students choosing one sample over the other as their answer are probably unaware of the specific programming process used to build both samples (iteration) and would probably consider their difficulty based more on screen presentation issues (scale, distance between objects, visual impression) than on how they are programmed.

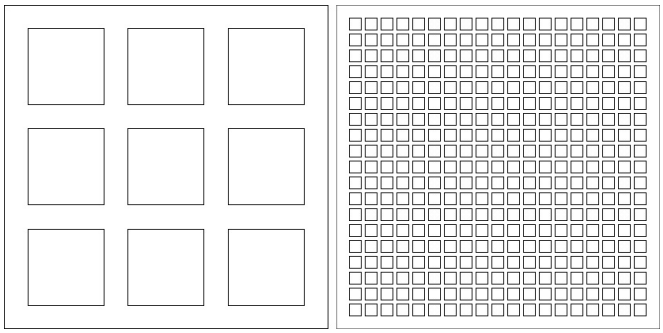

Fig. 2. Question including *Iteration* process

Fig. 3 shows a *Question* where the sample marked with (1) uses a *Hidden Line Removal* process to draw circles, while the program of the sample marked with (2) doesn't use this process, therefore the correct answer for this question was decided to be: *sample marked with (1)*.

Those students of programming knowing how difficult it is to draw circles the way they are displayed on the sample marked with (1) without using any libraries, or by using older programming languages (closer to machine language), would surely understand the difficulty of the *Hidden Line Removal* process used on the sample marked with (1).

By contrast, those students who are used to program with simplified programming languages, or by using libraries, would probably answer that *the difficulty is similar* since with those languages both samples can be produced by using the same code changing only its parameters. These students are surely unaware of what kind of algorithm is the *Hidden Line Removal* and how it is applied.

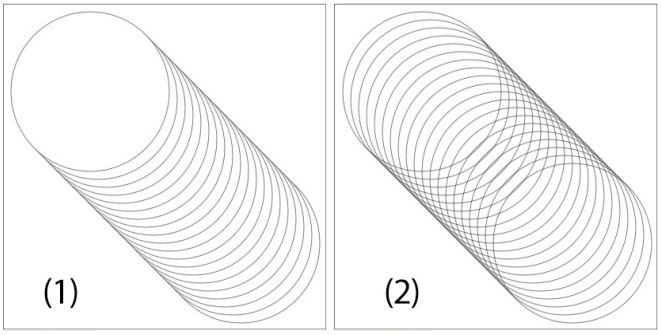

Fig. 3. Question including *Hidden Line Removal* process

In our previous study [5] we evaluated the potential of the *PVCC method* to assess programming ability. By using the aforementioned Web Testing System we applied a test to more than 120 students from different fields and with different curriculums; these students were divided into four groups

according to their field and level. The programming ability of all students was previously reported by their programming professors.

The results of the test were compared with the report on programming ability provided by programming professors. We established the result difference per questions between the groups and found that some had significant differences.

Further analysis on the test results showed their concordance with the programming ability reported by programming professors; this concordance validated the *PVCC method* to find programming abilities related with *PUP*.

III. Broadening The Range Of Identifiable Programming Abilities by a Proposal Of New Questions

The main objective for this stage of the research is to improve the capability of the *PVCC Method* to assess effectively programming abilities related with a *Panoramic Understanding of Programming (PUP)*.

The previous version of the test [5] was mainly focused in visual programming techniques and/or visual effects algorithms. Even though the emphasis on visual programming techniques was evident, some of the first test version questions categorized as *more related with programming* placed less emphasis in graphic techniques and more in basic programming processes, for example: the question shown in Fig. 2: *Nested Iteration* used very simple graphic elements (squares) and had no interactive or animated elements. For this question, students from IT and Game Development understood that its focus was on identifying what kind of process was managing the data inserted (amount of squares) for both samples.

By analyzing the questions targeted towards other programming processes different than visual techniques in the first version of the test we found that it is possible to extend the range of programming skills that the PVCC method can identify by enhancing or changing the samples compared to evaluate more and different aspects of programming.

A. New Questions Characteristics

We propose a New type of Questions where 2 or more samples are compared, but, instead of asking to compare only graphical output pictures, both input data (raw text, comma-separated data, associative arrays, XML-like data) and output (either picture or graph) are shown to the student.

