Object-Oriented Programming: Diagnosis Understanding by Identifying and Describing Novice Perceptions

Henry Julie
Namur Digital Institute
University of Namur, Belgium
julie.henry@unamur.be

Dumas Bruno
Namur Digital Institute
University of Namur, Belgium
bruno.dumas@unamur.be

Heymans Patrick
Namur Digital Institute
University of Namur, Belgium
patrick.heymans@unamur.be

Leclercq Tony
Namur Digital Institute
University of Namur, Belgium
tony.leclercq@unamur.be

***Abstract*—This work in progress focuses on oriented-object programming (OOP) education. The purpose of this contribution is to validate a methodology for building a concept inventory (CI) dedicated to OOP. The long-term study presented here concerns the variable programming concept. Teaching programming is a major issue due to the misconceptions of the students, notably in the OOP paradigm. Teachers need to be aware of the existence of these misconceptions, their variety and when they appear in order to quickly and efficiently correct them. A CI can measure students' understanding of a set of concepts at a precise time. To measure the evolution of student's understanding over a semester, the CI has to be administrated several times along a specific timeline. In the context of a 60-hour OOP course, a preliminary qualitative study was conducted during the 2018-2019 academic year. Semi-structured interviews were conducted with 13 students following the OOP course. Each student was interviewed three times over a semester: Before the course, at mid-term over the semester, and after the course. Six students' perceptions of the variable programming concept were identified and used to build the OOP CI. During the 2019-2020 academic-year, the CI was administred to 107 students following the OOP course. Three administrations were organized according to a specific timeline, aiming both to confirm the results of the preliminary study and to quantify the occurrence of the six perceptions in a class of students. Based on the results obtained, the methodology appears to be effective in building a OOP CI.**

***Index Terms*—Misonceptions, Novice programmers, Concept inventory, Long-term study, Mixed methods**

I. Introduction

Despite a large amount of available resources, many students find learning programming very difficult. According to Ben-Ari [1], one of the reasons for this frustration is that student have to build their (own) mental models while having to tackle potential misperceptions [2]. Perceptions, that are already there when a concept is taught, are likely to influence the learning of that concept. Therefore, the teacher needs to be aware of students' perceptions to teach more effectively and provide appropriate intervention in the future [3], [4]. This awareness can be achieved using a diagnostic tool, the concept inventory (CI).

A CI is *"a research-based multiple-choice test that seeks to measure a student's knowledge of a set of concepts while also capturing conceptions and misconceptions they may have about the topic under consideration"* [5]. It can be used, among other things, to identify appropriate teaching/learning activities, to assess the impact of a change in teaching methods on student understanding, to inform students of their weaknesses, or to assess overall learning.

CIs exist for core concepts in procedural programming [6], [7]. They have been developed based on misperceptions present in learners. In object-oriented programming (OOP), many studies deal with students' misconceptions of core concepts related to this paradigm [8]–[11]. However, there is not yet a CI dedicated explicitly to these concepts.

The purpose of this contribution is to validate a methodology for data collection and analysis within a study aimed at the development of a CI dedicated to OOP. This CI aims at measuring students' understanding of the core concepts in OOP, and their evolution over a given period.

The study began in January 2019 and was complete in June 2020. It ran over two academic years. During the first year, transcripts of semi-structured interviews (verbatims) were coded. A first part of the analysis focused on the variable programming concept in the context of OOP. As a result, descriptive categories reflecting students' perceptions were defined. These categories form the basis of the first draft of a CI submitted during the second academic year to over 100 students.

II. Context

An Object-Oriented Design and Programming (OODP) course held in the second year of a bachelor's degree at the university is the measurement ground for the study described in this article. This course consists of 30 hours of lectures and 30 hours of practical sessions. Its objective is to teach students the core concepts of OOP. These are illustrated and implemented using the Java language. To enter the OODP course, students must have acquired basic programming skills with an introductory course in procedural programming given during the first year of their degree. Therefore, the working hypothesis is that every student enrolled in the OODP course knows the core concepts of programming and can implement them through writing a computer program.

III. Methodology

At the time of writing this paper, the study is a work in progress in its final year. During the first year (Phase 1), a phenomenographic approach [12] was used to discover how students perceived the concept of variables in computer programming in the OODP course. During one semester, from February 2019 to May 2019, a semi-structured interview was conducted with a group of students enrolled in the OODP course. Each student was interviewed three times.

