BlueJ-UML: Learning Object-Oriented Programming Paradigm using Interactive Programming Environment

Jacky Keung, Yan Xiao, Qing Mi, Victor C. S. Lee
Department of Computer Science
City University of Hong Kong, Kowloon, Hong Kong
Email: {Jacky.Keung, csvlee}@cityu.edu.hk, {yanxiao6-c, Qing.Mi}@my.cityu.edu.hk

***Abstract*—Most of the students coming from various different backgrounds find software programming a difficult subject to learn and master, especially in learning the concept of object-oriented programming. Because they must be able to model physical beings into virtualized objects and define complex object relationships in their designs for object interactions in a very abstract fashion that is not easily understood. This study is an attempt to introduce a unique interactive learning platform to students learning Java programming, which was designed with a set of object-oriented programming pedagogical considerations in mind. This study further extends and enhances with model-driven software development technique, such that to supporting unified modelling language (UML) class diagrams and code fragments generation in order to ease the learning needs of students, a unique way to learn fundamental programming concepts. This paper introduces a new online interactive platform and environment called BlueJ-UML, which is to help students to learn and practice object-oriented programming in class. It also evaluates the success outcome of the proposed new learning method through a Technology Acceptance Model Framework, and that followed by a comprehensive statistical analysis to assess improved academic performance of the students. The result was encouraging: student programming capability had been significantly elevated and was positively correlated to their overall perception and adoption of this new technique introduced in the class.**

***Index Terms*—Interactive Programming Environment, Object-oriented Programming, Technology Acceptance Model, Online Platform and Environment**

I. Introduction

Object-oriented programming has been introduced in universities many years ago. Teaching object-oriented programming, however, remains difficult nowadays largely due to its complexity and abstraction of object orientation. It is generally included as a computer science introductory course, but there are limited teaching tools available particularly for pure novices [1]. BlueJ [2] is especially developed for teaching and learning object-oriented programming. Teaching design of object-orientation is more complex, novices not only need to learn the design principles, but also how they can realise in modelling as well as implementation. Modern Unified Modelling Language (UML) [3] can help model system design and reveal the principles to students, and it is the focus of this study. The original development of BlueJ integrated development learning platform directly represents an instant view of class diagram for a project of java source, however it does not offer a clear connection between modelling to implementation and lacks details for student to apprehend the fundamental basis of object abstraction.

This study develops an extended teaching tool based on BlueJ, called BlueJ-UML, that extends and enhances the original BlueJ platform, and to strengthen this weak connection and provide a valid pedagogical approach for novices to master the basic ideas of system design. According to UML specification [4], BlueJ-UML implements attribute, method, and constructor information in a class diagram. The original use-dependency in BlueJ was extended to stronger association, aggregation, and composition dependency. Multiplicity, role, and direction are also included in class diagram. BlueJ-UML, in addition, strengthens the interactions between student and BlueJ by providing manipulation on class diagram through UML notations to source code. It is an online platform and environment to help novices learn object-oriented programming paradigm that is publicly available.

An empirical assessment on the student acceptance of BlueJ-UML using the Technology Acceptance Model [5] has been carried out. Our results show that 1) the new pedagogical approach proposed in teaching object-oriented programming using BlueJ-UML in programming courses is well appreciated, 2) extending and implementing UML diagramming capabilities facilitate the important learning needs of essential object-oriented concepts within the learning platform. BlueJ-UML is currently available to general public for teaching object-oriented programming[1].

II. Motivation

Most of the students coming from various different backgrounds find software programming a difficult subject to learn and master, even with the reduced language syntactic complexity of the modern object-oriented programming languages such as Java [6]. In fact, many teachers found teaching object orientation more difficult than the traditional procedural programming, as the pedagogy of object-oriented program design is fundamentally different from traditional ways of teaching and learning programming and design, and it also requires

[1] http://bluej.cs.cityu.edu.hk

978-1-5386-7209-9/18/$31.00 ©2018 IEEE
DOI 10.1109/ISET.2018.00020

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on October 06,2026 at 08:29:55 UTC from IEEE Xplore. Restrictions apply.

strong Abstract Thinking capability of the students. Students must be able to model physical beings into virtualized objects and define complex object relationships and their interactions in a very abstract fashion that is not easily understood, and to describe computation using object-oriented programming languages.