For New Questions, a *Sample* consists of input data and output picture(s), and a *Question* consists of a set of samples to be compared.

Students are asked: *from which input data sample is more difficult to obtain the displayed output picture*, or *which output sample is more difficult to obtain from the given input data* depending on how many input and output samples are displayed. A new question may be composed of: multiple input data samples for a single output picture, a single input data set for multiple output pictures or multiple input data samples for multiple output data pictures.

With the first test questions the student didn't have a clear start point from where to think the problem, therefore having too many ways to go without knowing about what processes were making the difference (for example: which required longer time, or more resources, or were not optimized).

By including input data we are guiding the person through what is the intention of the comparison of samples, and we are making sure that the person who knows how to process data in programming, namely, how to read a data file knowing the format, how to store this data and how to apply it in algorithms, or other processes, will know how the input sample behaves and what kind of processes are added that are different in order to get the output samples from an input data set.

B. Examples of New Test Questions

Through the following examples we explain more in detail how New Questions are set up, what kind of programming processes are involved on each sample and how through the inclusion of input data the intention of the comparison can be identified.

Fig 4 is an example of a program to draw Pie Charts. For output samples in Fig. 4(1) and Fig. 4(2) country names, region and population values (percentage) are displayed in Pie Charts.

In Fig. 4(1) population value is ordered from minimum to maximum, while for Fig. 4(2) countries are grouped by continent and their population value is sorted from maximum to minimum.

If someone would perform both output samples comparison without having the source data used by the program, the output sample in Fig. 4(2) could be considered as the most difficult to obtain, but this person would be doubtful about what kind of data handling processes are carried on to obtain both output samples.

In Fig. 4(3) the original source input data for both samples, is written line by line including country, region and population values; this data is randomly arranged.

