Background

Worked Examples and Learning

Clark defines a worked example as “a step-by-step demonstration of how to perform a task or how to solve a problem" (Clark, Nguyen, Sweller, 2006, p. 190). Another definition for worked examples is given by Atkinson as “ instructional devices that provide an expert's problem solution for a learner to study.”(Learning from Examples: Instructional Principles from the Worked Examples Research). An effective worked example consists of a problem description, steps towards the solution and instructions at each step representing an expert’s process of thinking (Renkl, 2005). Of key importance is the step-by-step guidance for reaching the solution. It encourages the learner to form their own explanation for the undertaken step (Renkl et al, 2004) as well as think about what might follow next before they proceed. Due to the fact that worked examples present a step-by-step guidance towards reaching a solution, they are only useful for areas like programming where solving of the problem is done via applying algorithms (Renkl, 2005). In essence, worked examples help novices to build an understanding of a concept so that in later stages they are able to effectively apply this understanding to solve other problems related to this concept.

The common assumption that the best learning is by practicing solving problems is not necessarily true for learning Computing Science. Renkl(2005) argues that without being exposed to worked examples first, novices have a very restricted knowledge on the domain to be able to effectively reach a solution. Solving problems involves a lot of working memory resources. However, the memory capacity of beginners should be used for building new knowledge. Clark argues that solving practice problems leads to using too much memory capacity thus not leaving enough of it for learning new knowledge (e-Learning and the Science of Instruction: Proven Guidelines for Consumers – Ruth C. Clark, Richard E. Mayer p.204).

Studying worked examples “is one of the earliest and probably the best known cognitive load reducing techniques” (Paas et al., 2003). It has proven to be effective in learning how to solve problems (van Merriënboer, 1997). While worked examples reduce the cognitive load, they also? provide a better understanding of the concepts under consideration. This builds up the necessary expertise required to solve a particular type of problem effectively.

Worked Examples and Computing Science

Professor Quintin Cutts, who is the supervisor of this projects, has been  working with the computing science school teacher community across Scotland for ten years and has recently been awarded the MBE for services to computing science (<http://www.gla.ac.uk/news/headline_385680_en.html>). Having conversations with teachers throughout the years, he has identified that the traditional methods of teaching Computing Science in schools across the UK do not include the best proven method to learn a cognitive skill described above. Often in schools, Computing Science concepts are introduced by explaining what the concept is, followed by a simple example. Then students are presented with a problem to solve themselves. The importance of worked examples has not influenced the teaching methods. Keeping in mind that teachers are often limited time and money-wise, a possible reason for this is because there is no easy means of finding and adapting existing worked examples to the specific needs of a teacher.

As part of his research project, a Glasgow University PhD student in Computing Science, Yulun Song, developed software to facilitate the creation and viewing of worked examples. The thesis statement for the proposed research outlines the basic aims for the project. The system developed is such that it:

* “delivers usable, best practice interactive worked examples to students in a computing science context;”
* “enables teachers to create such interactive worked examples without bespoke programming, and to evolve them on the basis of feedback from the students.”

A particular class of problems in Computer science are of interest to Dr. Song – these are *transformation-based* in nature. An example described in the research thesis is building a database system from a specific set of requirements expressed in the form of a problem description in human language. The text describing the problem needs to be transformed into a graphical representation of the same problem - an ER diagram, which is then translated into a machine language such as SQL. Judgement and decision-making play a huge role in solving such a problem. However, these only come with experience and in order to gain such experience Reed & Bolstad (1991) claims that one example might be insufficient.

One can argue that worked examples can be found in many books and lectures so at first it may seem questionable what value would software bring to the existing provision. The thesis, however, raises some strong arguments to be taken into consideration. Firstly, the worked examples in books or lectures are not interactive enough. The university context is the more interactive option. Yet many students may not exploit the opportunity to raise a question due to shyness. Even if some interaction happens, this is not recorded or captured as part of the teaching process so the students cannot go back and review it. Secondly, the worked examples present may not fit well enough to the teaching needs. Last but not least, these worked examples provide little or no feedback on how they were used. For example, the only available information for a book would be the number of copies sold. This would not provide any insight on the value the examples brought to the reader.

