Acceptance of Verificator by Information Science Students

Tihomir Orehovacki, Danijel Radosevic, and Mladen Konecki
University of Zagreb, Faculty of Organization and Informatics
Pavlinska 2, 42000 Varazdin, Croatia
{tihomir.orehovacki, danijel.radosevic, mladen.konecki}@foi.hr

Abstract. *Verificator is an educational tool for teaching introductory programming at the university level. This study integrates programming anxiety and computer self-efficacy into the Technology Acceptance Model (TAM) with an aim to examine the factors that affect the acceptance of Verificator by information science students. Data were collected using an online survey questionnaire while the hypothesized relationships in the research model were tested employing the variance-based Partial Least Squares (PLS) technique. Theoretical and practical implications as well as future research directions are presented and discussed.*

Keywords. Verificator, Educational Tool, Teaching Programming, Technology Acceptance Model (TAM), Partial Least Squares (PLS)

1. Introduction

Verificator is an educational tool aimed for teaching and learning programming [27]. Four years ago it was introduced in introductory programming courses at the University of Zagreb, Faculty of Organization and Informatics in Varaždin. The idea underpinning the Verificator tool was to provide a customized learning programming interface to students as an alternative to standard programming tools. It had developed from the authors' several years' practical teaching experience. The issues that arose in the programming classes included the students' tendency to avoid some good programming habits in favor of dubious techniques like rote learning of programming and, consequently, to avoid true understanding of programming concepts, syntax and algorithms. Other frequently experienced problems were students copying assignments from their peers, lack of previous programming skills, difficulties in adoption of program syntax and even programming anxiety.

As an interface aimed for learning programming, Verificator consists of a programming editor, debugger and tools for testing programs and analysis of the C++ program code. It uses a standard C++ compiler, like gcc. The purpose of Verificator is to help students to acquire good programming habits and prevent them from adopting some bad ones. An example of a student's bad programming habit writing a program that contains a large amount of programming code without making any logic or even syntax check. Unlike standard programming tools, Verificator prevents such practices by stimulating students to compile and test their programs in prescribed intervals that are specified by the amount of written code. On the other hand, students often have problems with programming language syntax. Verificator amends a usual set of compiler/linker messages, such as error messages and warnings, by more intuitive tutor messages that are designed to explain the cause of errors.

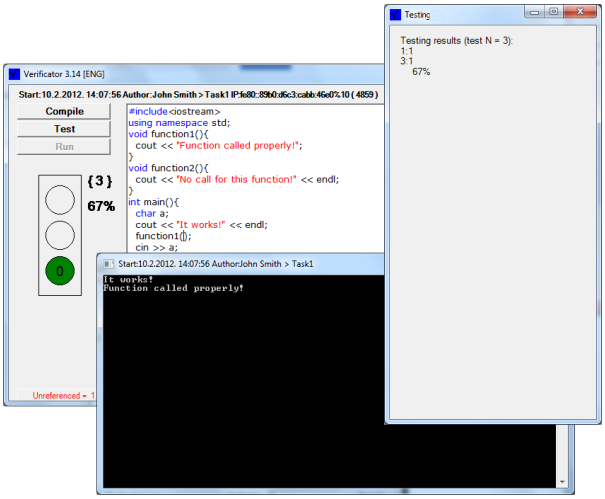

Figure 1. Verificator interface

The main functions of Verificator are personalization of programs, control points in program development and syntax help. Personalization of a program starts with a student's entry form, the data from which is saved into a C++ source file in form of comments. There is no possibility of copying

program code into/from Verificator, so the included checksum (first line in the example) virtually guarantees that a student's program is not written outside the tool. Internal copy/paste is enabled to improve students' efficiency.

The first kind of control points uses the semaphore as part of the programming interface (Figure 1) to indicate that program syntax should be checked by the compiler. The green light means that a student can continue writing the code, while the yellow light indicates that the code should be checked for errors by compilation. If the student continues writing the code after the red light appears, the compilation will not be possible until some code is reduced or changed into comments until the yellow light occurs.