Teaching Object-Oriented Programming in Computer Science should not be intrinsically more complex, but current challenges are lack of appropriate learning tools and pedagogical teaching experience of this programming paradigm with abstract thinking for Computer Science students. Existing programming development environments such as Eclipse[2] are professional software development tools commonly adopted by many computer science courses. Its complexity and advanced functionalities were actually designed for professional users with years of software development experience, which deters our students from learning fundamental concepts of object-oriented programming.

The other conceptual learning challenge with teaching Java or object-oriented programming perhaps is the large number of circular dependencies of programming language concepts and constructs, such as in object-oriented programming [7] involving UML class diagrams to the complex coding constructs using Java programming language. All of these characteristics will challenge the first few weeks of an introductory object-oriented programming course.

III. Objectives of BlueJ-UML

Unified Modelling Language (UML) is commonly taught in system design courses. It is a modelling language to model a system, and present a design of the system. Experienced developers corporately design and build computer systems with UML. It is well designed for this specific purpose. However, beginners such as our students who lack experience in programming will find it difficult to learn. Teaching UML not only considers the design, but also how it determines the actual implementation of a system.

The first difficulty that beginners find in learning UML is the notations that they are unfamiliar with. This should not be intrinsically difficult. Repeated practices could help students to learn, but a tool is missed for them to interact with. This fundamental knowledge should be acquired after practice. The new tool will provide a highly interacted platform so that students could repeatedly practice the newly learnt UML notations on it and receive responses from it.

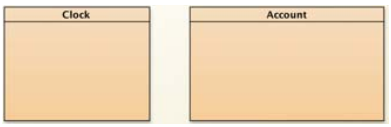
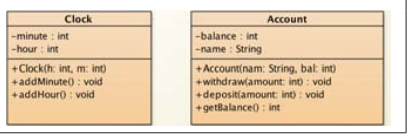
The second difficulty could be the connection of UML to source code. UML class diagrams do relate and reflect source code. However, many tools omit the connection due to complexity. Despite that BlueJ provides a weak connection (e.g., inherence relationship), it is not enough for the purpose of teaching system design courses. The introduction of them could take up most of the lecture time while the students get only a brief understanding. This obstructs students who are not well trained with Abstract Thinking from mastering the fundamental concepts. Our new tool provides not only a reflection from source code to class diagram, but also a reversed way. This tightens the connection that visually shows in class diagram and physically shows in source code. Students are encouraged to observe update in class diagram that reflects source code and in the opposite, observe the update of source code that is triggered by modifying class diagram. This tightened connection was extended from BlueJ to include types of associations (from the weakest association to strongest composition).

To this end, we exploited the advantages of BlueJ and Visual Paradigms [8] and introduced BlueJ-UML. It is a modification of BlueJ with an extension of detailed class diagram support. A pedagogical approach is designed for novices to master the foundations of system design. By combining the use of BlueJ-UML and the novice specific pedagogical approach, the students who did not have a basic understanding of object-oriented programming should be able to quickly acquire skills needed for the system design related courses. BlueJ-UML strengthens the weak connection in BlueJ, which helps students to quickly acquire programming skills but omits the deeper design aspects.

IV. Features of BlueJ-UML

In a class, attributes and features are the key components. UML defined specific notations for them while program languages defined different syntaxes. BlueJ-UML showed them in a class diagram that BlueJ lacks and could obstruct students to digest this abstract information. In fact, source code represents the actual structure of a class and a class diagram represents it at a higher level. Students could feel difficult to realise the relationship between a class diagram and source code because of the weak details. Table I shows a brief comparison between BlueJ and BlueJ-UML.

TABLE I
A comparison between BlueJ-UML and BlueJ in respect of Class detailed information

BlueJ	
BlueJ-UML	

UML notations [9] could be strange to students. An attribute consists of publicity, name, and type. A feature consists of publicity, name, parameters name, parameters type, and return type. These notations could express a detailed design, however, could be a bit complex to novices. A try-and-error approach is suggested for students to learning these notations. BlueJ-UML provides an instant update to the source code when editing

[2]https://www.eclipse.org/

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on October 06,2026 at 08:29:55 UTC from IEEE Xplore. Restrictions apply.

those content via UML notation. With the teaching materials, students can follow the notations to update and reflect source code as shown in Figure 1. While they are observing those changes, they can digest the usage of UML notations. After several round-trip updating, from the observation, students should easily understand how each notation reflects the source code.

