Digital Worked Example: An Experiment on Strategies to Enhance Computer Programming Skills

M. Reginamary, R. Bavani, Su Ting.Yong

Abstract: ***Learning programming in an example-based learning condition is beneficial to learners with low prior knowledge. This study explored digital worked example (DWE) with an instructional explanation to promote completion strategy and faded worked example (FWE) using screencast technology to promote self-explanation activity. Digital worked example (DWE), integrated with instructional explanation, a complete program that written in C-IDE and saved as a C source file for learners to run and modify to view the output. Fading worked example (FWE) designed to support interactivity using screencast; displays the steps to solve programming problems and fades one or more steps to promote self-explanation effect. Foundation Engineering students enrolled in C programming course for the first time participated in this study. An experiment with a control group (used DWE, n=53) and an experimental group (used FWE, n=51) carried out over seven weeks. Results from this study show that the effect of screencast not significantly effective when transferring basic programming knowledge to gain program writing skills. The DWE with instructional explanation is suitable for novices, and the impact of fading techniques in a worked example (FWE) using screencast is significantly effective in the near transfer performance but not for far transfer performance. However, participant perceived learning experience using worked example with different strategies shows a higher preference for FWE than DWE. The outcome of this study makes room for future research to focus on cognitive overload principles when using screencast technology for a worked example.***

Index Terms**:** ***digital worked example, fading technique, programming learning, screencast***

I. INTRODUCTION

A worked example includes the steps to solve a problem [1], and it has been examined in various instructional procedures and strategies [2, 3] and the effectiveness when employed in computer-based learning condition [4]. Renkl [5] note that strong understanding is possible when worked examples used in initial cognitive skill acquisition and when preparing learners for problem-solving skills beforehand by acquiring a basic understanding of the domain. A standard worked example (SWE) comprises a programming problem, and a complete set of programming statements and (example program) use a completion strategy that requires the modification of the existing computer program. Completion strategy is superior to the conventional approach that requires learners to design and write programs from scratch [6]. Brusilovsky [7] note that novices use the standard worked example to create a new program or to solve fundamental programming problems.

Predominately, SWE in a printed text has been one of the essential learning materials. Instructional explanation in SWE has been a strategy to enhance programming knowledge and understanding. Digital worked example (DWE) is a sample program compiled and tested beforehand and free from syntax errors. The use of DWE in programming learning is scant compared to printed SWE. The printed SWE is to encourage learners to understand the stages in program development such as create, compile, check for errors, execute, and view the output. Once the learners have gained the initial skills, they would be able to solve similar programming problem. However, as the learners develop higher writing skills, the use of printed SWE becomes a tedious programming task. Therefore, DWE with instructional explanation useful once learners have acquired the fundamental program writing skills.

Faded worked example (FWE) omits one or more steps gradually to promote self-explanation effect [8]. The fading strategy supports self-explanation when learner opts to find the missing steps, and Salden [9] note that self-explanation increase learners' understanding. In programming learning, the integration of screencast technology used to engage learners [10] and scaffold learning [11]. A screencast is an innovative learning technology that has recently gained research interest in the programming learning domain [12, 13]. A screencast is a digital recording of a series of actions on the screen and used to capture program snippet. Skudder [14] suggests that further exploration in fading techniques and procedures required, as the use of screencast in promoting the self-explanation effect in FWE for programming skills is scant in the literature. The purpose of this study is to compare digital worked example strategies to enhance programming skills. Digital worked example (DWE), integrated with instructional the explanation is a complete program that has been written in C-IDE and saved as a C source file for learners to run and modify to view the output. Fading worked example (FWE) designed to support interactivity using screencast; displays the steps to solve programming problems and fades one or more steps to promote self-explanation effect.

Revised Manuscript Received on October22, 2019.
Reginamary Matthews, Faculty of Engineering, University of Nottingham, Selangor, Malaysia.
Bavani Ramayah, Faculty of Engineering, University of Nottingham, Selangor, Malaysia.
Su Ting Yong, Faculty of Engineering, University of Nottingham, Selangor, Malaysia.