In addition to the syntax check, Verificator introduces a special testing mode to check the program logically. During the testing mode, Verificator notices all reached program blocks (corresponding pairs of curly brackets) and expresses it as a percentage of the total number of programming blocks. This prevents students from writing the code 'in advance', e.g. writing functions without their calls, thus retracting their testing. The Tutor function helps students with C++ syntax in a more intuitive way than compiler/linker errors and warnings do [28]. It is important that all tutor messages are optional, i.e. the student needs to decide whether to accept it or not. Tutor appears in a window with compiler/linker output, in form of a bulb. The student needs to click on the bulb to see Tutor's comment about the code. Students are themselves highly involved in the development of Verificator, especially in its testing and identification of program bugs, which is honored by extra points, in accordance with the Bologna teaching process.

The aim of this paper is to investigate to what extent programming anxiety, computer self-efficacy, perceived ease of use and perceived usefulness contribute to the acceptance of Verificator by undergraduate information science students. The remainder of the paper is structured as follows. Section 2 provides a brief background to the research. The research model and hypotheses are presented in the third section. Section 4 describes the research methodology while section 5 presents research results. A discussion of findings and indications of future research are provided in the last section.

2. Background to the research

Technology Acceptance Model (TAM) is based on the Theory of Reasoned Action [12] and the Theory of Planned Behavior [1]. Among numerous models proposed to explain and predict the use of a system, TAM has been the one to capture most attention by the Information System community [8]. Moreover, it has been the most widely employed model of information technologies adaptation and use [35]. TAM, proposed in 1985 by Fred Davis in his doctoral thesis, initially comprised two constructs: perceived usefulness (PU) and perceived ease of use (PEOU). They are included among the set of the perceived characteristics of innovations [30] through which we can capture all relevant beliefs in information technology usage contexts [4]. In the final version of TAM, it was established that both perceived usefulness and perceived ease have a direct influence on behavioral intention [8]. Although research has shown that e-learning and online learning are among top ten keywords in TAM research papers from 1991 to 2009 [6], the research in these areas is not as extensive as it would be expected. Most of it has been done in the fields of "Business" and "Management" [6]. In those studies, most of the participants were real-world knowledge workers, although numerous studies were conducted on students as well [21]. We believe that TAM can be applied in the field of learning programming and that such an approach can lead to data that can be useful for improving the progress of creating effective tutors for novices.

Over the years, TAM was refined. TAM2, introduced in 2000, provided more detailed explanations for the individual perception of usefulness of a given system, as it proposed additional variables as antecedents to the perceived usefulness. New variables were afterwards introduced to the perceived ease of use as well [8]. Two main groups of antecedents for the perceived ease of use were identified: anchors and adjustments. Anchors referred to general beliefs about computers and computer usage, whereas adjustments referred to beliefs that are shaped based on one's direct experience with the target system [8]. TAM3 proposed new relationships and suggested that experience will moderate the relationship between the following variables: perceived ease of use and perceived usefulness; computer anxiety and perceived ease of use; and perceived ease of use and behavioral intention [35].

The criticism concerning this model is based on the claim that a few studies on TAM make use of students as participants in a controlled environment, and therefore, results obtained from these studies cannot be generalized to the real world [21]. In our context, however, this is not relevant since we try to draw conclusions for students in schools and universities, which also constitute a controlled environment.

TAM has proven to be a useful theoretical model in helping to understand and explain use behavior in IS implementation. It has been tested in many empirical researches and the tools used with the model have proven to be of quality and to yield statistically reliable results [22]. TAM is generally regarded as the most momentous theory due to the increasing volume of relevant researches in recent years [6].