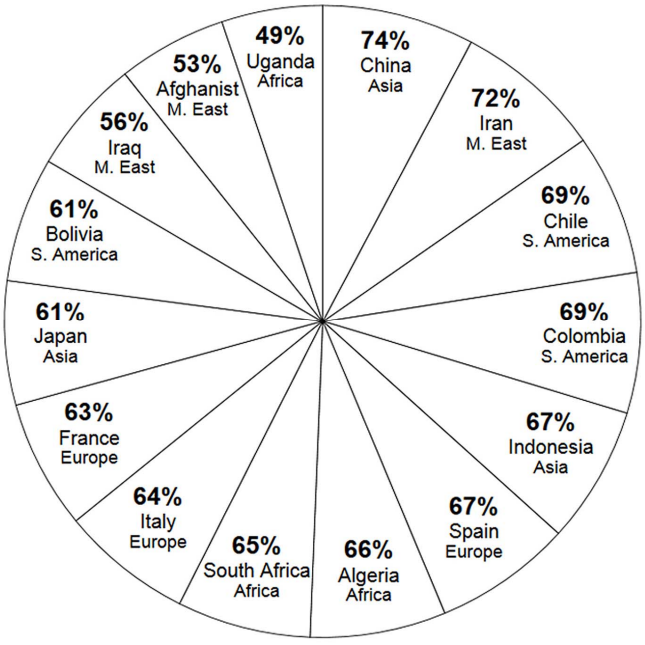

Fig. 4. (1) Pie Chart Output Sample 1

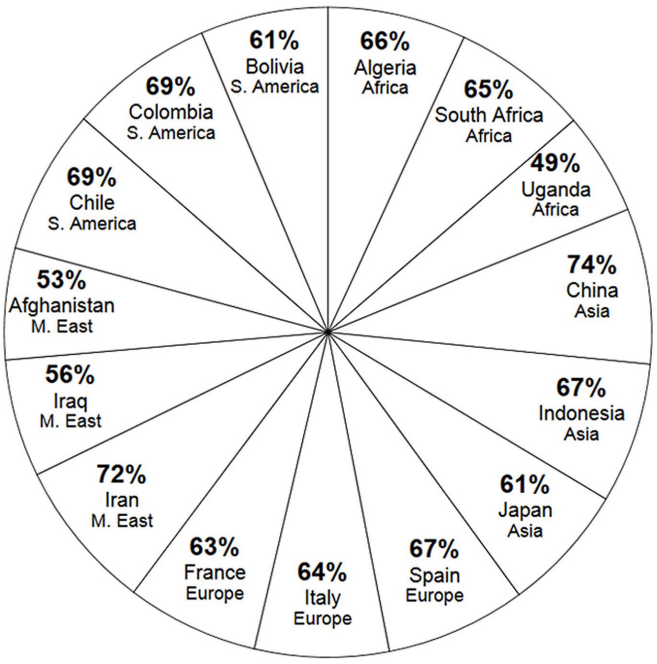

Fig. 4. (2) Pie Chart Output Sample 2

```
Population 15-64 years
Country,Region,Value
South Africa,Africa,65
Chile,South America,69
Iraq,Middle East,56
Indonesia,Asia,67
Spain,Europe,67
Uganda,Africa,49
Algeria,Africa,66
Italy,Europe,64
Iran,Middle East,72
Bolivia,South America,61
France,Europe,63
Colombia,South America,69
Japan,Asia,61
Afghanistan,Middle East,53
China,Asia,74
```

Fig. 4. (3) Pie Charts Input Data

By having the input data in Fig. 4(3) the person performing the comparison can confirm that the output sample in Fig 4(2) is the most difficult to obtain, because, the program would need to group the countries by continent and then sort the percentage value inside this groups from maximum to minimum to produce this output, while for Fig. 4(1) the program would only need to sort percentages from maximum to minimum.

But, by changing the source data to the data sample in Fig 4(4), where countries are already grouped by continent and percentage values are already sorted from maximum to minimum; the program would only need to read the data sequentially from the beginning, while for the Output Sample in Fig. 4(1) it will need to sort the percentage values from maximum to minimum anyway.

```
Population 15-64 years
Country,Region,Value
Algeria,Africa,66
South Africa,Africa,65
Uganda,Africa,49
China,Asia,74
Indonesia,Asia,67
Japan,Asia,61
Spain,Europe,67
Italy,Europe,64
France,Europe,63
Iran,Middle East,72
Iraq,Middle East,56
Afghanistan,Middle East,53
Chile,South America,69
Colombia,South America,69
Bolivia,South America,61
```

Fig. 4. (4) Additional Input Data for Pie Charts

This way, by changing the wat input data is sorted and formatted the difficulty of obtaining one or another output sample can change; in this sense input data is used in new questions to guide the sample comparison evaluation and final judgement.

The question displayed in Fig 5 was prepared to inquiry about data management through asking *from which one of the samples in Fig. 5(1) and Fig. 5(2) is more difficult to obtain the output picture in Fig. 5(3)?*

```
Country,Region,Prefecture,Area
Japan,Kanto,Gunma,6362
Japan,Kansai,Mie,5777
Japan,Kanto,Ibaraki,6095
Japan,Kanto,Tokyo,2188
Japan,Kansai,Hyogo,8396
Japan,Kanto,Kanagawa,2415
Japan,Kansai,Fukui,4189
Japan,Kanto,Tochigi,6408
Japan,Kansai,Shiga,4017
Japan,Kansai,Osaka,1899
Japan,Kansai,Nara,3691
Japan,Kanto,Saitama,3798
Japan,Kansai,Wakayama,4726
Japan,Kansai,Kyoto,4613
Japan,Kanto,Chiba,5156
Japan,Kansai,Tottori,3507
Japan,Kansai,Tokushima,4146
```

Fig. 5. (1) Circles Arrangement Input Data Sample 1

```
Japan
    Kanto
        Gunma,6362
        Tochigi,6408
        Ibaraki,6095
        Saitama,3798
        Tokyo,2188
        Chiba,5156
        Kanagawa,2415
    Kansai
        Fukui,4189
        Shiga,4017
        Mie,5777
        Osaka,1899
        Nara,3691
        Wakayama,4726
        Kyoto,4613
        Hyogo,8396
        Totori,3507
        Tokushima,4146
```

Fig. 5. (2) Circles Arrangement Input Data Sample 2

Output sample in Fig. 5(3) shows circles representing Japanese prefectures grouped by region (Kanto and Kansai), and at the same time those regions are grouped in one large group called Japan; the size of each circle represents the area value included in both data samples in Fig. 5(1) and Fig. 5(2). Input data in Fig. 5(1) contains, organized line by line: Country name, Region name, Prefecture name and Area value. All lines are randomly distributed and data is not sorted.