The piece of software proposed and developed as part of Dr Song's work aims to address all the issues mentioned above. The student becomes actively involved with the material since revealing the steps required to reach a solution is under their control. The entire problem solving process can be fully captured and the students can easily go and review parts causing confusion. Complete explanation of every step is provided, enabling the student to follow the process of thinking of an expert. Revealing the solution step by step encourages thinking about the next logical step and guides the student towards the correct direction of thinking before they get confused. Furthermore, usage data can easily be captured to give feedback on how these worked examples were used. Data intended to be collected includes time spent at each step and answers to any questions present in the examples. Such information can be beneficial to two groups of people. The first group is the author of such examples. If the majority of the students spend more time at a step than expected this may be an indication that the step is not explained clearly enough. The same kind of information may bring to a teacher the knowledge that their students do not understand the material for this step well enough and they might need to revisit it.

The prototype that was developed as part of Dr. Song’s research is called *Interactive Worked Examples (IWE)*. It is in the form of a Java standalone application. There are two well distinguished groups of users - teachers and students. The flow of interaction of these groups with the system, as presented in the thesis, is shown on Figure 1. The original figure can be found in Song’s thesis as Figure 2.6 (p.57).

On the teacher side, the prototype must provide an easy way to create and modify worked examples without the need for any programmatic effort whatsoever. It stores the created worked examples in XML documents. There are four types of documents containing different aspects of the worked examples. The first type contains the document representing the problem solution. This document is split into fragments defined by the creator of the example. The second XML document defines the steps for each worked example. Every step consists of actions to be performed to some fragments of the problem solution and an explanation why these actions were undertaken. In a step, specified fragments are shown, hidden or highlighted. Some of the steps may ask the student a question. The third type of XML document stores layout information about the worked examples. Such information includes the number of panels needed for each worked example, their order and which problem solution is associated with each panel. The final XML document stores stylistic information about the worked examples. Such information includes the font style, the font size, whether the fragment should be bold, italic, etc.

The creators of the examples don’t need to be aware of the structures that store their examples since a graphical user interface is provided to facilitate the creation and the modification of such examples. A screenshot of this interface can be seen on Figure 2.

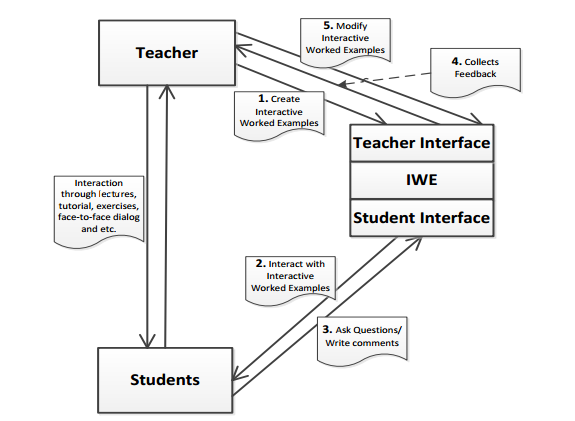
On the student side, the primary aim is the provision of a good and easy to use worked examples viewer. The primary characteristics are the step transition ability, a dedicated area for the expert’s explanation of the current step’s actions and the ability the students to be asked questions by the system. To draw the learner’s attention to the important point in each step, relevant parts of the solution are shown, hidden or highlighted as appropriate. Another feature of IWE is the ability to record data such as time spent at each step and answers to questions.