3. Research model and hypotheses

The main purpose of this study is to empirically validate the research model in the context of Verificator. The proposed research model presented in Figure 2 extends TAM [11] with computer self-efficacy and programming anxiety constructs. The hypotheses that follow were developed based on the previous research on this topic.

Programming Anxiety (PANX), as a constituent part of computer anxiety [36], refers to a situation-specific psychological state caused by negative experiences or expectations to lose self-esteem when confronting a computer programming situation [10]. Anxiety has been linked to attitude in several studies [19][26]. However, research related to the direct effect of programming anxiety on attitude towards use has been deficient. Therefore, we hypothesize:

H1. Programming Anxiety has a positive direct effect on Attitude Towards Using Verificator.

Computer Self-Efficacy (CSE) is defined as an individual's self-confidence in his or her ability to use computers [9]. In the context of our research, self-efficacy describes the students' perceptions of their abilities to use Verificator. Given that researchers have found a positive relationship between Computer Self-Efficacy and Perceived Ease of Use (e.g. [3][11][33]), we hypothesize as follows:

H2. Computer Self-Efficacy has a positive direct effect on the Perceived Ease of Use of Verificator.

Perceived Ease of Use (PEOU) refers to the extent to which students believe that using Verificator would be free of effort. As previous studies have demonstrated the significant effect of Perceived Ease of Use on Perceived Usefulness, and on Attitude Towards Using (e.g. [20][24][32]), we propose the following hypotheses:

H3. Perceived Ease of Use has a positive direct effect on Perceived Usefulness.

H4. Perceived Ease of Use has a positive direct effect on the Attitude Towards Using Verificator.

Perceived Usefulness (PU) is defined as an extent to which an individual believes that using a particular system would enhance his or her performance [11]. In the context of our research, PU determines the degree to which students believe that using Verificator would increase their performance in understanding of programming concepts. Many researchers have reported a strong influence of Perceived Usefulness on Attitude Towards Using, and Intention to Use (e.g. [11][24]). Accordingly, we hypothesize as follows:

H5. Perceived Usefulness has a positive direct effect on the Attitude Towards Using Verificator.

H6. Perceived Usefulness has a positive direct effect on the Intention to Use Verificator.

Attitude Towards Using (ATU) is defined as the extent to which the individual favors the use of a particular system. In regard to Verificator, ATU is viewed as students' desire to use Verificator to support their learning programming. The positive impact of Attitude Towards Using on behavioral intentions has been widely discussed in literature (e.g. [2][31]). Thus, we hypothesize:

H7. Attitude Towards Using Verificator has a positive direct effect on the Intention to Use Verificator.

4. Methodology

A quantitative study in form of an online survey questionnaire was performed to test the hypotheses stated in the third section. With an aim to ensure the content validity of the scales, we adapted items based on previous studies. Perceived Usefulness (PU) was measured with four items adapted from Davis [11]. We adapted six items from the same study for evaluating Perceived Ease of Use (PEOU) and three items for measuring Intention to Use (ITU). For measuring Computer Self-Efficacy (CSE) we adapted three items suggested by Compeau and Higgins [9]. Three items aimed for measuring Attitude Towards Using (ATU) were adapted from Gong et al. [16]. Finally, to measure Programming Anxiety (PANX), we adapted four items from Venkatesh [34].

To ensure they are relevant to our study, we made modifications of the items with regards to the context of programming and Verificator. All the items were measured on a five point Likert-type scale ranging from 1 – strongly agree to 5 – strongly disagree. The proposed research model consists of six constructs, while the measuring instrument contains 24 items.