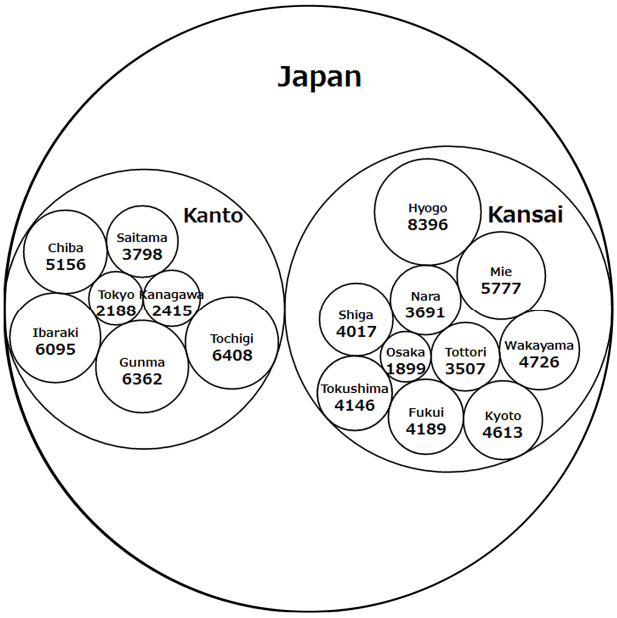

Fig. 5. (3) Circles Arrangement Output Picture

As opposed to data sample in Fig. 5(1), data sample in Fig. 5(2) is hierarchically organized. Data sample in Fig. 5(2) is easy to read (by a person) but each line has a different format.

The reason why it is more difficult to do a program to get the output picture with data sample in Fig. 5(2) is because the program will need to perform additional processes to identify the characters signaling the levels of hierarchy, and establish how many characters behind each text line represent what level; besides, in order to store the area value, the program will need to confirm for each line who is his parent and children, in the end it will also need to sort the data.

It doesn't matter what kind of program is to be done, data needs to be designed from the beginning; in the case of Fig. 5 if the input data isn't displayed together with the samples the difference on data reading processes cannot be identified.

The question displayed in Fig 6 was prepared to evaluate knowledge for coordinates mapping in maps, and it inquiries *from which program using any of the data samples in* Fig. 6*(1) and* Fig. 6*(2) is more difficult to obtain output pictures in* Fig. 6*(3) and* Fig. 6*(4)?*

Input data in Fig 6(1) includes, organized line by line, name of Japanese prefectures for the Kanto region, area value, prefecture capital city name, and its location coordinate. Input data in Fig. 6(2) contains the border line coordinates for each prefecture map.

```
Prefecture,Area,Capital,Capital Location
Fukui,4189,Fukui,(136.221642,36.065219)
Shiga,4017,Otsu,(135.868590,35.004531)
Mie,5777,Tsu,(136.508591,4.730283)
Osaka,1899,Osaka,(135.519711,34.686316)
Nara,3691,Nara,(135.832744,34.685333)
Wakayama,4726,Wakayama,(135.167506,34.226034)
Kyoto,4613,Kyoto,(135.755608,35.021004)
Hyogo,8396,Kobe,(135.183025,34.691279)
Tottori,3507,Tottori,(134.237672,35.503869)
Tokushima,4146,Tokushima,(134.559303,34.065770)
```

Fig. 6. (1) Mapping Map and Circles Arrangement Input Data Sample 1

```
Prefecture, Border line coordinates
Fukui,
(134.375851,35.608377),(134.518312,35.274
572),(134.405531,35.237561),(134.181714,3
5.166679),(134.000147,35.349643)
... (continues)
Shiga,
(134.44359,34.205071),(134.444986,34.2085
63),(134.647853,34.175741),(134.58605,34.0
32233),(134.749112,33.832509)
... (continues)
Mie,
(136.245995,36.290651),(136.342016,36.177
87),(136.757176,36.085689),(136.825264,35.
893298),(136.781967,35.795531)
... (continues)
Osaka,
(136.153465,35.694621),(136.278816,35.661
101),(136.445369,35.387004)
... (continues)
```