During the second year (Phase 2), the CI is submitted to students following the OODP course.

A. Participants

The heterogeneity of our students presents an interesting diversity in the context of studying the behavior of novice programmers.

For Phase 1, the sample is composed of 13 students from three majors: 8 students in Computer Science (CS), and 5 students in Business Engineering (BE).

For Phase 2, the sample is composed of 107 students from three majors: 58 students in Computer Science (CS), 48 students in Business Engineering (BE), and one student in Mathematics (M).

B. Data Collection and Analysis

The study follows the model of pre-test-post-test evaluations [13]. In **Phase 1**, each participant participated in three interviews based on a single guide. These interviews were scheduled before the start of the course (pre-interview), midway through the semester (mid-interview), and at the end of the course before the final exam (post-interview).

The interview guide draws on both the misconsceptions identified among novice programmers by Sorva [2] and the content of the OODP course. The interviews were fully recorded and transcribed. Paper and pencil were provided for the students to use if they wished to support their speech with a diagram. The verbatims obtained were encoded according to the three steps prescribed by Lejeune [14]: an open encoding, an axial encoding, and finally a selective encoding. In the context of this work in progress, the analysis focused on the questions about the variable concept. Categories of statements were identified and inspired the first version of CI.

In **Phase 2**, this CI, composed of 12 statements of the variables concept, was submitted to students. They were asked to position themselves via a Likert scale on a series of statements around the variable programming concept. The values of the Likert scale are: “Entirely true”, “True, but to be expanded”, “Partially true, it contains errors”, “Entirely false”, and “I don’t know”. The proposed statements are based on the definition given to students in the introductory programming course they followed, the statements defined in Phase 1 of this study, but also on misconceptions defined in the literature [2].

The CI is intended to be evolutionary, i.e., the proposed statements may evolve according to the time the CI is submitted to students. These statements respect the evolution of perceptions measured during Phase 1.

IV. Perceptions of the Variable in Programming

A. Phase 1: Interviews

The results presented in this section are derived from the encoding of 39 verbatims, corresponding to interviews with 13 students, and concerning the variable concept in OOP. Six categories of statements were identified in the transcripts. Their occurrences vary according to the time of the interview (pre-, mid-, or post-).

1) Pre-interviews: Before the course, the student clings to his/her previous knowledge: his/her perceptions of the structure and functioning of a computer, and mathematics or programming notions (Table I). In category **var-a**, the students discuss the location/storage of a variable potentially in a memory space: *“We store a data item in the ram[...] the PC’s random access memory”*, *‘It’s a reference to something in the memory[...]”*. In category **var-b**, a comparison can be made with the concept of variable in mathematics, *“for equations [...]”*, or previous learning in programming, *“The same variable as the one seen last year [...] in the introductory programming course [...] it is a data value, it can be a string of characters”*. In category **var-c**, the students stay on the global idea they have of a programming course: *“a variable is code”*.

2) Mid-interviews: During learning, the category **var-c** is always present : *“a variable is something that takes up space in the program, in the memory’*. The category **var-a** evolves and the category **var-b** disappears in favor of the categories **var-d** and **var-e** (Table I). The category **var-d** is an evolved version of the category **var-a** where it is specified that the storage is done for a certain purpose: *“It is a storage space of a data value to make a calculation”*. The category **var-e** is the direct (but incomplete) reuse of the conceptual model transmitted by the teacher: *“So it is either an instance of an object or a primitive type and it corresponds to a memory area [...] Primitive types are in the stack [...]”*.

3) Post-interviews: After learning, the **var-c** category disappears while the **var-e** category evolves. The category **var-d** is still present: *“A variable is the storage of data that can be used to make calculations or simply stored for use afterwards”*. A new category appears (Table I). The category **var-f** is an evolution of the category **var-e** where the teacher’s conceptual model is appropriated by the student and thus becomes his/her mental model: *“A variable contains a data value and is identified by its name. It can be either a primitive type variable or a data type variable. If it is a data type variable, it has a reference [...] If it is a primitive type variable, it will be in the stack, otherwise in the heap”*.