Table 1. Demographic profile of the sample

Variable	n	%
Gender		
Male	61	68.5
Female	28	31.5
Age		
18	1	1.1
19	6	6.7
20	30	33.7
21	36	40.4
22	10	11.2
23	2	2.2
24	2	2.2
25	1	1.1
31	1	1.1
Frequency of Verificator use (hours per week)		
1-3 hours per week	40	44.9
4-10 hours per week	48	53.9
11-20 hours per week	1	1.1
Frequency of Verificator use (times per week)		
1-2 times a week	59	66.3
3-6 times a week	29	32.6
1-2 times a day	1	1.1

The participants in the study were 89 second-year students enrolled in the Programming 2 course. The sample consisted of 61 male (68.5%) and 28 female (31.5%) students. The average age of students was 20.88 (SD = 1.558). The use of Verificator was obligatory during lab-based sessions which were held once a week for 90 minutes. However, students were also active users of Verificator outside the class hours. According to the results presented in Table 1, 53.9% of students used Verificator for between four and ten hours a week, while 33.7% of them reported to have had interaction with Verificator more than twice a week.

5. Results

The findings of this study provide support for the research model regarding the acceptance of the Verificator education tool by information science students. The research model was tested using Partial Least Squares (PLS) [23][36], a variance-based structural equation modeling (SEM) technique. There are two main reasons why PLS was chosen over covariance-based techniques: (i) PLS can be applied to small sample sizes [13], and (ii) PLS supports both exploratory and confirmatory research and is consequently appropriate for testing theories in the early stages of their development [15]. According to Chin [7] and Gefen et al. [15], the minimum sample size for a PLS analysis should be (i) 10 times larger than the number of items for the most complex construct, or (ii) 10 times the largest number of independent constructs impacting a dependent construct. In the proposed research model, the most complex construct (perceived ease of use) is measured with six items while the largest number of independent constructs estimated for a dependent construct is three (for attitude towards using). Accordingly, the minimum required sample size was 60, which indicates that our sample size of 89 was adequate for PLS estimation procedures. The structural and measurement models were estimated using SmartPLS 2.0 M3 software [29].

Table 2. Construct reliability and convergent validity

	AVE	Composite Reliability	Cronbach's α
ATU	0.8040	0.9425	0.9186
CSE	0.6892	0.8691	0.7853
ITU	0.7284	0.8891	0.8117
PANX	0.7634	0.9278	0.8965
PEOU	0.6439	0.9155	0.8890
PU	0.7302	0.9147	0.8734

The reliability and validity of the measurement model was assessed by means of indicator reliability, construct reliability,

convergent validity, and discriminant validity. Construct reliability was examined by calculating the composite reliability and Cronbach's alpha. A threshold for sufficient values of both coefficients is 0.7 [18][25]. As reported in Table 2, Cronbach's alpha and composite reliability values exceeded the required threshold for all items, thus implying high internal consistency of the scales. The average variance extracted (AVE) is a common measure to examine convergent validity. AVE values higher than 0.5 are considered to be acceptable [14]. As shown in Table 2, all average variances extracted were in excess of 0.5, thereby demonstrating adequate convergent validity.

Table 3. Discriminant validity using square root of AVE

	ATU	CSE	ITU	PANX	PEOU	PU
ATU	**0.8967**					
CSE	0.3590	**0.8302**				
ITU	0.7820	0.3592	**0.8535**			
PANX	0.4275	0.1834	0.3674	**0.8737**		
PEOU	0.6751	0.5393	0.6002	0.2513	**0.8024**	
PU	0.6816	0.2664	0.7389	0.2531	0.5585	**0.8545**

Indicator reliability was estimated by using confirmatory factor analysis (CFA). As a rule of thumb, loading values larger than 0.7 are frequently judged as acceptable [19]. According to the results presented in Table 4, all loadings for the items in the measurement model were greater than 0.7, thereby demonstrating sufficient indicator reliability. Discriminant validity was tested using the criterion which requires that the square root of every AVE should exceed the correlations among any pair of constructs [14]. Moreover, items should load more strongly on their own corresponding construct than on other constructs in the model. As shown in Tables 3 and 4, all constructs in the research model satisfied these two criteria for discriminant validity. The above results indicate sound reliability and validity of the measurement model, which is the prerequisite for structural model assessment and hypotheses testing.