Fig. 6. (2) Mapping Map and Circles Arrangement Input Data Sample 2

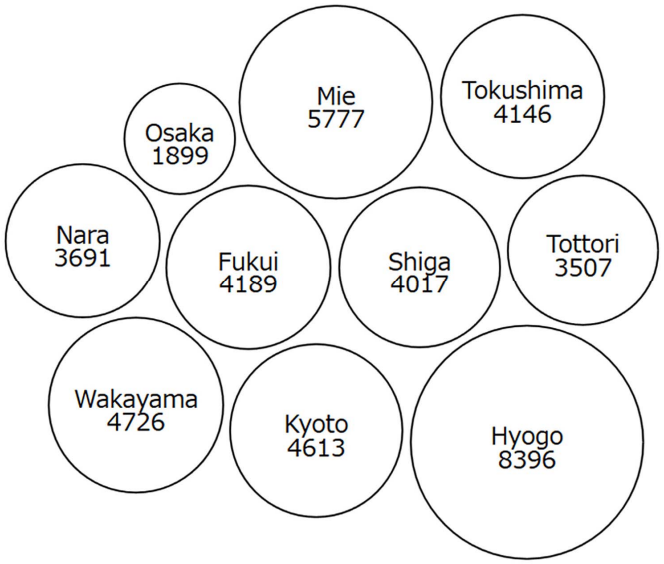

Fig. 6. (3) Circles Arrangement Output Sample 1

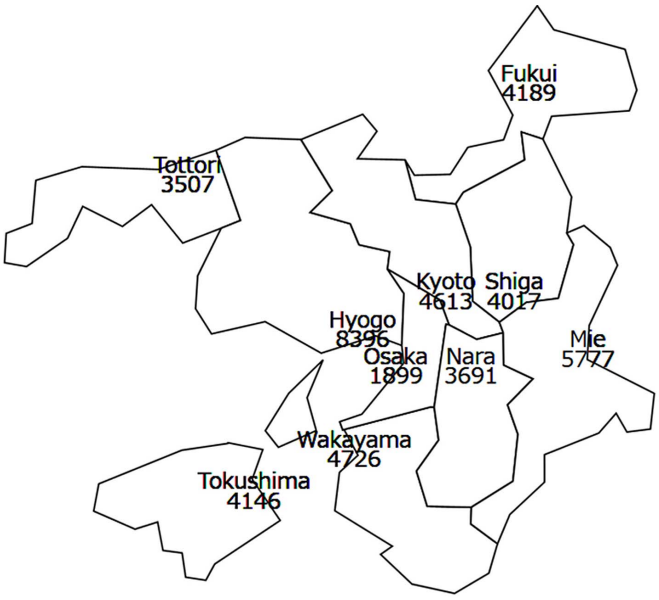

Fig. 6. (4) Mapping Map Output Sample 2

The program producing the output sample in Fig. 6(3) is using only the input data set in Fig. 6(1). It needs to assign the area value as circles size, then apply a nearest neighbor algorithm to group the circles avoiding circle intersection, additionally it has to calculate the position of the text (centroid of each circle).

The program producing output sample in Fig. 6(4) is using both input data in Fig. 6(1) and input data in Fig. 6(2). The program uses the coordinates from data sample in Fig. 6(2) to draw lines that constitutes the border of each prefecture, and the city location coordinates from data sample in Fig. 6(1) to allocate area and prefecture name texts.

By including both input data samples the person answering will understand that from those samples is used as parameters to draw the map for the second output sample, while the first sample have additional more complex processes related to data analysis to make that particular group of circles. In this sense the selected output sample should be the output sample in Fig. 6(1).

C. *Classification For Programming Processes and Identifiable Programming Abilities*

Following the previously exemplified pattern to set up questions we could divide the processes a program needs for any sample in any question into two categories: Data Processing and Data Display as shown in Table I.

Data processing related programming processes can be of three types: Data reading, Data storing and Data Analysis; in Table I each subcategory for Data processing as well as Data display contains examples of processes that belong to each one, but there could be many more programming processes for each category.

If we want to identify a programming ability through setting up a question like the ones previously explained, we can classify this programming ability inside the items of Table I and from here it can be decided what kind of processes can be emphasized in each program to be compared.