B. Phase 2: Concept Inventory

The results presented in Figures 1 and 2 are those obtained by the CI submission to 107 students. The question concerning the variable needed students to position themselves in relation to 12 statements (Table II). These statements find their sources in the statements defined from the verbatims (Phase 1), in the definition given to students in the introductory programming course, but also in literature misconceptions [8]–[11].

var-a	A variable allows storing a data value in the memory of a computer.
var-b	A variable in programming is similar/assimilated to a mathematical variable.
var-c	A variable is a portion of code in a program.
var-d	A variable allows storing a data value in the memory of a computer (category var-a) in order to use it (read, write).
var-e	A variable is in the stack
var-f	A variable is in the stack (category e) and may have a reference to the heap.

TABLE I
PERCEPTIONS OF THE VARIABLE IN PROGRAMMING

	Proposed Statements	Source
var-1	A variable allows storing a data value in the memory of a computer.	var-a
var-2	A variable in programming can be assimilated to a mathematical variable.	var-b
var-3	A variable is the component of a function, allowing to vary the function.	var-b
var-4	A variable is a portion of code in a computer program.	var-c
var-5	A variable is used to manipulate a storage area.	var-d
var-6	A variable is a portion of code that can be referenced in a program.	Inspired by var-c and var-d
var-7	A variable is a memory address that is named and referenced in a program.	Inspired by var-d
var-8	A variable contains a unique value.	Introductory programming course
var-9	A variable associates a name with a value.	Introductory programming course
var-10	A variable can be modified.	Introductory programming course
var-11	A variable contains a history of values that can be used by a program.	Literature misconception
var-12	A variable is a pointer.	Literature misconception

TABLE II
STATEMENTS ABOUT THE VARIABLE IN PROGRAMMING IN THE CI

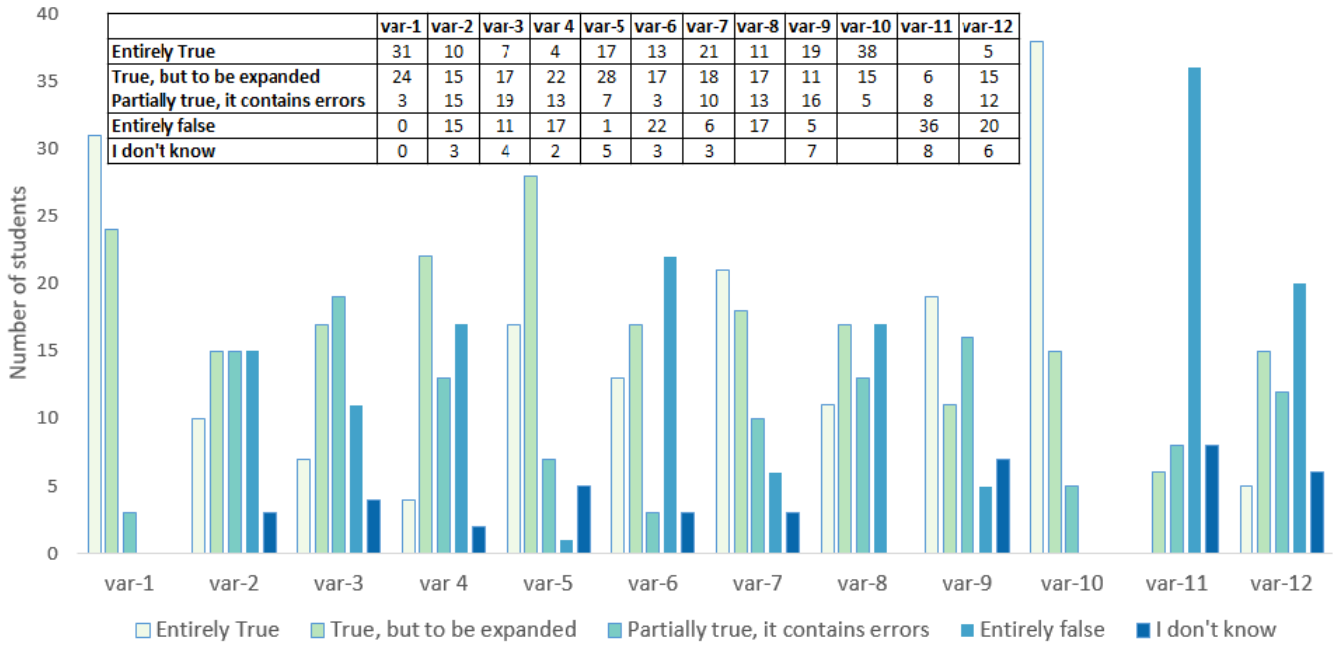
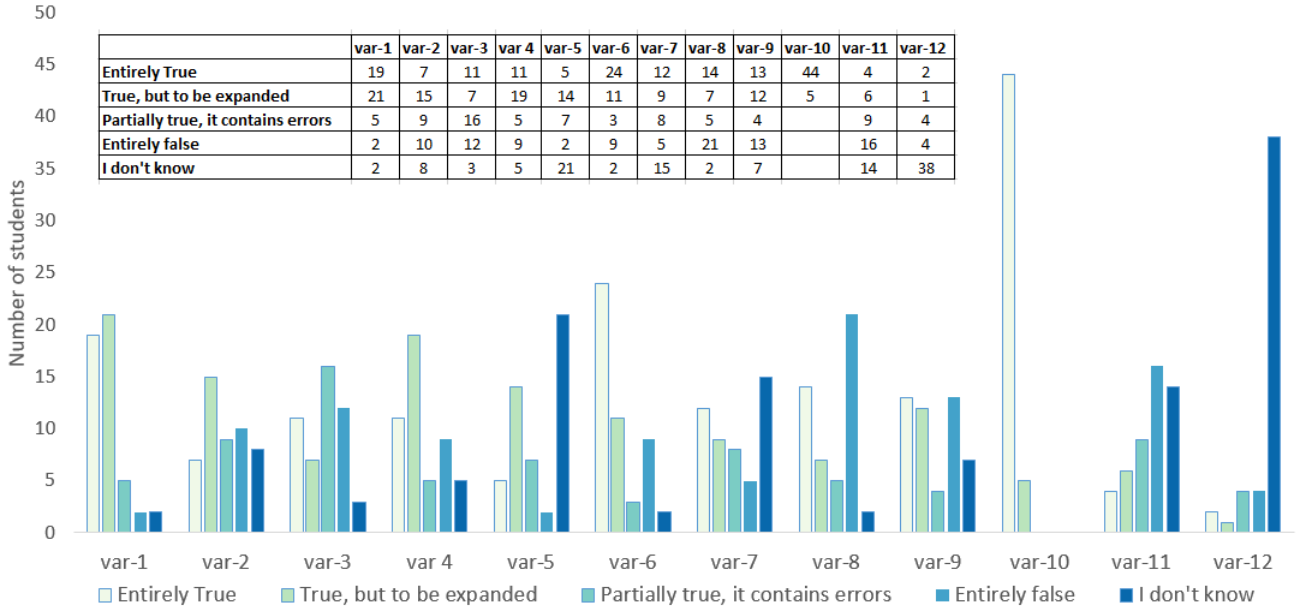