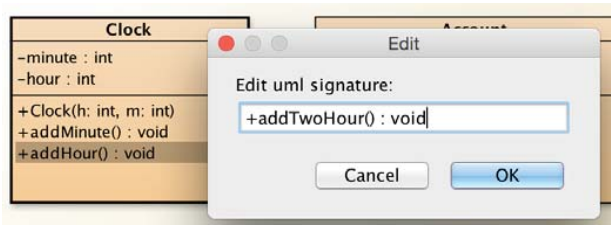


Fig. 1. BlueJ-UML capability of updating UML details

Important information in a class diagram can be the correlation of classes. Students could find obstacles to understand how a class interacts with another. BlueJ represents this information by a weak use-dependency. It might be too abstract for novices to master and in fact, it does not realise the general needs in system design. On the other hand, use-dependency could be realised in several implementations that novices may have never seen. This leads to a need to introduce association, aggregation, and composition dependency in BlueJ. BlueJ-UML, therefore, reflects these dependencies from source code and allows students to interact with the class diagram for an update on source code. This could consolidate their understanding on how these different strength dependencies are visually represented and actually reflected.

Novices could feel confused when they are learning the dependency strength. Association, aggregation, and composition dependency are reflected by BlueJ-UML in source code directly. Students could observe the changes in the source code and relate them to the class diagram. It provides a practical level experience to the beginners.

BlueJ-UML also highlights the source code that relates to a dependency that makes the relationship between source and diagram clearer. In Figure 2, a composition aggregation owned by Car_Composition and directed to Engine is selected with source code that related to the composition highlighted in the Car_Composition class. It is suggested that the students read the codes and realise them in its class diagram, vice versa.

BlueJ-UML not only remains the features from BlueJ, but added many new features to facilitate teaching and learning object-oriented programming and designing courses. Besides the features talked above, Table II compares and contrasts the feature differences of BlueJ and BlueJ-UML.

V. Evaluation of Usability and Object-Oriented Paradigm Support

This study concerns the usability of the developed BlueJ-UML towards learning object-oriented programming, which

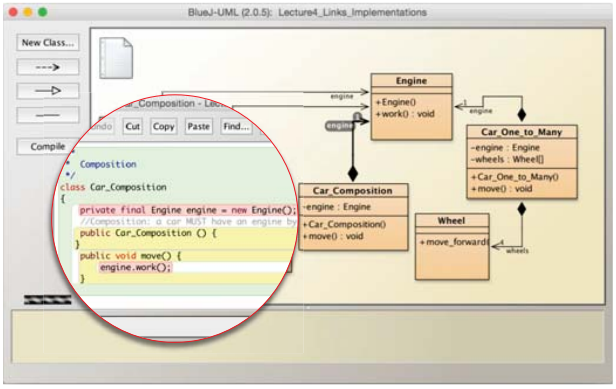


Fig. 2. Source code highlight for selected association

TABLE II
BlueJ and BlueJ-UML Matrix

	BlueJ	BlueJ-UML
Object-first support	√	√
Basic Integrated Development Environment	√	√
Basic Class Diagram	√	√
Showing attributes		√
Showing operations		√
Showing associations		√
Showing aggregations		√
Showing compositions		√
Showing directions for dependencies		√
Showing multiplicities for dependencies		√
Showing roles for dependencies		√
Updating source by editing attributes		√
Updating source by editing operations		√
Creating associations		√
Removing associations		√
Updating source by changes of strength of dependency		√
Updating source by changes of multiplicity		√
Updating source by changes of role		√
Showing comment node		√
Basic Use-case Diagram		√