II. REVIEW OF LITERATURE

A. Digital worked example (DWE)

Huang [15] note that learners found to have difficulty in transferring skills when exposed to a standard worked example (SWE). Several studies proposed self-explanation and instructional explanation strategies to overcome this shortcoming. Wittwer [16] suggest that instructional explanations help learners to (i) understand the problem exhibited in the worked example (ii) develop complete understanding and (iii) detect misconceptions. Robins, Rountree [17] classify the studies related to programming learning into two aspects, i.e. (a) program comprehension; the learner's ability to demonstrate how a program works and, (b) program generation; the ability to write a program to solve programming problems. Program [17] comprehension is essential before a novice learner attempts to write a program. In short, programming comprehension requires strong basic programming knowledge, for a novice to explain the semantics and mechanisms of a program and the generation of a program involves several complicated tasks such as design, development, compilation, testing and debugging.

Novice learners' ability to understand the programming concept and syntax not solely depends on the type of programming language they learn. Object-oriented programming languages such as Java, C++, and C# regarded as difficult compared to procedural /imperative programming languages such as C and FORTRAN. Python programming has gain popularity recently because of easy programming syntax. Learning programming concept is difficult [18] for the first time learners, even the programming syntax is easy to understand. C and C++ programming languages are still popular engineering courses that taught as introductory modules. The difficulty of programming learning arises when learner fails to understand the underlying programming concepts when solving a programming problem and becomes intense when the programming language syntax is also difficult.

The ability to write a program comes with effective practical and problem-solving skills, including a worked example. The basic C programming concepts taught for novices include variables, operators, selection control structures, iterative control structures, and the concept of an array, functions, and pointers. The concepts classified into levels based on the complexity of learning. A strong grasp of the basic programming concepts (variable, operators, selection control structure, and iterative control structure) is essential to assimilate difficult concepts (the concept of an array, functions, and pointers). The concept of the loop [17] and array [19, 20] regarded as difficult for novices.

B. Faded worked example (FWE)

Renkl [21] and Clark [22] suggest fading techniques used in worked example opted for promoting self-explanation to foster cognitive skills. Gradual fading of worked steps engage learners to work out the problem gradually. Gregory [23] note fading technique in the worked example had enhanced learners overall problem-solving skills. However, the use of technology to support the fading effect in worked examples is a novel approach [24]. For instance, Schwonke, Renkl [4] tested the worked example effect in an intelligent tutoring system and found it is robust even when compared to well-structured problem-solving conditions.

C. Screencast to aid fading technique in worked

A screencast is a digital recording of a series of actions on the screen that may include audio narration or music to demonstrate how to use software applications. The empirical evidence on the effect of worked example screencast in a programming learning environment is not apparent. However, related experimentation with screen casting technologies examined students' perception towards mini-lecture and homework solution screencasts and course performance among engineering majors. The outcome of the study suggested that screencasts are useful for explaining concepts and procedures [12]. Other related studies developed screencasts to demonstrate how to install a web server software or how to include a table in a word processor, proposed instructional strategies for developers [25]. Lee [13] note, there is no significant effect of Blue J screencast in the object-oriented programming domain that used to scaffold learning.

III. METHOD

Foundation in Engineering students enrolled in C programming course participated in this experiment. The participants randomly assigned to a control group (n=71) and an experimental group (n=70). Participants in the control group used digital worked example (DWE) incorporated with instructional explanation, and the experimental group used faded worked example (FWE) supported with screencast technology to promote self-explanation.

A. Instructional Treatments

Transfer problems related to programming concepts such as variables, data types, C operators, and selection control structure, for loop and array developed. Cooper [1] suggest that too many examples would result in extraneous load. Renkl [5] define the extraneous load as irrelevant mental activities during learning. Therefore, two to four examples are developed based on the difficulty of the programming concepts. In sum, fourteen digital worked examples interspersed with the instructional explanation. Figure1 describes two digital worked examples with instructional explanation for each programming concepts (Lesson 1 –variables, Lesson 2 - data types, Lesson 3 - C operators, Lesson 4 - selection control structure, Lesson 5 - for loop) and Figure 2 describes four digital worked examples developed to support the learning of the concept of the array (Lesson 6). Figure 3 depicts the screenshot of one of the digital worked examples (programming problem 1) for the concept of the array (Lesson 6).

