Worked Examples in Computer Science

Ben Skudder **Andrew Luxton-Reilly**

Department of Computer Science
The University of Auckland,
PO Box 92019, Auckland, New Zealand
Email: bsku002@aucklanduni.ac.nz, andrew@cs.auckland.ac.nz

Abstract

Most instructors teaching Computer Science use examples to help students learn, and many instructors use worked examples (either in a static or a dynamic style) in their courses. However, the research on worked examples is not well known in the Computer Science Education community. This paper provides an overview of how worked examples have been studied, and the major findings from the literature, particularly as they relate to Computer Science.

Keywords: cognition, learning, cognitive load theory, worked examples

1 Introduction

Shulman (2005) uses the term *signature pedagogies* to describe pedagogical practice that is characteristic of a given discipline. These are the ways of teaching that spring to mind when we think of a particular discipline — for Medicine, it is the bedside teaching that occurs during clinical rounds where groups of students are involved in discussions with a resident; for Law, it is the case dialogue method in which a complex case is dissected through discussion and argument. We believe that the use of worked examples to demonstrate problem solving and software development is a signature pedagogy for Computer Science. Yet this key pedagogical practice, characteristic of education in Computer Science, has not been widely studied in the very context of Computer Science.

According to Atkinson et al. (2003) "Worked-out examples typically consist of a problem formulation, solution steps, and the final answer itself". A problem is presented, accompanied with step-by-step instructions which lead to the solution. These are usually textual but may include pictures, diagrams or animations. We consider that this definition of worked examples would include dynamic demonstrations of problem solving (such as live demonstrations of writing programs that solve simple problems). Students are expected to study the worked example and from it learn how they might apply it to similar problems. Figure 1 illustrates a typical worked example in Computer Science.

According to Miller (1956), humans have a limited working memory, where only a few chunks of information can be processed at one time. Cognitive load (Sweller 1988, Chandler & Sweller 1991) describes the amount of information that must be held in working memory during the process of problem solving. If the working memory is overtaxed, for example, by trying to solve a problem without enough scaffolding, learning performance will suffer. It is for this reason that Kirschner et al. (2006) argue that problem solving fails to be an effective learning strategy when there is insufficient guidance in place.

Problem statement:

Write a function that calculates the area of a rectangle.

Solution Design:

1. Determine what parameter(s) the function needs to calculate an answer, as well as their type(s):
   - *width (float), height (float)*
2. Determine what result the function will return, including the type:
   - *the area of the rectangle (float)*
3. Determine the steps needed to calculate this result:
   - *To calculate the area of a rectangle we will use the formula:*
     *(area of rectangle = width * height)*

Implementation:

1. Using the identified parameters, write the function header:

```
def rectangle_area(width, height):
```

2. Using the identified steps, calculate the result:

```
def rectangle_area(width, height):
    area = width*height
```

3. Return the final result

```
def rectangle_area(width, height):
    area = width*height
    return area
```

Figure 1: An exemplar worked example

Humans also have a long-term memory with a much larger capacity (Baddeley & Hitch 1974). Long-term memory consists of a set of schemas, and with practice, information stored according to the schemas can automatically be recalled and applied with minimal impact on working memory. In this model of human cognition, the aim of teaching is to help students form appropriate schemas, which can in turn be used to solve both familiar and novel problems.

Recent literature distinguishes between different types

of cognitive loads — *intrinsic* cognitive load, *extraneous* cognitive load and *germane* cognitive load (Paas et al. 2004).

Intrinsic cognitive load is imposed by the degree of interactivity between elements in the problem domain. It may not be reduced unless the content is in turn reduced, and is therefore unaffected by altering the presentation of material by an instructor.

Extraneous cognitive load is caused by activities which do not assist with the formation of schemas. These activities interfere with learning because they require the use of working memory for processes that are not related to the focus of learning.

Germane cognitive load relates to the higher level processes (scaffolding) that supports the formation of schemas, and therefore improve the effectiveness of the activity for learning.