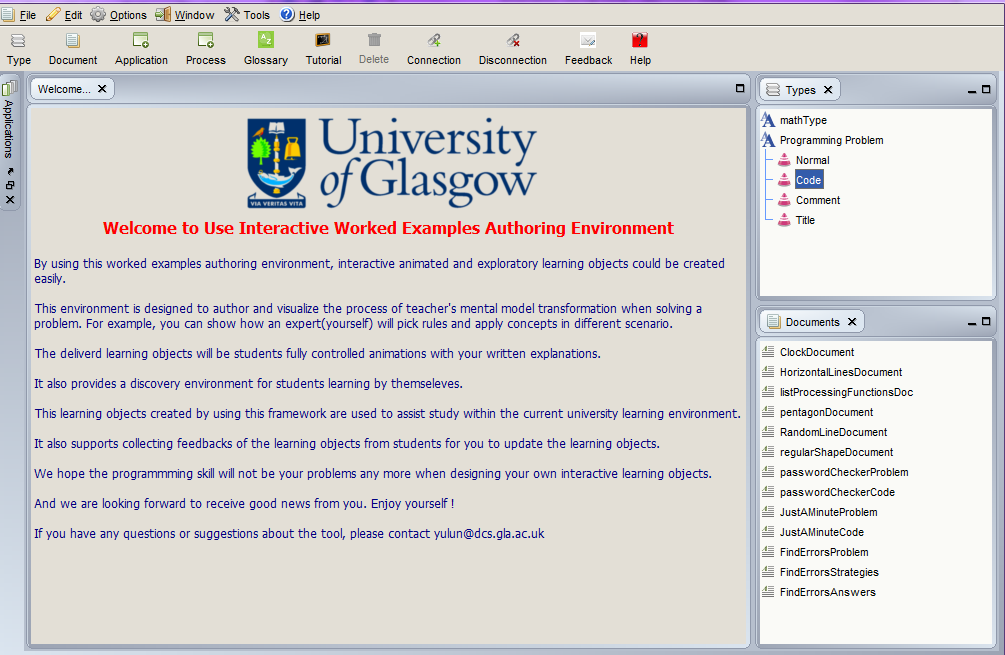
Since the nature of the examples is transformation-based often more than one area containing parts of the problem and its solution are needed. For instance, the database example described earlier would require 3 problem areas. The problem description would be placed in one part of the screen. This description needs to be translated first into an ER diagram and then into SQL statements. In order for the learning experience to be optimal all 3 areas need to be visible to the student at each step. This is the motivation for IWE to support multiple panels.

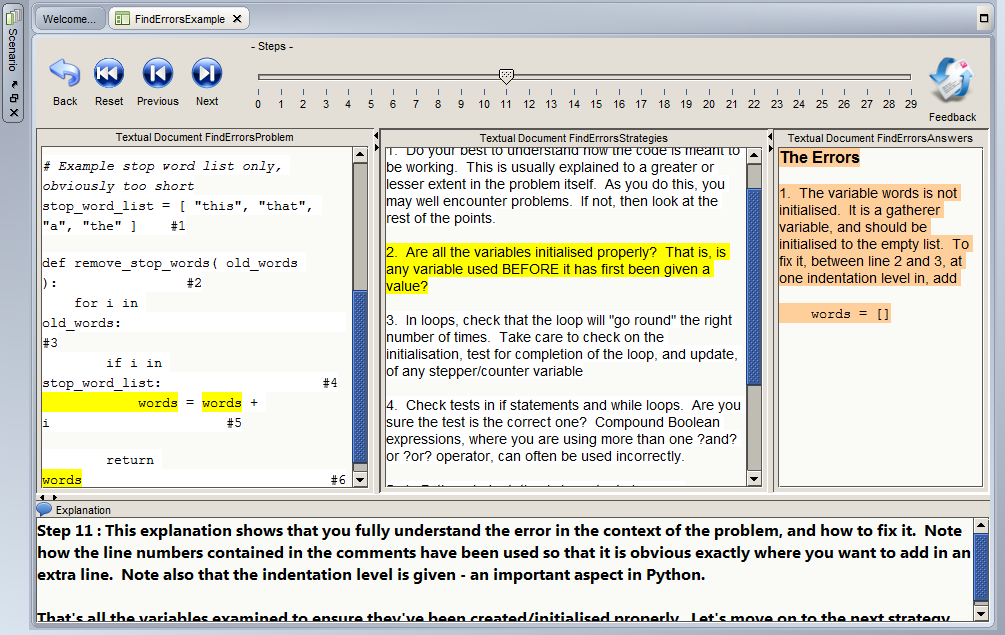
Figure 3 shows how the required features for the student interface were incorporated in the tool.

The evaluation on IWE conducted as part of Dr. Song’s research clearly shows the benefits of the tool. It has proven to be well accepted and valuable as a technique to enhance a student’s learning experience. It also achieved its goal to enable teachers to more easily and quickly develop worked examples to fit their needs. The aim of this prototype, however, doesn’t cover deployment of the software in educational institutes but rather it proves that it would bring benefit to both students and teachers.

The motivation for the current project is to make use of the findings in Dr. Song’s research and take a step forward in deploying the software in schools across the UK. For the rest of the dissertation, the reader’s attention will be drawn to the transition process between the Java standalone application IWE into a deployable online version called *Worked Examples Viewer (WEAVE)*.

Figure 1.

 Figure 2.

 Figure 3.