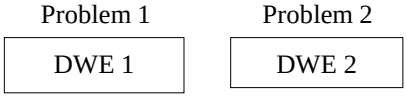


Figure 1. Digital worked example for basic programming concept

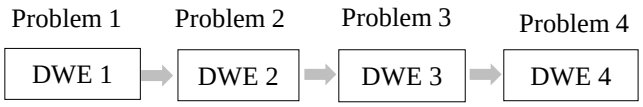


Figure 2. Digital worked example for the concept of array

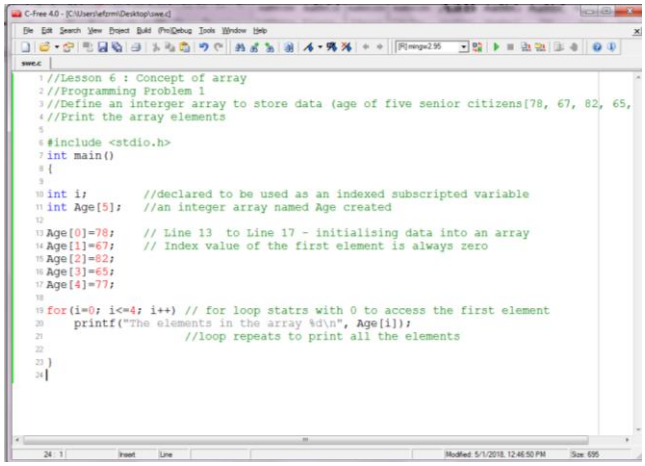


Figure 3. Screenshot of DWE for the concept of array

Six faded worked example screencasts developed with two to four examples. The first five faded worked example screencast created for basic programming concepts (Lesson 1 to Lesson 5) and each integrated two examples (see Figure 4). Figure 5 describes the fading technique; the first example is complete, and the latter faded one or more steps based on the difficulty of the problem.

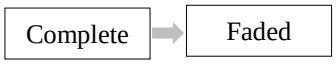


Figure 4. Faded worked example for basic programming concept

Figure 5. Faded worked example for the concept of array

Figure 6 depicts the screenshot of one of the faded worked examples for the concept of the array. The complete worked example displayed the screencast (actual recording of the program snippet in C-IDE) following the sequence of lines of the statement in a program. The underlying computing concept applied to show the steps (step 1: variable declaration, step 2: Input operation (reading data), step 3: control structures (process) and step 4: output operation (displaying output) to solve problems in writing C programming statements. Based on the programming problems and concepts, the number of steps to solve the problem differed.

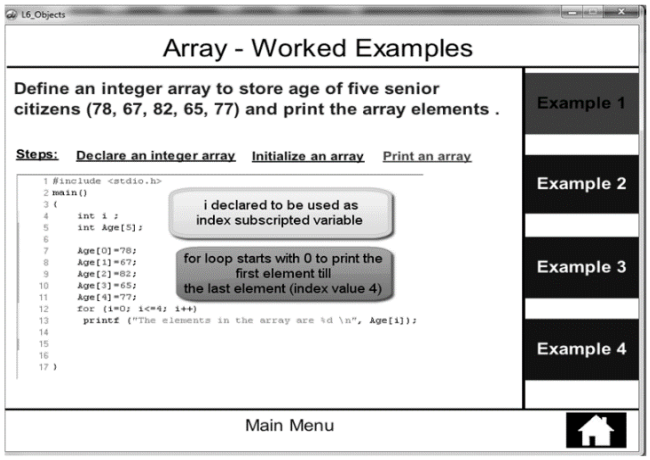


Figure 6. Screenshot of faded worked example using screencast for the concept of array

B. Research Question

This study aims to address the following research questions:

(1) Would digital worked example or faded worked example influence to enhance program-writing skills?

(2) Would digital worked example or faded worked example strategies influence learners' program learning experience?

C. Experimental design and procedure

This study carried out in a computer lab for three hours over seven teaching weeks and comprised of a treatment phase and a transfer phase. A pre-test administered before the treatment phase. In the treatment phase, the lab session is divided into three instructional approaches (learn, practice and assess) to learn six programming concepts (variables, data types, C operators, selection control structure, for loop and, array).