For further information, work by Caspersen & Bennedsen (2007), and Caspersen (2007) provide excellent overviews of the theory of cognitive load theory as it applies in the practice of instructional design for Computer Science.

Although examples, and in particular, worked examples are widely used to teach Computer Science, there are few studies that have investigated their effectiveness in the Computer Science context. In this paper we present the theoretical basis and research findings for worked examples, which may encourage practitioners to be more deliberate about the organization of their own examples. We also show how worked examples have been studied in related fields like engineering and statistics, and examine the literature to identify potential avenues for further research in Computer Science.

2 Ways of presenting worked examples

We first consider the different ways that worked examples can be integrated into the overall instructional design for a given topic.

Examples only: In this approach, students are simply provided with a set of worked examples. There are no activities, such as exercises or problems to solve, associated with the examples.

Example-problem blocks This approach provides students with a block of worked examples of various types to study, then a set of related problems are given which students are expected to solve.

Example-problem pairs These are one of the most common ways of presenting worked examples, where each example is paired with a problem similar to the example for students to complete. Students alternate between studying a worked example and solving a related problem.

Faded worked examples In this approach, a complete worked example is presented, then another worked example with one step missing is presented, and students are expected to fill in the missing step. They are presented with a series of worked examples, with an extra step removed each time, until a student is presented with just a problem to solve.

The most common orders for fading steps are known as forward fading - where steps are removed starting from the beginning, and backwards fading - where steps are removed from the end first.

2.1 Other techniques that support worked examples

These basic forms of presenting worked examples are often augmented with other techniques, such as:

Subgoal labeling A technique where groups of steps are given a label, to help organize the information into a meaningful structure. According to Margulieux et al. (2012) subgoal labels allow students to focus on groups of steps rather than individual steps, giving them fewer problem-solving steps to consider and, reducing cognitive load. The highlighted structure given by subgoals is also supposed to assist with schema formation, or provide "mental model frameworks" to internally explain how problems are solved.

Self explanation prompts Self-explanation is a process some learners undergo when provided with a worked examples. Students who try to explain to themselves the reasons for a step or set of steps in an example were found to learn more than those who don't (Atkinson et al. 2003), so self-explanation prompts are designed to elicit such self-explanations. Self-explanation prompts can be in the form of asking students to justify a step or choosing what principle a particular step is invoking. When employed correctly, these prompts are considered to be a source of germane cognitive load.

3 What is the effectiveness of Worked Examples?

The benchmark for evaluating worked examples is usually some form of problem solving task. The task typically requires a student to solve a problem, and the student is told when their solution is correct. Usually a set of questions is given, and some of these questions are swapped for worked examples — people in the problem solving condition solve all the questions, and people in the examples condition study several examples and solve some problems.

They are also often evaluated for their ability to promote near transfer and far transfer. Near transfer is the ability of students to solve questions which are isomorphic to the ones they saw in their training phase, whereas far transfer is the ability for students to solve novel problems which use many of the same skills from the training phase, but in a different sequence or with some of the learned techniques requiring minor modifications.

3.1 Examples only

The provision of examples over giving problems to solve reduces extrinsic cognitive load and directs student's attention to the relationships between different problem steps, thereby encouraging students to construct relevant problem-solving schemas around it. Problem-solving with no guidance, however, requires a large cognitive load for novices, but all the effort goes to finding an answer rather than schema formation.

Studies have investigated the use of isolated worked examples to illustrate how to solve a given problem in fields such as Accounting (Stark et al. 2002), Electrical Engineering (van Gog et al. 2006), and CNC Programming (Paas et al. 2004).

In the domain of CNC programming, Paas et al. (2004) found that presenting multiple worked examples with high variability resulted in improved learning compared with multiple worked examples with low variability. They also compared worked examples only with a problem-example pair condition, and found that attempting to solve a problem prior to the worked example actually impeded learning.

A later study by van Gog et al. (2011) compared worked examples on their own, example problem pairs, problem example pairs and problem solving on its own for teaching high school students to diagnose a faulty electrical circuit.