For example, the question in Fig 5 is designed to identify programming abilities related with data management, specifically data reading and data storing. To be precise, the person being able to identify the difference between the two programming samples will surely have the skills to understand how the data format is built (the first data format can be read in an easier way than the second hierarchical data format) how the data is divided into strings (the first data sample will be divided by identifying the comma, while the second data sample needs to analyze each line to identify the strings), and that the program processing the second data sample will need more reading steps to build the hierarchy in lists or structs.

In summary, the classification presented in Table I is not only for programming processes but it defines the programming abilities identifiable with New Questions.

IV. Test Oriented to Professors to Verify the Suitability of New Questions

We wanted to know if any of the New Questions were suitable to identify programming abilities related with PUP in the categories mentioned before (namely: Data Processing and Data Display). For this purpose we built a second version of the test specifically oriented to obtain feedback from programming professors regarding the inclusion of input data, the division in categories and the comparison of the most difficult programming processes.

Programming professors from universities and technical colleges in Japan answered this test in the sequence described as follows:

TABLE I PROCESSES INCLUDED IN QUESTIONS DIVIDED BY CATEGORIES

Data Processing: every algorithm or programming process that involves reading a data file and storing and handling its data falls in this category, having three subtypes:	**Data Reading** for example:	Read specific data formats (tables (e.g. raw text, comma-separated data, associative arrays etc.)
		Execute data Input/Output processes (e.g. I/O functions depending on the programming language)
		String operations or functions for dividing data into manageable strings, identify the most common characters etc.
	Data Storing for example:	Setting up and storing in data structures (e.g. arrays, nested arrays, lists etc.)
		Making references to elements in lists (e.g. pointers)
		Keeping count of the times a process is executed etc.
	Data Analysis for example:	Sort processes (algorithms or functions)
		Different ways to analyze and prepare data (Pattern Recognition, Data mining, complex data structures like Binary trees) etc.
Data Display: Every algorithm or programming process that involves the visual or graphic display or arrangement of data falls into this category, for example:		Plotting graphic objects (e.g. Bezier lines, Circles, 3D objects etc)
		Scaling and transforming graphic objects (e.g. translate, rotate)
		Allocate an object in a coordinate.
		Assign graphic object properties like color, size etc. from data.

1. Without knowing our assumed answer beforehand, they answered the question: *which output sample is more difficult to obtain from the given input data* or *from which input data sample is more difficult to obtain the displayed output picture* depending on how many input or output samples did the question have.
2. After answering the previous question, they read an explanation about the processes to identify per sample and which process is assumed as the one marking the difference for the whole question; knowing this information they told us if they agree or disagree with our assumption.
3. They confirmed if the "new question" is or is not appropriate to be included in a test to identify PUP related programming abilities.
4. Finally, we offered them the opportunity of comment freely about the samples used or the whole question.

V. ARE THE NEW QUESTIONS SUITABLE FOR IDENTIFYING PROGRAMMING ABILITIES RELATED TO PUP?

Table II and III show a summary of how many questions received the respective percentage of matching answers from professors and how many received the totally opposite answer; for example, if 6 questions correspond to an 88% in Table I that means that 6 questions were answered right by the 88% of professors. And if 6 questions correspond to 0% in Table II that means that 6 questions didn't received totally opposite answers at all by any of the professors.

Without knowing the assumed answer beforehand, more than 75% of the total of professors' answers matched the assumption regarding the programming processes through which the difference on the comparison for each question can be determined.

Within the 25% of not matching answers, more than 70% were "the difficulty is similar" this could indicate that, even when the data is present, it is incomplete or is not working as expected to establish the difference between the samples.

TABLE II. AMOUNT OF QUESTIONS RECEIVING MATCHING ANSWERS (HIGH PERCENTAGE RANGE)

percentage of matching answers received	amount of questions
100%	1
88%	6
75%	1
63%	2
50%	0

TABLE III. AMOUNT OF QUESTIONS RECEIVING TOTALLY OPPOSITE ANSWERS (WITHIN THE LOW PERCENTAGE MATCHING RANGE)

percentage of totally opposite answers received	amount of questions
0%	6
13%	4
25%	4
38%	0
50%	0

Also, within the 25% of not matching answers, more than 80% after reading the explanation considered the question

appropriate to identify programming abilities, these aspect is likely to be related with a lack of concision in data or graphic samples; probably these elements for themselves aren't enough to guide a person towards identifying the required difference between programs.