The results of hypotheses testing using a PLS path analysis for the structural model are summarized in Figure 2 and Table 5. In contrast to covariance-based approaches, the PLS technique does not generate an overall goodness-of-fit index. Therefore, model validity is primarily assessed by examining R^2 values [7] and effect size [5]. R^2 reflects the amount of variance explained by the model, thus indicating its predictive power. The model explains 63.76% of the variance in Attitude Towards Behaviour, 31.19% in Perceived Usefulness, and 29.08% in Perceived Ease of Use. Additionally, 69.07% of the variance in the Intention to Use was accounted for by the constructs in the model.

Table 4. Indicator reliability (loadings) and discriminant validity (cross-loadings)

	ATU	CSE	ITU	PANX	PEOU	PU
ATU1	**0.9022**	0.3517	0.7844	0.3585	0.5504	0.6364
ATU2	**0.9014**	0.2923	0.7016	0.4233	0.5989	0.5701
ATU3	**0.9162**	0.3081	0.6687	0.4365	0.6483	0.6074
ATU4	**0.8662**	0.3349	0.6449	0.3139	0.6272	0.6304
CSE1	0.5040	**0.8731**	0.4392	0.2213	0.5763	0.3618
CSE2	0.0846	**0.8000**	0.1534	0.1228	0.3390	0.0662
CSE3	0.1778	**0.8156**	0.2175	0.0738	0.3575	0.1482
ITU1	0.7380	0.3298	**0.9245**	0.2895	0.5771	0.7201
ITU2	0.6321	0.3783	**0.8195**	0.2953	0.5614	0.5720
ITU3	0.6255	0.2108	**0.8118**	0.3639	0.3915	0.5888
PANX1	0.2658	0.0201	0.1923	**0.7766**	0.1421	0.0581
PANX2	0.3920	0.1669	0.3191	**0.9019**	0.2195	0.2032
PANX3	0.3926	0.2063	0.3756	**0.8806**	0.2361	0.3201
PANX4	0.4177	0.2046	0.3627	**0.9282**	0.2598	0.2553
PEOU1	0.5074	0.4498	0.4517	0.1845	**0.7752**	0.3123
PEOU2	0.5212	0.3927	0.4708	0.3019	**0.7779**	0.4046
PEOU3	0.5298	0.3480	0.4765	0.1881	**0.7974**	0.5220
PEOU4	0.5482	0.3777	0.4867	0.2166	**0.7748**	0.4654
PEOU5	0.5833	0.4912	0.4700	0.1326	**0.8092**	0.4292
PEOU6	0.5576	0.5243	0.5308	0.1985	**0.8754**	0.5350
PU1	0.5696	0.2964	0.6321	0.2319	0.5055	**0.8690**
PU2	0.6595	0.2752	0.7168	0.2818	0.5403	**0.9265**
PU3	0.5805	0.2112	0.6510	0.2026	0.4455	**0.8944**
PU4	0.5092	0.1056	0.5064	0.1303	0.4067	**0.7124**

Effect size examines whether an independent latent construct has a substantial influence on the dependent latent construct. It is determined by comparing the R^2 of the dependent construct with and without the presence of each independent construct [7]. Programming Anxiety, Perceived

Usefulness, and Perceived Ease of Use were shown to have a small effect size ($f^2 = 0.13$, $f^2 = 0.12$, $f^2 = 0.12$, respectively) on Attitude Towards Use.

The levels of significance were estimated using two-tailed t-statistics derived from a bootstrapping procedure with 1000 resamples, as suggested by Chin [7]. All path coefficients of the hypothesized causal links in the research model are significant. Thus, hypotheses H1 through H7 are supported.