The use of worked examples resulted in improved learning and transfer compared with traditional problem-solving techniques. This improvement in learning was also observed in the condition where students were presented with example-problem pairs. Students reported lower mental effort and scored better results upon testing than those in the problem solving condition, or the problem-example paired condition. No difference was found between example-only and example-problem pairs van Gog et al. (2011).

Although it might seem that presenting a problem first would motivate a student to engage more deeply with the worked example, the results of these studies suggest that greatest learning occurs if the worked examples are presented prior to the problem.

3.2 Example-problem blocks

The use of example-problem blocks is uncommon, but has been studied in a programming context. In one notable study, Gregory et al. (1993) compared using example-problem blocks, example problem pairs, alternating similar problem-solving task, and blocks of problem-solving tasks.

The tasks were 6 pairs of LISP programming questions, to solve after having gained some familiarity with LISP before the experiment proper started. Each pair tested the same skills, with one being the source problem and the other being the target. The idea was that the source provided a chance to initially learn to solve the problem, and the target allowed them to practice the techniques learned from the source.

For the example-problem pairs and block conditions source problems were swapped for a worked example. In the block conditions, sources were separated from targets whereas in the pair conditions targets immediately followed sources. In other words, the example-problem paired condition involved a sequence of problems where each problem was preceded with a worked example. The block condition involved a sequence of worked examples, followed by a sequence of problems to solve.

Example-problem blocks were the worst preforming group in post-tests. Students in this condition spent as much time studying source examples as the example-pair group, but spent more time on the target problems. Gregory et al. (1993) suggest that difficulty in remembering the examples once they met the equivalent problem would hinder later problem solving, and that if students are unable to recall the appropriate example, the benefit of studying them over problem solving disappears.

Indeed, both of the problem-solving groups performed better than the example-problem blocks group, suggesting the extra practice afforded to the problem-solving block group outweighed the benefits of having worked examples. The example-problem pairs were the best performing group on post-tests.

3.3 Example-problem pairs

Extensive work by Sweller and his colleagues has established that worked examples, when paired with problems, are superior to problem-solving without worked examples in a variety of subject areas (Sweller & Cooper 1985, Mwangi & Sweller 1998). Fewer studies have compared the use of example-problem pairs with other configurations of example and problem presentation.

The use of example-problem pairs is thought to foster learning better than example-problem blocks, as students can better select and recall the most relevant example (i.e. the one just studied) to relate the problem to when they are given one directly after the other. Separating them may make it harder to recall the relevant example to relate to the current problem.

As described previously in section 3.1, van Gog et al. (2011) compared worked examples on their own, example problem pairs, problem example pairs and problem solving, and found example-only and example-problem pairs to work more effectively than the other conditions. Example-only and example pairs performed similarly.

When example-problem pairs are compared with example problem blocks, results suggest that the example-problems pairs are effective for learning Gregory et al. (1993). Students studying examples in both paired and block conditions spent equal time, but those who were given problems to solve immediately following the examples appeared to be able to solve later problems more efficiently than those students who studied a block of examples prior to practicing the problem solving skills.

Renkl et al. (2002) conducted three experiments comparing backward and forward fading with example-problem pairs. The first experiment compared the effectiveness of backwards fading with example problem pairs for solving Statistics problems. The second experiment compared forward fading with example-problem pairs in the context of Physics. The third experiment compared both forward and backward fading with example-problem pairs. In all three cases, students in the fading conditions outperformed those using example-problem pairs for near transfer problems. Students also produced fewer errors during learning. This suggests fading may offer better learning outcomes in a shorter amount of time for near-transfer tasks than example-problem pairs.

Atkinson et al. (2003) explores the use of backwards fading compared to example-problem pairs for solving statistics problems. Under a variety of conditions, backwards fading resulted in higher post-test results than example-problem pairs on both near and far transfer problems.