D. Treatment phase

In the first hour –lecture integrated with cognitive activities. In the second hour – guided practical session, the control group received programming problems and DWE while the experimental group received FWE (for the same programming questions). In the final hour – an unguided practical session where participants are asked to practice writing a program for the given programming problems without the teacher's guidance. The same programming teacher intervened in both groups, set in a well-structured problem example condition and given the programming problems and worked examples (DWE / FWE). Participants are encouraged to raise their doubts, and the programming teacher gave feedback when required. During the treatment week, the participants perceived programming learning experience measured with the Likert rating scale on six items as follows; 1– strongly disagree, 2 – disagree, 3–uncertain, 4–agree and 5–strongly agree. Three learning experience survey items designed to measure the perceived relevance (item A1), perceived challenge (item A2), and perceived confidence (Item A3) of DWE and FEW.

The other three items measured the effectiveness of the DWE / FWE helping participants to write a program (item A4 - the ability to write a program), usefulness in solving problems (item A5) and the learners' preference (item A6).

Treatment phase

Near transfer assessment administered after the concepts of the array learned in the first hour and after both groups have used the instructional treatments. A 15-minute near transfer test in the final hour of the lab session conducted to measure participants' ability to solve programming problems using the concept of an array. The programming problem resembled the issues displayed in the worked examples. Far transfer assessment (one-hour post-test) administered after the experimental week. The test consisted of two parts; part A with thirty multiple-choice questions (maximum score 30) to measure basic programming knowledge and part B required participants to write a program using concept or array to measure program writing skills (maximum score 10).

IV. RESULTS

Thirty-three percent (n=18) of the participants in the control group and thirty-seven percent (n=19) of the participants in the experimental group did not sit for the post-test. Therefore, data trimming performed, and the final sample size is one hundred and four. Fifty-three participants observed in the control group and fifty-one in the experimental group. Ninety percent (n=46) of male participants observed in the experimental group, while the control group consisted of seventy percent (n=37). More female students observed in the control group (n=16) than in the experimental group (n=5).

Table 1. Descriptive statistics

	Group	mean	SD	Equality of variance	*r*
Pre-test	G1	3.81	2.228	.273**	-
	G2	3.73	1.888		
near transfer	G1	5.47	2.614	-	0.18^{S}
	G2	6.12	1.956		
far transfer	G1	21.87	2.450	.061**	0.26^{M}
	G2	20.18	3.751		
far transfer	G1	6.37	2.691	-	0.35^{L}
	G2	4.14	3.151		

G1 – control G2-experimental

Ninety-four percent of the novice programming students aged in the range of 17-19 years old and five percent in the range of 20-22 years old. Test of normality using Shapiro-Wilk set α at .001 shows pretest and far transfer measure on basic programming knowledge are statistically significant.

Levene test for pre-test (control group: mean=3.81, SD=2.228; experimental group: mean=3.73, SD=1.888) suggest equal variances between groups and independent-sample t-test t(102)=.212, p>.05 indicate no significant difference. This result suggests the prior programming knowledge between the groups significantly similar. Independent sample t-test for far transfer test (control group: mean=21.87, SD=2.450; experimental group: mean=20.18, SD=3.751) on the basic programming concepts t(102)=2.733, p>.05 indicate significantly no difference exists between groups (see Table 1).

Table 2. Comparison of DWE and FEW

items	Group	chi-square	df	p
perceived relevance and program writing ability	G1	20.847	9	.013*
	G2	27.873	9	.001*
perceived challenge and useful to solve a problem	G1	24.585	12	.017*
	G2	23.258	8	.003*
perceived confidence and preference	G1	30.216	12	.003*
	G2	20.028	12	.067

G1 – control G2-experimental

A. Program writing skills

Mann–Whitney selected to accommodate the non-normal distribution of data and tested at α = .05. The result for near transfer test suggests (U=1067, p=.060) that there is no significant difference between the DWE (mean rank = 47.15) and FWE (mean rank =58.08). However, far transfer test on program writing suggests (U=774.5, p=.000) significant difference between DWE (mean rank = 63.39) and FWE (mean rank = 41.19). Large and medium effect sizes observed for DWE in far transfer performance in both basic programming knowledge gaining and program writing skills (see Table 1).