deals with human-computer interface and performance related issues. This paper accesses this area by Enhanced Technology Acceptance Model (well known as TAM2) [10], which is an existing empirical evaluation technique. TAM2 is an enhancement of TAM [5] that is a commonly used framework for evaluating user perceptions towards a technology. Its theory suggests that when users are presented with a new technology, a number of factors will influence their decision about how and when they will use it. TAM has a strong focus on *Perceived Usefulness* of the user and *Perceived Ease of Use* of the user. TAM2 as an enhancement provides more detailed explanations on why the participants found the system useful. TAM2 performs well in both mandatory and voluntary settings. As shown in Figure 3, it suggests that *Image* of the system, *Job Relevance* of the system, and *Output Quality* of the system correlate system usefulness; while *Subject Norm* has no effect in *Mandatory* setting, it does has effect in *Voluntary* setting.

The survey had been conducted in week 11. There were

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on October 06,2026 at 08:29:55 UTC from IEEE Xplore. Restrictions apply.

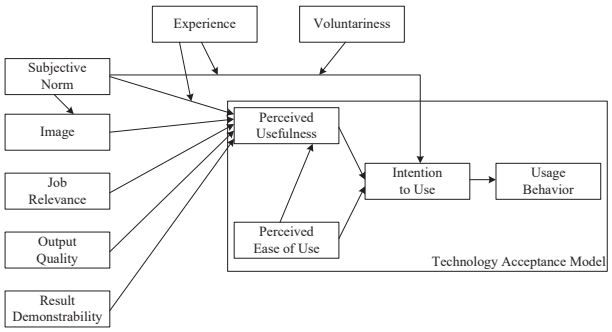


Fig. 3. TAM2 Model

total 95 responses out of 152 students who were enrolled in CS3342 (Software Design). A questionnaire was distributed to and returned from them. The questionnaire was designed upon TAM2 model and there were nine facets including Intention to Use, Perceived Usefulness, Perceived Ease-of-Use, Subjective Norm, Voluntariness, Image, Job Relevance, Output Quality, Result Demonstration. In addition, a direct measure was included which determines whether BlueJ-UML is valuable or not.

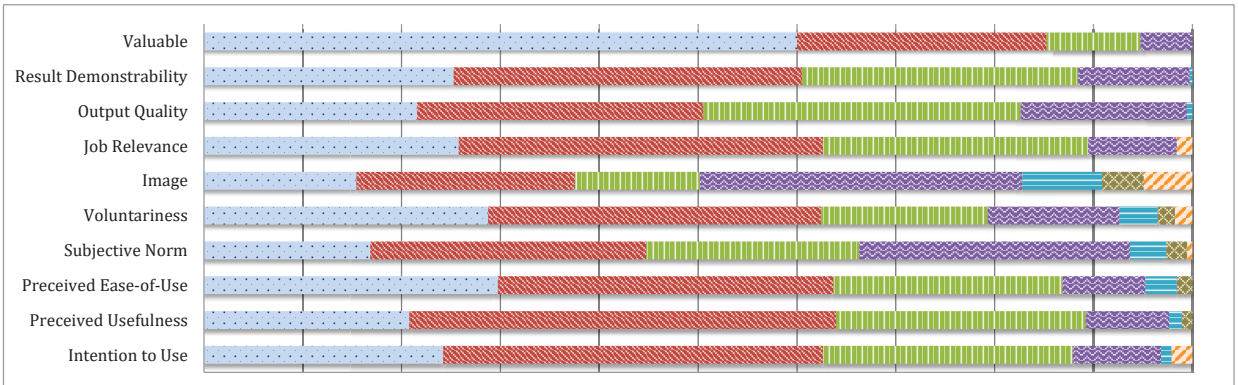
The quantitative scale was ranged from 1 to 7 points, respectively, strongly disagree, moderately disagree, somewhat disagree, neutral, somewhat agree, moderately agree, and strongly agree. Participants were required to select a scale for each question. In data clearing, for each single empty response, all responses were averaged from the same participant and round the calculated point by 0.5. In Table III, the summary of measurement scales, all skewness are negative that the majority of participants tend to the positive side. The majorities of standard deviation are around 1. The summary of the results is positive. Many participants considered BlueJ-UML is valuable. An evaluation on the usage of BlueJ-UML is conducted by a survey in class using TAM2 questionnaire. It accesses how useful BlueJ-UML is. The result as shown in Figure 4 was positive. Excepting the facet of Image, every facet gained about or more than 80% positive responses. About 95% of participants agree that BlueJ-UML is valuable in the course; about 88% participants intent to use BlueJ-UML; almost 90% participants consider BlueJ-UML is useful; almost 90% participants thought that BlueJ-UML is relevant to the course; and around 82% of participants treated that the output quality of BlueJ-UML is high.

