**Chapter 1**

**Introduction**

Teaching programming is inherently difficult. Literature on learning suggests that the most efficient way to learn a skill is by practice (reference). The way schools in the UK incorporate this approach is by explaining the basic concepts related to a specific topic, presenting simple examples to illustrate how these concepts can be applied and posing a more complex problem for pupils to solve. However, in the initial stages of becoming programmers, often beginners lack a good enough understanding of the domain to be able to solve the problem. This leads to pupils struggling to find a good solution, rather than gaining a better understanding of the problem-solving process.

A good way of teaching somebody an intellectual activity is by showing them the process of thinking involved in carrying it out This is a form of apprenticeship known in the literature as "cognitive apprenticeship"(Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18, 32-42). In any apprenticeship model, the learner needs to see many examples of the activity to be learned in order to develop the experience necessary to attempt a new, related, activity. Unfortunately, due to limited number of hours dedicated to each individual subject in schools, teachers are somewhat restricted to using only the traditional methods of teaching. Time simply wouldn’t allow them to show their students many examples of what cognitive steps they should undertake in order to solve a problem.

Research has shown that step-by-step guidance of the process of solving particular problems can help beginners gain a better understanding of the problem-solving process generally. Books provide such guidance in the form of worked examples that have proven to be effective. However, such books may not necessarily accommodate the needs of a particular teacher. Furthermore, finding a close enough example for a particular topic may become a time-consuming and discouraging activity for a teacher.

Having this in mind, a Glasgow University PhD student, Yulun Song, has developed a Java standalone application called *Interactive Worked Examples (IWE).* It aims to address the issues mentioned above as well as to evaluate to what extent such an application will prove effective in lowering the learning curve for pupils in Computing Science. It consists of two interfaces: one for students and one for creators of worked examples (who are typically teachers as well). The author interface enables the creation of examples to accommodate a teacher’s specific needs. The student interface provides users with a selection of examples to work on.

The application has proven to be effective at enhancing the teaching of Computing Science in university. Since the research questions around IWE were to explore the extent to which it can fit in the teaching process in a university context and whether it would be a potentially successful learning technique, the prototype does not aim at large scale deployment. A sensible next step is to put the system into use in schools, where support for computing education is urgently needed. However, many issues in deploying IWE arise because of it being in the form of a Java standalone application. In schools in the UK there tends to be a blanket policy about the systems provision on any subject. In order to install a program on a school machine, a request to the service provider responsible for the particular school needs to be made. The service provider will then need to analyse the risk that installing new program will pose to the whole system and submit a further request to a local authority responsible for the particular school. This overhead would be enough to prevent most teachers from considering adoption, both from a time and cost standpoint.

The issue of software provisioning in schools gives the major motivation for this project to recreate IWE as a web-based application in order to start effectively presenting worked examples in a larger context. This will avoid the complicated and time-consuming process of installing IWE in schools. Furthermore, schools will be able to receive the latest updates of the application and its worked examples with no effort. Ultimately, a web-based system could share worked examples developed nationally and even internationally. The web-based version of IWE is called *Worked Examples Viewer (WEAVE)*.

In addition to being a more easily deployable version of IWE, WEAVE takes a step further to move from author-student to author-student-teacher target user groups. This brings in interesting new aspects. Teachers will be able to see personalised information about how their students interact with the examples. Authors, on the other hand, will receive information about the general usage of these examples, rather than personalised one.

Another benefit of WEAVE being web-based is that the worked examples in the system will not be limited to the ones created by one teacher or a group of teachers only. Instead, examples created by any teacher will immediately be available to everyone. This would contribute to a collaborative way of developing such examples and would give the chance for students to undertake further learning if they desired so. Furthermore, teachers would be able to benefit from their colleagues’ expertise as well as get ideas and adjust them to their specific needs with less effort than creating new examples from scratch. Ideally, such a system can be revolutionary in improving the teaching practices in schools, help teachers understand the difficulties of their students and enable them to help each other to become better in teaching Computing Science.