Fig. 1. Results of CS major students (n = 58)

The **var-1** statement is considered entirely true or to be expanded by a large majority of students. This confirms the Phase 1 results regarding the **var-a** statement.

The results concerning statements related to mathematics (**var-2** and **var-3**) show that there is still confusion between programming and mathematical variables for more than 2/3 of the students. This confirms the Phase 1 results regarding the **var-b** statement.

The **var-4** statement is more confusing to CS students, probably due to a too vague question. Nevertheless, this confirms the Phase 1 results regarding the **var-c** statement. This confusion is also measured at the level of the **var-6** statement. 1/3 of CS students label it as entirely false, while 50% of BE/M students consider it to be entirely true.



	var-1	var-2	var-3	var 4	var-5	var-6	var-7	var-8	var-9	var-10	var-11	var-12
Entirely True	19	7	11	11	5	24	12	14	13	44	4	2
True, but to be expanded	21	15	7	19	14	11	9	7	12	5	6	1
Partially true, it contains errors	5	9	16	5	7	3	8	5	4		9	4
Entirely false	2	10	12	9	2	9	5	21	13		16	4
I don't know	2	8	3	5	21	2	15	2	7		14	38

Fig. 2. Results of BE/M major students (n = 49)

Nearly 40% of BE and M students do not position themselves with regard to the **var-5** statement. This result shows that the understanding of the CS students is more ”concrete”, linked to their knowledge of the machine. The students in BE and M do not follow any course about the functioning of a machine. This pattern of behavior is confirmed by the positioning of the students with respect to the **var-7** statement.

The **var-8** and **var-9** statements, linked to the theoretical knowledge of the introductory programming course, reponses are scattered among the different possibilities. These statements seem to be difficult for students to interpret or these points need to be better taken into account when learning. In contrast, the students seem to be quite confident about the **var-10** statement.