B. Perceived learning experience

The Cronbach's alpha for all the survey items is 0.78 and indicates a good internal consistency. Findings from survey item A1 suggested that FWE received a slightly higher positive response (n=36, 71%) than the DWE (n=35, 66%) [Item A1: The digital worked example / faded worked example used to learn C programming facilitated a better learning experience]. Forty-three percent (n=23) of the participants in the control group (DWE) responded that they like to solve challenging questions and forty-five percent (n=24) of them are uncertain whether they would be able to solve challenging programming questions. A similar pattern observed in the experimental group (FWE). Only forty-nine percent of the participants (n=25) able to solve challenging problems and the uncertainty observed is similar to the control group (n=21, 41%)

[Item A2: I like to solve challenging programming questions]. Participants indicated that they have the confidence to write a program after using the FWE (n=29, 57%) but the DWE did not increase the participants level of confidence (n=20, 37%) [Item A3: I have the confidence to write C programs]. FWE using screencast has helped learners to recall the steps to write a program (n=37, 73%) than DWE (n=28, 53%) [Item A4: I can remember and understand the steps to write program after using DWE / FWE]. However, participants find DWE more useful (n=44, 83%) than FWE (n=38, 74%).

Meanwhile, the response to the uncertainty about the relevance is observed lower in the control group (n=7, 13%) than experimental group (n=13, 25%) (Item A5: The digital worked example / faded worked example useful to solve programming questions in the practical session). Interestingly, participants preferred the DWE (n=44, 83%) than the FWE (n=38, 75%) [Item A6: novice learners like me need DWE / FWE to learn program effectively].

C. *Comparing DWE and FEW*

A Chi-square test performed to examine the association between the survey items. Table 2 shows that participants found DWE and FWE relevant in the example-based learning condition, and it has helped to foster program writing ability. Besides, participants who attempted to solve challenging or complicated problems found FWE and DWE useful in guiding them. However, the significant association found between perceived confidence and preference in using DWE. Thirty seven percent of the participants indicated having the confidence to write a program using DWE and preferred to use it.

V. DISCUSSION

Digital worked example (DWE) with instructional explanation and faded worked example (FWE) using screencast technology to promote self-explanation compared to study its effectiveness when fostering near and far transfer on programming basic knowledge and writing skills. The overall results indicate, DWE significantly useful when fostering both basic programming knowledge and program writing skills.

A. Instructional and self-explanation strategies

The ability to write a computer program is a skill acquired with lots of practice. Instructional explanation incorporated in DWE supports the acquisition of essential programming learning and writing skills. Findings from this study suggest instructional explanation in digital worked example significantly useful to foster programming skills and fading strategy to promote self-explanation using screencast technology is not significant. Empirical evidence suggests the self-explanation effect in worked example superior to instructional explanation [26] and instructional explanation supplemented with self-explanation yields a positive impact [27].

Sern [28] studied instructional, and self-explanation prompts in engineering content for novice learners, conclude self-explanation promote better near-transfer performance and similar far-transfer performance for both approaches. A similar result observed from this study, fading effect using screencast does help to encourage near transfer, but the detrimental effect of fostering far transfer performance. Jalani [29] notes one of the limitations of near-transfer is the inability of a learner to apply knowledge or skill gained when the learning condition changes. The faded strategy does support programming learning with modest effect; however, the use of screencast technology may impede learning.

The size of the screencast could be one of the reasons to cease the effect in supporting program writing skills attainment. Morris [30] note the length of the screencast is crucial to learner's engagement. Participants viewed the digital worked example without any interactivity, whereas faded worked example used interactivity feature to show screencast for each step. Increased mental effort load is possible when participants process each screencast to understand the example program. Duration of learning is vital for schema automation to occur [31], and high element interactivity is challenging to understand [32].

Green [12] suggest that screencast works if learners understand the relevancy of learning. Findings from this study show a significant association between the perceived learning experience and program writing ability (see Table 2). Participants perceived FWE positively (n=38, 74%) indicated that they can write computer program (n=37, 74%). However, participants' perceived relevance not reflected in the far transfer on program writing skills. Participants' more likely to solve challenging programming problems using FWE and DWE observed to be significantly moderate, and they perceived both worked examples useful in solving programming problems. Novice learners perceived confidence level is poor when using DWE compared to FWE.

In summary, learning programming in an example-based learning condition is beneficial for the learner with lower prior programming knowledge. The findings from this study showed that the digital worked example with instructional explanation (DWE) had helped novices to understand programming concepts and syntax before attempting to write a program. Results show a well-designed digital worked example with instructional explanation significantly fosters programming skills attainment. The fading strategy in worked example able to promote near transfer performance but insignificant for far transfer performance. Participants perceived learning experience using different approaches in worked example indicates participants' preference for faded worked example (FWE) higher compared to digital worked example (DWE).