The rest of this dissertation describes more background for the motivation of the project, the requirements for, as well as the design and the implementation of WEAVE together with the testing methods that were used to ensure that the application works as intended. An evaluation chapter follows making conclusions about how easy and effective it is to integrate WEAVE successfully integration in everyday teaching practice. The final chapter is dedicated to the future developments for the system which will be addressed shortly.

**Chapter 2**

**Background**

**2.1. Worked examples**

**2.1.1. Definition of a Worked Example**

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. 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.

**2.1.2. Worked Examples and Learning**

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.

**2.1.3. Worked Examples in Computing Science Context**

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 jump to problem solving is too quick and 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, the former Glasgow University PhD student Dr. Yulun Song developed software to facilitate the creation and viewing of worked examples. The thesis statement for the 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.”

Dr. Song was particularly interested in Computing Science problems due to their transformation-based nature. They involve the analysis and the transformation of one representation of the problem, such as text definition or a diagram, into another representation, i.e. the solution. 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) claim that one example- which is the typical case in schools- might be insufficient. In his research, Dr. Song argues that a system that enables the user to view multiple worked examples would prove efficient in such a context. He therefore developed a tool for the provision of worked examples in Computing Science so that the user is exposed to more than one of those examples.

**2.1.4. Problems with existing methods for delivering worked examples**

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.

* The worked examples in books or lectures are not interactive enough. The readers of books or the attendees of a lecture are presented with some examples, but often the process of thinking why a particular action is undertaken or is a better option for reaching a solution remains unexplained. One can argue that the university context has some grounds for interactivity or discussions. Yet many students may not exploit this due to shyness or simply because they might not know what questions to ask. 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.
* The worked examples present may not fit well enough to the teaching needs. Books aim to target a large portion of potential readers so they need to be general enough to fit every reader’s needs. However, this means that one particular reader may need to adjust their studying or teaching around this general example. What would be more beneficial - and Dr. Song aims to address - is to adjust the worked examples depending on the teaching or learning needed.
* Worked examples in books provide little or no feedback on how they were used to the author or to teachers who benefit from using such examples in their teaching. 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. What is desired is information about how a particular worked example was used, were there any problematic areas and how the readers benefited from it. Such information would allow the authors to improve their future work at constructing worked examples. In addition, this information could be beneficial to teachers or lecturers who could use them for assessing what parts of the example were problematic and adapting their teaching accordingly.

**2.1.5. How does a computer-based application solve the problems with the traditional methods of delivering worked examples?**

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. In addition, 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:

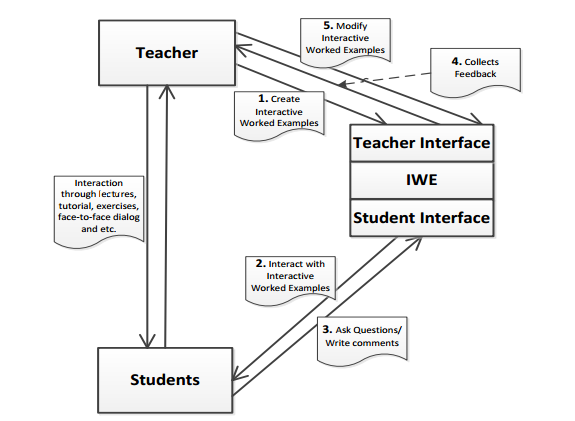
* Authors of worked examples. They could benefit from the knowledge of the time the majority of the students spend at each step. If this time exceeds dramatically the expected time for this step, this can be an indication that the step is unclear and brings confusion. Knowing this, authors could update the example by including a better explanation or by breaking this step into more than one steps and examine the effect this has. This way authors will learn how to build their examples and this will also bring benefits to the reader in terms of provision of improved worked examples.
* Teachers. The knowledge that their students visit a step multiple times or spend too long before proceeding would indicate to the teacher that their students do not understand the material for this step well enough and they might need to revisit it in class.