Table IV shows how many questions were considered appropriate by professors and Table V displays how many questions needed adjustments; for example, if for Table IV 2 questions correspond to an 88% that means that 88% of the professors considered 2 questions as appropriate, and if in Table V 4 questions correspond to a 13% that means that 4 questions were considered as needing fix by the 13% of the professors.

For each question needing adjustments professors suggested specific modifications, what follows is a summary of the most important of them.

TABLE IV. AMOUNT OF QUESTIONS CONSIDERED APPROPRIATE BY A HIGH PERCENTAGE OF PROFESSORS

percentage of "appropriate" answers	amount of questions
100%	2
88%	2
75%	3
63%	0
50%	1

TABLE V. AMOUNT OF QUESTIONS PROFESSORS CONSIDERED AS NEEDING FIX

percentage of "need fix" answers	amount of questions
0%	3
13%	4
25%	1
38%	2
50%	3
63%	1

It is necessary to define the difference between writing algorithms and writing the programs for each question. There could be complex algorithms that could make a difference depending on the input data, but making the output pictures could be as difficult (long, tedious) as writing the data processing algorithms. This issue can be solved by doing programs that could produce simple output pictures by using the same type of input data samples including simpler data.

Some of the questions in this test included very difficult to read or interpret, data and graphs. There are comparisons where the difference cannot be identified, not even by reading and interpreting the input data. In the other hand some comparisons are too obvious, or the difficulty difference can be easily identified by anyone, doesn't matter if he has or not experience on programming.

This could be solved by balancing the complexity and concision of questions, the structure of the data sample could be the same using less data columns, and output pictures don't need to be charts or graphs necessarily. If the objective of including data is to guide towards identifying the difference and make sure people can understand the intention, it is fundamental to make data concise.

CONCLUSION

In this paper we propose an enhancement to the *Programmed Visual Contents Comparison Method (PVCC)*. This enhancement is based on the proposal of new questions oriented to extend the range of identifiable programming abilities related to *Panoramic Understanding of Programming (PUP)*.

Through the addition of input data to comparisons of output sample pictures we were able to build appropriate questions to identify programming abilities.

Even when some of the proposed new questions presented different kind of misleadings, professors identified mistakes and points of confusion and suggested optimal ways to correct them.

Additionally, results from a questionnaire performed at the end of the test showed a positive feedback regarding the PVCC method and new questions. Professors thought that this method can definitely be used to evaluate programming abilities, and, through the arrangement of new questions as proposed in this paper, there is a high probability of evaluating more programming abilities.

ACKNOWLEDGEMENT

We would like to thank all the Professors and Students from the Graduate School of Intercultural Studies Kobe University, the College of Computing of Kobe Institute of Computing and the Institute of Advanced Media Arts and Sciences who through their participation and collaboration made possible the realization of this project.

REFERENCES

[1] M. Burnett and B. Myers, "Future of End-user Software Engineering: Beyond the Silos," in *Proceedings of the Future of Software Engineering Conference (FOSE 2014)*, New York, NY, 2014.

[2] ASSOCIATION FOR COMPUTING MACHINERY (ACM); ASSOCIATION FOR INFORMATION SYSTEMS (AIS), Curriculum Guidelines for Undergraduate Degree Programs in Information Systems (IS 2010), 2010.

[3] F. Kurzat, K. Miso, J. Zimmerman, S. Oney and B. Myers, "How to Support Designers in Getting Hold of the Immaterial Material of Software," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*, New York, NY, 2010.

[4] B. Myers, S. Young Park, Y. Nakano, G. Mueller and A. Ko, "How Designers Design and Program Interactive Behaviors," in *Proceedings of the 2008 IEEE Symposium on Visual Langauges and Human-Centric Computing (VLHCC '08)*, Washington, DC, 2008.

[5] D. Martinez Calderon, M. Kin, H. Kiyomitsu, K. Ohtsuki and Y. Miyamoto, "An Evaluation Method for Panoramic Understanding of Programming by Comparison with Visual Examples," in 2015 IEEE *Frontiers in Education Conference (FIE 2015)*, El Paso, TX, 2015.