Table 5. Results of hypotheses testing

Hypothesis	β	t value	p value	Supported
H1: PANX → ATU	0.2264	2.8390	p<0.01	Yes
H2: CSE → PEOU	0.5393	6.5790	p<0.0001	Yes
H3: PEOU → PU	0.5585	5.9302	p<0.0001	Yes
H4: PEOU → ATU	0.3917	3.4468	p<0.001	Yes
H5: PU → ATU	0.4055	4.0662	p<0.0001	Yes
H6: PU → ITU	0.3845	3.7585	p<0.001	Yes
H7: ATU → ITU	0.5199	4.4826	p<0.0001	Yes

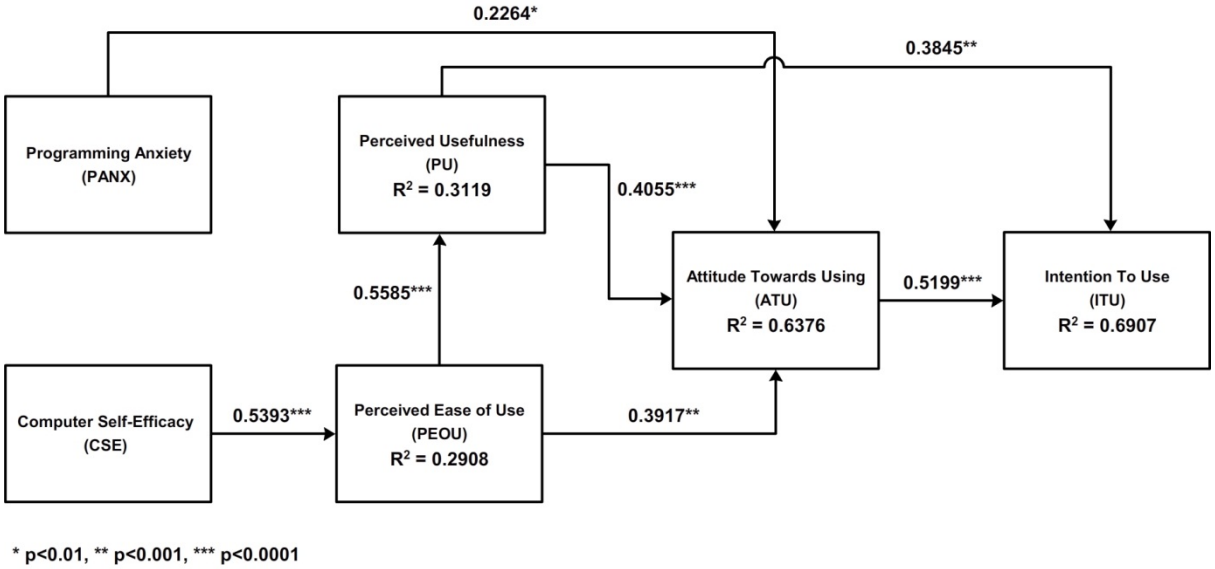

Figure 2. PLS structural model

6. Discussion and Conclusion

This study investigated the factors that impact the acceptance of the Verificator educational tool by information science students. According to the results of the PLS path analysis, programming anxiety, perceived usefulness, and perceived ease of use directly affect students' attitude towards using Verificator, whereas perceived usefulness and attitude towards using are strong and significant determinants of students' intention to use Verificator. This means that students who are both anxious about programming and who believe that the use of Verificator is free of effort will have a positive attitude towards its use. Besides, students who assume that using Verificator would enhance their performance in acquiring programming skills intend to keep using it and recommending it to their peers. Other important results indicate that (i) computer self-efficacy has a positive significant effect on perceived ease of use, and (ii) perceived usefulness is significantly attributed to perceived ease of use. All the above-presented findings are in line with results reported by other studies in the literature. However, we must emphasize that this is the first study to show that programming anxiety has a direct and significant impact on attitude towards using an educational tool aimed for learning programming.