3.4 Faded worked examples

Although worked examples appear to be more effective than simple problem solving under a variety of conditions, as a student gains expertise from studying worked examples, the benefits of studying them over problem solving disappears (Renkl et al. 2002, Atkinson et al. 2003). It is thought that partial schema formation means that the elements that were once a source of germane cognitive load become a source of extrinsic cognitive load. At this point problem-solving without worked examples becomes more effective (Renkl et al. 2004, Kalyuga et al. 2003).

To ease this transition, faded worked examples begin with a fully worked example, but as they study it and gain expertise steps are removed to encourage a manageable amount of problem solving, fostering germane cognitive load. By the end of the fading sequence, students will have studied many of the worked example steps and will be able to problem solve on their own.

Studies investigating the sequence of fading have not produced reliable findings. Renkl et al. (2002) suggested that backwards fading worked would produce better results than forward fading on near transfer items. However, later work was unable to confirm these findings. Renkl et al. (2004) investigated whether the sequence of fading affected near and far transfer more than the types of steps removed. In their two experiments no difference was found in learning outcomes or errors during learning.

Their results also suggested that students learn most

about those steps which are faded. The implication is that the learning activities elicited by removing steps focuses students on those area. For this reason, they suggest the earlier results must be attributed to the learning material they used and the type of the steps removed. The backward procedure removed those steps that may be 'prerequisites' or otherwise helped students learn principles which were helpful for earlier steps. Doing it the other way means they would not learn the important principles first, which would hinder subsequent learning.

Moreno et al. (2006) also compares forward fading and backward fading. Those who used forward fading were found to outperform those using backwards fading. They suggest this has to do with the ease of the material they were learning. Having studied the first example, students may have gained all the initial knowledge they needed.

According to the expertise reversal effect, if a student already has some expertise in the area, further learning is better gained by problem-solving, and techniques like worked examples may hinder or decrease subsequent performance. This is because for an expert, studying a worked example is a source of extraneous cognitive load rather than germane cognitive load, and problem-solving promotes germane cognitive load for those with some expertise in the targeted domain. Because forward-fading gets students to start problem-solving as the first, rather than the last step, the early problem-solving may have benefited students as opposed to those who had to wait until the final step to problem-solve.

3.5 Self-explanation prompts

Atkinson et al. (2003) cites research with mixed results on the effects of activities designed to elicit self explanations. It has been suggested that self-explanations are a source of germane cognitive load, helping students to from schemas around the materials they're learning, rather than e.g. just memorizing a set of steps to a solution. Experiments where students were prompted by an online tool to fill in templates for self-explanations , or where students were encouraged to write their own self-explanations as comments, failed to increase learning gains consistently. Another study found self-explanation prompts during the problem solving phase rather than during example study received positive results on learning.

In their own study into solving statistics problems, Atkinson et al. (2003) prompted students with a set of principles a given step in the worked examples may be drawing from. Students were expected to choose one of the principles, and this was expected to foster self-explanations. Students in the self-explanation groups performed better on post-tests for near and far transfer problem than those not prompted in the equivalent fading or example-pair groups not prompter. No extra time was required to achieve this result. The results for self-explanation prompts with backwards fading were replicated for university and high school students.

3.6 Subgoal labeling

Margulieux et al. (2012) studied the use of subgoal labelling in video demonstrations and instructional material for creating mobile applications. In the subgoal conditions, the steps in the demonstration video and instructional material were labelled with subgoals grouping several steps into a cohesive group.

In post-tests participants in the subgoal group better identified subgoals necessary to complete a solution whether or not they complete it correctly or not. They also were more likely to correctly complete the subgoals necessary for the assessments. Overall the subgoal condition outperformed their counterpart on both assessments immediately after training and assessments one week later. They did so spending less time on the assessments, and were less likely to drag out blocks in the assessments.

3.7 The expertise reversal effect

Although studies of worked examples generally shows positive benefits for learners, Kalyuga et al. (2003) demonstrate instances where providing worked examples can hinder learning.