Finally, we compute Cronbach's alpha, a measure of internal consistency (i.e., how closely related a set of items are as a group). Note that the value of Cronbach's alpha cannot be calculated for groups with the number of items less than 3. The Cronbach's alpha of Perceived Usefulness, Perceived Ease-of-Use, Voluntariness, Image and Result Demonstration are 0.97, 0.94, 0.84, 0.95 and 0.97 respectively, which shows very good (>0.80) and even excellent (>0.90) internal consistency.

TABLE III
SUMMARY OF MEASUREMENT SCALES (SD STANDS FOR STANDARD DEVIATION)

Constructs items	Mean	SD	Skew
Intention to Use			
A1. Assuming I have access to BlueJ-UML, I intend to use it.	5.71	1.17	-1.38
A2. Given that I have access to BlueJ-UML, I predict that I would use it.	5.79	1.15	-1.54
Preceived Usefulness			
B1. Using BlueJ-UML improves my performance in my learning.	5.70	0.99	-0.72
B2. Using BlueJ-UML in class increases my productivity.	5.69	0.96	-0.86
B3. Using BlueJ-UML enhances my effectiveness in my learning.	5.78	0.97	-0.91
B4. I find BlueJ-UML to be useful in my learning.	5.87	1.00	-1.11
Preceived Ease-of-Use			
C1. My interaction with BlueJ-UML is clear and understandable.	5.97	1.06	-1.11
C2. Interacting with BlueJ-UML does not require a lot of my mental effort.	5.76	1.16	-0.93
C3. I find BlueJ-UML to be easy to use.	5.81	1.16	-1.07
C4. I find it easy to get BlueJ-UML to do what I want it to do.	5.70	1.13	-0.93
Subjective Norm			
D1. People who influence my behavior think that I should use BlueJ-UML.	5.27	1.25	-0.23
D2. People who are important to me think that I should use BlueJ-UML.	5.21	1.32	-0.43
Voluntariness			
E1. My use of BlueJ-UML is voluntary.	5.59	1.30	-1.18
E2. My teacher does not require me to use BlueJ-UML.	5.62	1.43	-1.25
E3. Although it might be helpful, using BlueJ-UML is certainly not compulsory in class.	5.79	1.27	-1.23
Image			
F1. Classmates who use BlueJ-UML have more prestige than those who do not.	4.79	1.62	-0.44
F2. Classmates who use BlueJ-UML have outstanding performance.	4.88	1.53	-0.31
F3. Having BlueJ-UML is a status symbol in class.	4.65	1.69	-0.54
Job Relevance			
G1. In class, usage of BlueJ-UML is important.	5.60	1.20	-1.57
G2. In class, usage of BlueJ-UML is relevant.	5.93	0.93	-0.58
Output Quality			
H1. The quality of the output I get from BlueJ-UML is high.	5.50	1.05	-0.16
H2. I have no problem with the quality of BlueJ-UML's output.	5.62	1.00	-0.10
Result Demonstrability			
I1. I have no difficulty telling others about the results of using BlueJ-UML.	5.76	0.96	-0.40
I2. I believe I could communicate to others the consequences of using BlueJ-UML.	5.75	0.95	-0.27
I3. The results of using BlueJ-UML are apparent to me.	5.78	0.91	-0.22
I4. I would not have difficulty explaining why using BlueJ-UML is beneficial or not.	5.80	0.99	-0.38
Valuability			
O. Overall, I consider BlueJ-UML valuable.	6.36	0.85	-1.25

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on October 06,2026 at 08:29:55 UTC from IEEE Xplore. Restrictions apply.