This study provides useful contributions and implications to both researchers and practitioners. From a theoretical point of view, TAM was extended with two constructs (programming anxiety and computer self-efficacy), which turned out to have an important role in the explanation of the model. All the hypothesized relationships were satisfactorily supported, thus implying that the validated model can be applied to: (i) predicting students' acceptance of educational

tools aimed for learning programming, (ii) diagnosing possible reasons for lack of their acceptance. In order to increase the acceptance of educational tools, practitioners must not only focus on fulfilling students' technological expectancies related to usefulness and ease of use but also consider developing an environment that would decrease programming anxiety.

The findings of this research must be considered in light of its limitations. First, generalizability is an issue that plagues most empirical studies. Given that participants in our study were undergraduate information science students, other demographic samples should be explored in order to make sound conclusions. Similarly, only one type of educational tool was evaluated in this study, whereas an assessment of diverse educational tools may reveal different results. Secondly, only one research method (online questionnaire survey) was employed, potentially leading to a bias due to common method variance. Therefore, in our future work we will employ several methods in order to determine to what extent findings presented in this paper can be applied to other participants, contexts, and educational tools. Besides, we intend to extend the proposed model with other constructs that may have influence on the acceptance of educational tools aimed for learning programming. Finally, we also plan to examine moderating and mediating effects of the introduced constructs and estimate the model's predictive validity.

7. References

[1] Ajzen I. The Theory of Planned Behavior. Organizational Behavior and Human Decision Processes 1991, 50(2): 179-211

[2] Ajzen I, Fishbein M. Understanding attitudes and predicting behavior. Englewood Cliffs, NJ: Prentice Hall, 1980.

[3] Agarwal R, Sambamurthy V, Stair RM. Research report: The evolving relationship between general and specific computer self-efficacy - An empirical assessment. Information Systems Research 2000, 11(4): 418-430.

[4] Benbasat I, Barki H. Quo vadis, TAM? Journal of the Association for Information Systems 2007, 8(4): 211-218

[5] Cohen J. Statistical power analysis for the behavioral sciences, second edition. Hillsdale: Lawrence Erlbaum Associates, 1988.

[6] Chang S-H, Chou C-H, Yang J-M. The Literature Review of Technology Acceptance Model: A study of the Bibliometric Distributions, In Proceedings of the Pacific Asia Conference on Information Systems; 2010 July 9-12; Taipei, Taiwan. AISeL; 2010. pp. 1634-1640.

[7] Chin WW. The partial least squares approach to structural equation modeling. In: Marcoulides GA, ed., Modern Methods for Business Research. Lawrence Erlbaum Associates, Mahwah, NJ, 1998. pp. 295–336.

[8] Chuttur M. Overview of the Technology Acceptance Model: Origins, Development and Future Directions. Working Papers on Information Systems 2009, 9(37), http://sprouts.aisnet.org/785/1/TAMReview.pdf

[9] Compeau DR, Higgins CA. Computer self-efficacy: development of a measure and initial test. MIS Quarterly 1995, 19(2): 189-211.

[10] Connolly C, Murphy E, Moore S. Programming Anxiety Amongst Computing Students - A Key in the Retention Debate? IEEE Transactions on Education, 52(1): 52-56.

[11] Davis FD. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly 1989, 13(3): 319-340.

[12] Fishbein M, Ajzen I. Belief, Attitude, Intention and Behavior: An Introduction to Theory and Research. Addison-Wesley, Reading, MA, 1975.

[13] Fornell C, Bookstein FL. Two structural equation models: LISREL and PLS applied to consumer exit-voice theory. Journal of Marketing Research 1982, 19(4): 440-452.

[14] Fornell C, Larcker DF. Evaluating structural equation models with unobserved variables and measurement error. Journal of Marketing Research 1981, 18(1): 39–50.

[15] Gefen D, Straub D, Boudreau MC. Structural equation modeling and regression: guidelines for research practice. Communications of the Association for Information Systems 2000, 4(7): 2–77.