For novices, worked examples direct their attention to important features of the problem and help in forming relevant problem-solving schemas. This is a better use of their cognitive resources than problem solving, which requires extensive search of the problems space (Sweller 1988). Unguided problem solving imposes a heavy cognitive load unrelated to schema formation. In other words, it is a source of extraneous cognitive load, but not germane cognitive load.

However someone with some expertise already has partial or full schemas in long-term memory. For experts, worked examples are redundant. The effort required to analyse worked examples becomes a source of extrinsic cognitive load rather than germane cognitive load. Kalyuga et al. (2003) identify studies involving trades apprentices, students working with databases and other experiments where people with more experience fail to gain any benefit from worked examples. In these studies, as novices' expertise increases, they learn more from problem solving rather than studying examples.

4 Examples in Computer Science

There is little research into worked examples in Computer Science. Early research into cognitive load theory drew upon work in teaching LISP (e.g. Anderson et al. (1984)), where it was observed that students would rely heavily on provided examples as opposed to instructional texts. Much of the worked example literature rely on the results of these studies, but nonetheless worked examples have not been well studied in Computer Science education, as Merrinboer & Paas (1990) and Mason & Cooper (2012) observed.

A few reports in the CS Education literature focusing on the instructional design of introductory programming courses have advocated the use of worked examples during the course (Hsiao et al. 2013, Caspersen & Bennedsen 2007, Lui et al. 2008, Gray et al. 2007). However, formal studies of worked examples in the context of Computer Science, such as that of Gregory et al. (1993) and Margulieux et al. (2012) are the exception rather than the rule.

4.1 Faded Worked Examples in Computer Science

Gray et al. (2007) provides a detailed discussion of how faded worked examples might be applied in an introductory programming course in Computer Science. We examine their approach in this section. The task of programming is decomposed into components whose cognitive load they claim can be adequately managed. The decomposition is based on two parts: the abstract algorithmic dimensions and the associated concrete programming constructs. The algorithmic dimensions identified are design, implementation and semantics (the meaning of supplied code). The semantic dimension is divided in three, into assertion (students should be able to state true statements about the code at various point of execution), execution (given an input, provide the output) and verification (be able to test the code). The programming constructs chosen were selection, iteration and subroutine calls. Each of these would be taught in pairs (design of a selection

algorithm, implementation of an iterative algorithm etc.), with the learning of each pair supported by sets of faded worked examples.

Concrete, fully worked examples are provided for all of the design-construct and implementation-construct pairs, and provide an example of semantic-assert and semantic for selection algorithm. Although it is useful for instructors considering adopting this pedagogy to have such examples, they have not been used in any formal studies or actual courses.

Although Gray et al. (2007) suggest the use of backwards fading, Renkl et al. (2004) suggests that the success of backwards fading compared to forward fading is an artefact of the teaching materials people use rather than something inherent in the backwards sequencing. The sequencing of fading should be examined to see which steps may be prerequisites for understanding other steps — Renkl et al. (2004) suggests these kinds of steps should be removed first.

The use of 'ASSERT' during the semantic part of training is designed to get students to state what is known about certain parts of code in the form of code comment. This is motivated by the same principals motivating the use of self-explanation prompts in Atkinson et al. (2003). However, it is not clear how students will learn how to develop their own assertions without scaffolding. An explicit process to help students develop assertions is provided for the selection statements, but no such process is provided for other syntax constructs.

As mentioned earlier, the research on self-explanation prompts is not unanimous. Atkinson et al. (2003) suggests the interface allowing students to write down self-explanations may have an effect on whether it will be effective, and the prompts they provide in their own experiment require students to make choices from a list, rather than generating them on their own. This requires a low amount of activity from students. The scaffolding provided means they won't have to come up with assertions from scratch like in some previous studies, but the suggested 'ASSERTS' require a little more than picking options from a list. Further study on the use of self-explanation prompts, or assertions during code development, is required, both theoretical and empirical.

However, all in all, Gray et al. (2007) provide a clear framework for using and testing faded worked examples in Computer Science. Such techniques could straightforwardly be extended to other C derived languages like C, Java or C♯, or any kind of imperative or procedural language. Other constructs or dimension of programming could be considered too.