Only 1/3 of BE/M students consider the **var-11** statement to be false, against 2/3 of CS students.

More than 3/4 of BE/M students do not know how to position themselves in relation to the **var-12** statement. More than 50% of the CS students feel that the **var-12** statement is true, or partially. This result is a testament to the greater experience of this specific audience, even though students all count the same number of hours of official classes.

The results obtained while writing this paper show the difference between the two audiences (CS students and BE/M students). Being aware of these differences can help to re-establish a balance and ensure a better understanding for any student, whatever his/her major.

V. Conclusion and Future Work

The validation of a methodology for building a concept inventory (CI) dedicated to object-oriented programming concepts is at the heart of this contribution. This CI aims to measure students’ understanding of OPP concepts and, more importantly, to visualize the evolution of this understanding over time. The long-term study described here concerns the variable programming concept and extends over two years.

In the context of a 60-hour OOP course, a preliminary qualitative study was conducted during the first year. Semi-structured interviews were conducted with 13 students following the OOP course. Each student was interviewed three times over a semester: Before the course, at mid-term over the semester, and after the course. Six students’ perceptions of the variable programming concept were identified.

These six perceptions were used to establish 12 statements of a CI. It was administrered to 107 students following the OOP course during the second year. At the time of writing this article, two additional administrations are planned, following a timeline similar to the interviews. However, the obtained results show that the CI confirm results from the first year’ pre-interviews. It should therefore be able to highlight when misunderstanding exist during learning and how they evolve.

The data collection has to continue during this second year of study and the CI has to be finalised for the variable programming concept. Then, the methodology has to be extended to all OOP concepts. A generalization of the CI should also be envisaged, through its implementation in other teaching contexts.

REFERENCES

[1] M. Ben-Ari, "Constructivism in computer science education," *SIGCSE Bull.*, vol. 30, no. 1, pp. 257–261, Mar. 1998.

[2] J. Sorva *et al.*, *Visual program simulation in introductory programming education*. Aalto University, 2012.

[3] Y. Qian, S. Hambrusch, A. Yadav, S. Gretter, and Y. Li, "Teachers' perceptions of student misconceptions in introductory programming," *Journal of Educational Computing Research*, p. 0735633119845413, 2019.

[4] C. Taylor, D. Zingaro, L. Porter, K. C. Webb, C. B. Lee, and M. Clancy, "Computer science concept inventories: past and future," *Computer Science Education*, vol. 24, no. 4, pp. 253–276, 2014.

[5] L. Wittie, A. Kurdia, and M. Huggard, "Developing a concept inventory for computer science 2," in *Frontiers in Education Conference (FIE)*. IEEE, 2017, pp. 1–4.

[6] A. Luxton-Reilly, B. A. Becker, Y. Cao, R. McDermott, C. Mirolo, A. Mühling, A. Petersen, K. Sanders, J. Whalley *et al.*, "Developing assessments to determine mastery of programming fundamentals," in *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*. ACM, 2018, pp. 47–69.

[7] A. E. Tew, "Assessing fundamental introductory computing concept knowledge in a language independent manner," Ph.D. dissertation, Georgia Institute of Technology, 2010.

[8] L. C. Kaczmarczyk, E. R. Petrick, J. P. East, and G. L. Herman, "Identifying student misconceptions of programming," in *Proceedings of the 41st ACM technical symposium on Computer science education*. ACM, 2010, pp. 107–111.

[9] J. Sorva, "The same but different students' understandings of primitive and object variables," in *Proceedings of the 8th International Conference on Computing Education Research*. ACM, 2008, pp. 5–15.

[10] ——, "Students' understandings of storing objects," in *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88*. Australian Computer Society, Inc., 2007, pp. 127–135.

[11] A. Eckerdal and M. Thuné, "Novice java programmers' conceptions of object and class, and variation theory," in *ACM SIGCSE Bulletin*, vol. 37. ACM, 2005, pp. 89–93.

[12] J. T. Richardson, "The concepts and methods of phenomenographic research," *Review of educational research*, vol. 69, no. 1, pp. 53–82, 1999.

[13] M. Shuttleworth, "Pretest-posttest designs," https://explorable.com/pretest-posttest-designs, retrieved Jun 18, 2019.

[14] C. Lejeune, *Manuel d'analyse qualitative*. De Boeck Supérieur, 2019.