The self-explanation using screencast in fading worked example makes room for further examination on the factors that hinder far transfer performance. Extraneous load in the worked example may impede learning. Splitting screen to reduce cognitive overload assessed to investigate the merits and pitfalls of screencast when integrated into worked example.

ACKNOWLEDGMENT

Our thanks to the University of Nottingham Malaysia Campus Foundation Engineering Students.

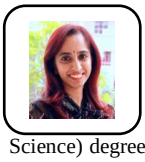
REFERENCES

1. Cooper, G., *Research into cognitive load theory and instructional design at UNSW. University of New South Wales, Australia.* 1998.
2. Abdul-Rahman, S.-S. and B. du Boulay, *Learning programming via worked-examples: Relation of learning styles to cognitive load.* Computers in Human Behavior, 2014. **30**: p. 286-298.
3. Moreno, R., *When worked examples don't work: Is cognitive load theory at an impasse?* Learning and Instruction, 2006. **16**(2): p. 170-181.

4. Schwonke, R., et al., *The worked-example effect: Not an artefact of lousy control conditions.* Computers in Human Behavior, 2009. **25**(2): p. 258-266.
5. Renkl, A., & Atkinson, R.K., *Structuring the transition from example study to problem-solving in cognitive skill acquisition: A cognitive load perspective.* Educational Psychologist, 2003. **38, 1**: p. 15-22.
6. Van Merriënboer, J.J.G., *Strategies for Programming Instruction in High School: Program Completion vs. Program Generation.* Journal of Educational Computing Research, 1990. **6**(3): p. 265-285.
7. Brusilovsky, P., Sosnovsky, S., Yudelson, *Adaptive e-learning service for accessing interactive examples.* Conference on E-Learning in Corporate, Government. Healthcare, and Higher Education, (E-Learn, World Conference on E-Learning, At Chesapeake, 2004: p. 2556 – 2561.
8. Renkl, A., Atkinson, R.K., and Maier, U.H, *From studying examples to solving problems: Fading worked-out solution steps helps learning.* Proceedings of the 22nd Annual Conference of the Cognitive Science Society, 2000.
9. Salden, R.J.C.M., Aleven, V., Schwonke, R. et al., Instr Sci, 2006. **38**: p. 289.
10. Ruffini, M., *Screencasting to engage learning*, in *EDUCAUSE Review*. 2012.
11. Thompson, R., *Talking with students through screencasting: Experimentations with video feedback to improve student learning.* Journal of Interactive Technology and Pedagogy, 2012. **1**: p. 1.
12. Green, K.R., Pinder-Grover, T., and Millunchick, J.M, *Impact of screencast technology: Connecting the perception of usefulness and the reality of performance.* Journal of Engineering Education, 2012. **101**(4): p. 717-737.
13. Lee, M.J.W., Pradhan, S. and Dalgarno, B. *The effectiveness of screencasts and cognitive tools as scaffolding for novice object-oriented programmers.* Journal of Information Technology Education, 2008. **7**: p. 61-80.
14. Skudder, B., and Luxton-Reilly, A, *Worked examples in computer science.* Proceedings of the Sixteenth Australasian Computing Education Conference 2014: p. 5964.
15. Huang, X., and Reiser, R. *The effect of instructional explanations and self-explanation prompts in worked examples on student learning and transfer.* International journal of instructional media, 2012. **39**(4): p. 349-362
16. Wittwer, J., and Renkl, A, *How effective are instructional explanations in example-based learning? A meta-analytic review.* Educational Psychology Review, 2010. **22**(4): p. 393-409.
17. Robins, A., J. Rountree, and N. Rountree, *Learning and Teaching Programming: A Review and Discussion.* Computer Science Education, 2003. **13**(2): p. 137-172.
18. Öqvist, M.N., J. J, *Coding by hand or on the computer? Evaluating the effect of assessment mode on performance of students learning programming.* Comput. Educ. , 2018(5): p. 199.
19. Moss, B., *Iwave: Interactive web-based algorithm visualization environment*, in *Internal Visual Learning Lab Report*. 2007, University of Nottingham: UK.
20. Kaila, E., Rajala, T., Laakso, M.J., and Salakoski, T. *Effects of course-long use of a program visualization tool.* Proceedings of the Twelfth Australasian Conference on Computing Education 2010: p. 97-106.
21. Renkl, A., *The Worked-Out Examples Principle in Multimedia Learning*, in *The Cambridge Handbook of Multimedia Learning*, R. Mayer, Editor. 2005, Cambridge University Press: Cambridge. p. 229-246.
22. Clark, R., Nguyen, F., and Sweller, J, *Cognitive load and efficiency in learning. Efficiency in learning*. Evidence-based guidelines to manage cognitive load. 2006: Cambridge.
23. Gregory, T.L.H.J.L., *Exploring the Use of Faded Worked Examples as a Problem Solving Approach for Underprepared Students.* Higher Education Studies, 2015. **5**(6): p. 36-46.
24. Ryland, A., *Worked examples: Teacher practices.* International HETL Review 2013. **3**: p. 8.
25. Sugar, W., Brown, A., and Luterbach, K., *Examining the anatomy of a screencast: Uncovering common elements and instructional strategies.* The International Review of Research in Open and Distributed Learning, 2010. **11**(3): p. 119.
26. Schworm, S., and Renkl, A, *Learning by solved example problems: Instructional explanations reduce self-explanation activity.* Proceedings of the Cognitive Science Society, 2002: p. 816-821.
27. Reisslein, J., Atkinson, R.K., Seeling, P., and Reisslein, M., *Encountering the expertise reversal effect with a computer-based environment on electrical circuit analysis.* Learning and Instruction, 2006. **16**(2): p. 92-103.
28. Sern, L.C., *Learning with worked-out problems: The impacts of instructional explanation and self-explanation prompts on transfer performance.* Journal of Technical Education and Training, 2010. **2**(2).
29. Jalani, N.H., and Sern, L.C, *Effects of example problem based learning on transfer performance in circuit theory.* Journal of Technical Education and Training, 2014. **6**(2).
30. Morris, C., and Chikwa, G, *Screencasts: How effective are they and how do students engage with them?.* Active Learning in Higher Education, 2014. **15**(1): p. 25-37.
31. Van Gog, T., Paas, F. & van Merriënboer, J.J., *Process-oriented worked examples: Improving transfer performance through enhanced understanding.* Instructional Science 2004: p. 32:-83.
32. Van Merrienboer, J.J.G., and Sweller, J, *Cognitive load theory and complex learning: Recent developments and future directions.* Educational Psychology Review, 2005. **17**(2): p. 147-177.