5 Implications for Computer Science

The use of examples is extremely common in the discipline of Computer Science, particularly in courses that introduce programming concepts. It is fairly typical in lectures, and in most textbooks, for numerous examples of code to be shown to students. These examples frequently take the form of code traces, where the instructor presents some code and proceeds to demonstrate how it would be executed by tracing the execution one step at a time; and problem solution pairs, in which a problem is posed by the instructor (e.g. "Write a method that determines whether a given number is a prime number or not"), and a solution is subsequently presented and the code is explained in detail. Less commonly, instructors may demonstrate the development of software by programming in real time during the lecture.

However, it is far less common for students to engage in problem solving activities during lecture time. Certainly, reports of active learning in the Computer Science classroom illustrate how such activities are possible, but these are not widespread in practice. In most courses, it is only much later, during homework or in laboratory sessions, that students solve problems similar to those covered during lectures. In other words, most courses use the instructional design of example-problem blocks. Although the use of example-problem blocks has not been extensively studied, there is some indication that it is one of the least effective approaches Gregory et al. (1993).

It is possible that some of the difficulties observed in the novice programming literature may be due to intrinsic cognitive load imposed by the complexity of programming tasks. If, as claimed by Sweller & Chandler (1994), programming is an intrinsically difficult area, then it is extremely important to minimize the extraneous cognitive load if students are to be successful. Although the studies presented here suggest that some ways of organising worked examples are more effective than others, more research on the cognitive load imposed by programming is required to better understand how to organise and present content in the most effective way.

Additionally, it may be beneficial for practitioners to reflect on the organisation of their course material in the light of the studies discussed here. Some simple changes in the way examples and exercises are structured could improve learning for students in most programming courses.

6 Conclusions

The evidence suggests certain worked example techniques (primarily example-problem pairs and faded worked examples) are an improvement over standard problem solving techniques, in terms of learning time and performance on near transfer tests in novices.

In situations where the student is not a novice, faded worked examples appear to improve performance and decrease learning time on near transfer tasks. In addition, techniques such as self-explanation prompts may promote far transfer as well if applied appropriately.

Since much of the research involves well structured domains like Statistics, Physics and Engineering, it is likely that findings would transfer readily to the domain of Computer Science. However, further studies are required to confirm the effectiveness of pedagogies based on worked examples in the context of Computer Science. The use of faded worked examples with self-explanations has the potential to help students to learn more effectively, but the best order of fading problems is currently unknown Renkl et al. (2004). Future research into what steps should be faded first for a given problem in Computer Science would also help us understand how faded worked examples could most effectively be employed.

References

Anderson, J. R., Farrell, R. & Sauers, R. (1984), 'Learning to program in lisp', *Cognitive Science* **8**(2), 87 – 129.

Atkinson, R. K., Renkl, A. & Merrill, M. M. (2003), 'Transitioning from studying examples to solving problems: Effects of self-explanation prompts and fading worked-out steps', *Journal of Educational Psychology* **95**(4), 774–783.

Baddeley, A. & Hitch, G. (1974), Working memory, *in* G. Bower, ed., 'Recent advances in learning and motivation', Vol. 8, Academic Press, New York, pp. 47–90.

Caspersen, M. (2007), Educating Novices in the Skills of Programming, PhD thesis, University of Aarhus, Denmark.

Caspersen, M. E. & Bennedsen, J. (2007), Instructional design of a programming course: a learning theoretic

approach, *in* 'Proceedings of the third international workshop on Computing education research', ICER '07, ACM, New York, NY, USA, pp. 111–122.

Chandler, P. & Sweller, J. (1991), 'Cognitive load theory and the format of instruction', *Cognition and Instruction* pp. 293–332.

Gray, S., St. Clair, C., James, R. & Mead, J. (2007), Suggestions for graduated exposure to programming concepts using fading worked examples, *in* 'Proceedings of the third international workshop on Computing education research', ICER '07, ACM, New York, NY, USA, pp. 99–110.