	Intention to Use	Preceived Usefulness	Preceived Ease-of-Use	Subjective Norm	Voluntariness	Image	Job Relevance	Output Quality	Result Demonstrability	Valuable
Storngly Agree	24.21%	20.79%	29.74%	16.84%	28.77%	15.44%	25.79%	21.58%	25.26%	60.00%
Moderately Agree	38.42%	43.16%	33.95%	27.89%	33.68%	22.11%	36.84%	28.95%	35.26%	25.26%
Somewhat Agree	25.26%	25.26%	23.16%	21.58%	16.84%	12.63%	26.84%	32.11%	27.89%	9.47%
Neutral	8.95%	8.42%	8.42%	27.37%	13.33%	32.63%	8.95%	16.84%	11.32%	5.26%
Somewhat Disagree	1.05%	1.32%	3.16%	3.68%	3.86%	8.07%	0.00%	0.53%	0.26%	0.00%
Moderately Disagree	0.00%	1.05%	1.58%	2.11%	1.75%	4.21%	0.00%	0.00%	0.00%	0.00%
Strongly Disagree	2.11%	0.00%	0.00%	0.53%	1.75%	4.91%	1.58%	0.00%	0.00%	0.00%

Fig. 4. Results of usability survey of BlueJ-UML

VI. Conclusion

This project successfully develops a new interactive learning method for object-oriented programming at introductory level, that extends and enhances BlueJ platform incorporating important UML elements. This was to strengthen existing teaching method in object-oriented programming, and to provide a proven pedagogical approach for students to master the basic principles of software design. According to UML specification, BlueJ-UML implements attribute, method, and constructor information in class diagram and presenting important use case in the design. The original use-dependency in BlueJ was extended to stronger object association, aggregation, and composition dependency. Object multiplicity, role, and direction could also be included in class diagram as presets useful for learning. BlueJ-UML, in addition, strengthens the interactions between student and BlueJ by providing manipulation on class diagram through UML notations to source code. Our results show that 1) the new pedagogical approach introduced in teaching object-oriented programming using BlueJ-UML in Software Engineering courses is well appreciated, and largely accepted by all students, 2) extending and implementing UML diagramming capabilities facilitate the important learning needs of essential object-oriented concepts within the learning platform. Actual evaluation and impact on student learning have been conducted using our comprehensive TAM survey in the course, where BlueJ-UML have been utilized. Students class performance has been significantly improved in which positively correlated with the adoption of BlueJ-UML.

Acknowledgment

This work is supported in part by the research funds of City University of Hong Kong (No. 9042499, 6000485, 9678149).

References

[1] J. Sorva, V. Karavirta, and L. Malmi, "A review of generic program visualization systems for introductory programming education," *ACM Transactions on Computing Education (TOCE)*, vol. 13, no. 4, p. 15, 2013.

[2] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg, "The bluej system and its pedagogy," *Computer Science Education*, vol. 13, no. 4, pp. 249–268, 2003.

[3] G. Booch, *The unified modeling language user guide*. Pearson Education India, 2005.

[4] W.-J. Park and D.-H. Bae, "A two-stage framework for uml specification matching," *Information and Software Technology*, vol. 53, no. 3, pp. 230–244, 2011.

[5] R. H. Shroff, C. C. Deneen, and E. M. Ng, "Analysis of the technology acceptance model in examining students' behavioural intention to use an e-portfolio system," *Australasian Journal of Educational Technology*, vol. 27, no. 4, 2011.

[6] C. Larman, *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Interative Development*. Pearson Education India, 2012.

[7] J. Whittle, J. Hutchinson, and M. Rouncefield, "The state of practice in model-driven engineering," *IEEE software*, vol. 31, no. 3, pp. 79–85, 2014.

[8] V. Paradigm, "Visual paradigm for uml," *Visual Paradigm for UML-UML tool for software application development*, p. 72, 2013.

[9] S. Maoz, J. O. Ringert, and B. Rumpe, "Modal object diagrams," in *European Conference on Object-Oriented Programming*. Springer, 2011, pp. 281–305.

[10] R. Cheung and D. Vogel, "Predicting user acceptance of collaborative technologies: An extension of the technology acceptance model for e-learning," *Computers & Education*, vol. 63, pp. 160–175, 2013.

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on October 06,2026 at 08:29:55 UTC from IEEE Xplore. Restrictions apply.