[16] Gong M, Xu Y, Yu Y. An Enhanced Technology Acceptance Model for Web-Based Learning. Journal of Information Systems Education 2004, 15(4): 365-374.

[17] Götz O, Liehr-Gobbers K, Krafft M. Evaluation of Structural Equation Models Using the Partial Least Squares (PLS) Approach. In Esposito Vinzi V, Chin WW, Henseler J, Wang H, eds. Handbook of Partial Least Squares: Concepts, Methods and Applications, Springer, Heidelberg, 2009, pp. 691-711.

[18] Hair JF, Black WC, Babin BJ, Anderson RE, Tatham RL. Multivariate data analysis, Sixth edition, New Jersey: Prentice-Hall, 2006.

[19] Igbaria M, Chakrabarati A. Computer anxiety and attitudes towards microcomputer use. Behaviour & Information Technology 1990, 9(3): 229-241.

[20] Jackson CM, Chow S, Leitch RA. Toward an understanding of the behavioral intention to use an information system. Decision Sciences 1997; 28(2): 357-389.

[21] Lee Y, Kozar K A, Larsen R T. The Technology Acceptance Model: Pat, Present, and Future. Communications of the Association for Information Systems 2003, 12(1): 752-780

[22] Legris P, Ingham J, Collerette P. Why do people use information technology? A critical review of the technology acceptance model. Journal Information & Management 2003, 40(3): 191-204.

[23] Lohmöller J. –B. Latent variable path modeling with partial least squares. Heidelberg: Physica, 1989.

[24] Moon JW, Kim YG. Extending the TAM for a World-Wide-Web context. Information & Management 2001, 38(4): 217-230.

[25] Nunnally JC. Psychometric Theory, Second Edition, McGraw Hill, New York, 1978.

[26] Parayitama S, Desai KJ, Desai MS, Eason MK. Computers in Human Behavior 2010, Computer attitude as a moderator in the relationship between computer anxiety, satisfaction, and stress 26(3): 345-352.

[27] Radošević D, Orehovački T, Lovrenčić A. Verificator: Educational Tool for Learning Programming. Informatics in Education 2009, 8(2): 261-280.

[28] Radosevic D, Orehovacki T. An Analysis of Novice Compilation Behavior using Verificator. Proceedings of the 33rd International Conference on Information Technology Interfaces (ITI), 2011 June 27-30; Cavtat, Croatia. Zagreb: SRCE University Computing Centre, University of Zagreb; 2011. pp. 325-330.

[29] Ringle CM, Wende S, Will A. SmartPLS 2.0. Retrieved February 2, 2012 from http://www.smartpls.de

[30] Rogers E M. Diffusion of Innovations (5th edition), New York, NY, Free Press, 2003

[31] Taylor S, Todd PA. Understanding information technology usage: A test of competing models, Information Systems Research 1995, 6(2): 144–176.

[32] Van der Heijden, H. Factors influencing the usage of websites: the case of a generic portal in the Netherlands. Information & Management 2003, 40(6): 541–549.

[33] Venkatesh V, Davis FD. A model of the antecedents of perceived ease of use: development and test. Decision Sciences 1996, 27(3): 451–481.

[34] Venkatesh V. Determinants of Perceived Ease of Use: Integrating Control, Intrinsic Motivation, and Emotion into the Technology Acceptance Model. Information Systems Research 2000, 11(4): 342-365.

[35] Venkatesh V, Bala H. Technology Acceptance Model 3 and Research Agenda on Interventions. Journal of Decision Science 2008, 39(2): 273-315

[36] Wold H. Path models with latent variables: The NIPALS approach. In Blalock HM, Aganbegian A, Borodkin FM, Boudon R, Capecchi V, eds. Quantitative sociology: International perspectives on mathematical and statistical modeling. New York: Academic Press, 1975. pp. 307-357.