Gregory, J., Trafton, G. & Reiser, J. (1993), 'The contributions of studying examples and solving problems to skill acquisition'.

Hsiao, J.-Y., Hung, C.-L., Lan, Y.-F. & Jeng, Y.-C. (2013), 'Integrating worked examples into problem posing in a web-based learning environment', *The Turkish Online Journal of Educational Technology* **12**(2), 166–176.

Kalyuga, S., Ayres, P., Chandler, P. & Sweller, J. (2003), 'The expertise reversal effect.', *Educational Psychologist* **38**(1), 23 – 31.

Kirschner, P. A., Sweller, J. & Clark, R. (2006), 'Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching', *Educational Psychologist* **41**(2), 75–86.

Lui, A. K., Cheung, Y. H. Y. & Li, S. C. (2008), 'Leveraging students' programming laboratory work as worked examples', *SIGCSE Bull.* **40**(2), 69–73.
URL: *http://doi.acm.org/10.1145/1383602.1383638*

Margulieux, L. E., Guzdial, M. & Catrambone, R. (2012), Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications, *in* 'Proceedings of the ninth annual international conference on International computing education research', ICER '12, ACM, New York, NY, USA, pp. 71–78.

Mason, R. & Cooper, G. (2012), Why the bottom 10% just can't do it – mental effort measures and implication for introductory programming courses, *in* M. de Raadt & A. Carbone, eds, 'Australasian Computing Education Conference (ACE2012)', Vol. 123 of *CRPIT*, ACS, Melbourne, Australia, pp. 187–196.

Merrinboer, J. J. V. & Paas, F. G. (1990), 'Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice', *Computers in Human Behavior* **6**(3), 273 – 289.

Miller, G. (1956), 'The magical number seven, plus or minus two: Some limits on our capacity for processing information.', *Psychological Review* **63**, 81–97.

Moreno, R., Reisslein, M. & Delgoda, G. (2006), 'Toward a fundamental understanding of worked example instruction: Impact of means-ends practice, backward/-forward fading, and adaptivity', *Frontiers in Education, Annual* **0**, 5–10.

Mwangi, W. & Sweller, J. (1998), 'Learning to solve compare word problems: The effect of example format and generating self-explanations', *Cognition and Instruction* **16**(2), pp. 173–199.

Paas, F., Renkl, A. & Sweller, J. (2004), 'Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture', *Instructional Science* **32**, 1–8.

Renkl, A., Atkinson, R. K. & Grosse, C. S. (2004), 'How fading worked solution steps works - a cognitive load perspective', *Instructional Science* **32**(1-2), 59–82.

Renkl, A., Atkinson, R. K., Maier, U. H. & Staley, R. (2002), 'From example study to problem solving: Smooth transitions help learning', *The Journal of Experimental Education* **70**(4), pp. 293–315.

Shulman, L. S. (2005), 'Signature pedagogies in the professions', *Daedalus* **134**(3), pp. 52–59.

Stark, R., Mandl, H., Gruber, H. & Renkl, A. (2002), 'Conditions and effects of example elaboration', *Learning and Instruction* **12**(1), 39 – 60.

Sweller, J. (1988), 'Cognitive load during problem solving: effects on learning', *Cognitive Science* pp. 257–285.

Sweller, J. & Chandler, P. (1994), 'Why some material is difficult to learn', *Cognition and Instruction* **12**(3), 185–233.

Sweller, J. & Cooper, G. A. (1985), 'The use of worked examples as a substitute for problem solving in learning algebra', *Cognition and Instruction* **2**(1), pp. 59–89.
URL: *http://www.jstor.org/stable/3233555*

van Gog, T., Kester, L. & Paas, F. (2011), 'Effects of worked examples, example-problem, and problem-example pairs on novices learning', *Contemporary Educational Psychology* **36**(3), 212 – 218.

van Gog, T., Paas, F. & van Merrinboer, J. J. (2006), 'Effects of process-oriented worked examples on troubleshooting transfer performance', *Learning and Instruction* **16**(2), 154 – 164.