AUTHORS PROFILE

Regina is an Assistant Professor at the Faculty of Engineering, Engineering Foundation program. She received her Bachelor degree in Computer and Mathematical Sciences from Victoria University of Technology (Australia) and Ph.D. in Creative Multimedia from Multimedia University (Malaysia). Dr. Regina has more than twenty years of teaching experience in higher learning institutions. Her main research area includes multimedia learning objects in teaching and learning an introductory programming language, learning analytics (educational data mining) and technology support for individuals with learning disability.



Bavani is an Assistant Professor where she has been a faculty member since 2007 and has obtained her Ph.D. (Visual Informatics) from National University of Malaysia and BSc (Computer Science), MSc (Computer Science) degrees, from the University Putra Malaysia (UPM). Dr. Bavani was working in the industry for four years as a programmer, prior to the education field as a lecturer. Her research interests are in the area of Human-Computer Interaction (HCI) such as Social Computing, Behavioural sciences, Physiological interaction, User experience (UX), Emotional Design, Visually Impaired Users' related studies the and social media. Besides, she has collaborated actively with researchers in several other disciplines such as political sciences, law, and psychology.

Su-Ting Yong is an Assistant Professor in the Department of Foundation in Engineering, The University of Nottingham Malaysia Campus. She joined the University in 2008, having taught in a few universities for several years. Dr. Yong obtained her Bachelor's Degree in Science and Computer with Education (Mathematics) and a Master's Degree in Information Technology from The University of Technology Malaysia. She completed her Ph.D. in Engineering Education at The University of Nottingham. She is a fellow of The Higher Education Academy and has a wide variety of research interests, mostly focused on technology in mathematics education, educational games, gamification, and programming.