**2.2. Interactive Worked Examples Tool**

Dr. Song’s research product is called *Interactive Worked Examples (IWE)*. It is in the form of a Java standalone application. The following sections provide more information about the tool.

**2.2.1. Intended Flow of Interaction**

There are two well-distinguished groups of users – authors of examples and students. Each group is serviced by a separate interface of the application. 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).

Figure 1.

Authors can create and modify interactive worked examples through the author interface. This interface also presents them with any student feedback on these examples.

The student interface serves as a worked examples viewer where students are given the opportunity to ask questions and write comments.

The interaction between teachers and students is direct rather than through the system.

**2.2.2. Structure of IWE**

IWE stores the worked examples in XML files. There are three types of files that are of particular interest for this Level 4 project- Documents.xml, Applications.xml, and Processes.xml. The structure of those files is graphically represented on Figure 2.

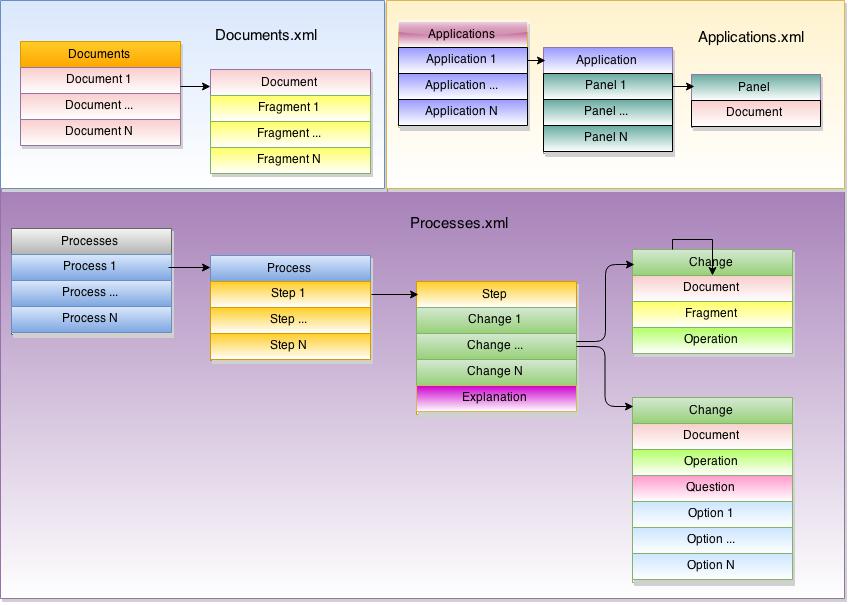


Figure 2.

The Documents.xml file stores the collection of documents created by an author. A document is one of the representations involved in a particular worked example- perhaps it is the problem specification, or an intermediate solution, or the solution. It is split into fragments which are small logically separated portions of the document. The reason for splitting the document into fragments is so that the document can be revealed gradually, to show the step-by-step problem solving process. Individual fragments can also be highlighted to be brought to the viewer’s attention.

The Applications.xml file stores layout information about worked examples, bringing together the particular documents involved in the worked example. The way the documents are laid out visually, in panels, is recorded.

The Processes.xml file defines the steps for the worked examples. For each step there are a number of changes and an explanation of those changes. There are two types of changes. The first type specifies which fragment of a document is involved in this change. These fragments can be shown, hidden or highlighted depending on the effect the author is aiming to achieve. The second type of changes corresponds to a question and possibly a set of options the user can select from in an attempt to answer it.

There is one more type of XML file which is not shown on Figure 2. It contains information about different styles that can be used for the worked examples- there is a similar, although simpler, version of the style mechanism found, for example, in word processors or CSS style sheets. There is no need this file to be discussed in any detail. However, the reader needs to know that documents have styles associated with them depending on the type of document, enabling different fragments within a document to be shown in different typographical styles.

**2.2.3. Main Features of IWE’s Student Interface**

The student interface aims to provide an effective worked examples viewer. A screenshot of the final version of Dr. Song’s prototype can be seen on Figure 3. The most important characteristics are labelled with numbers and are detailed below.

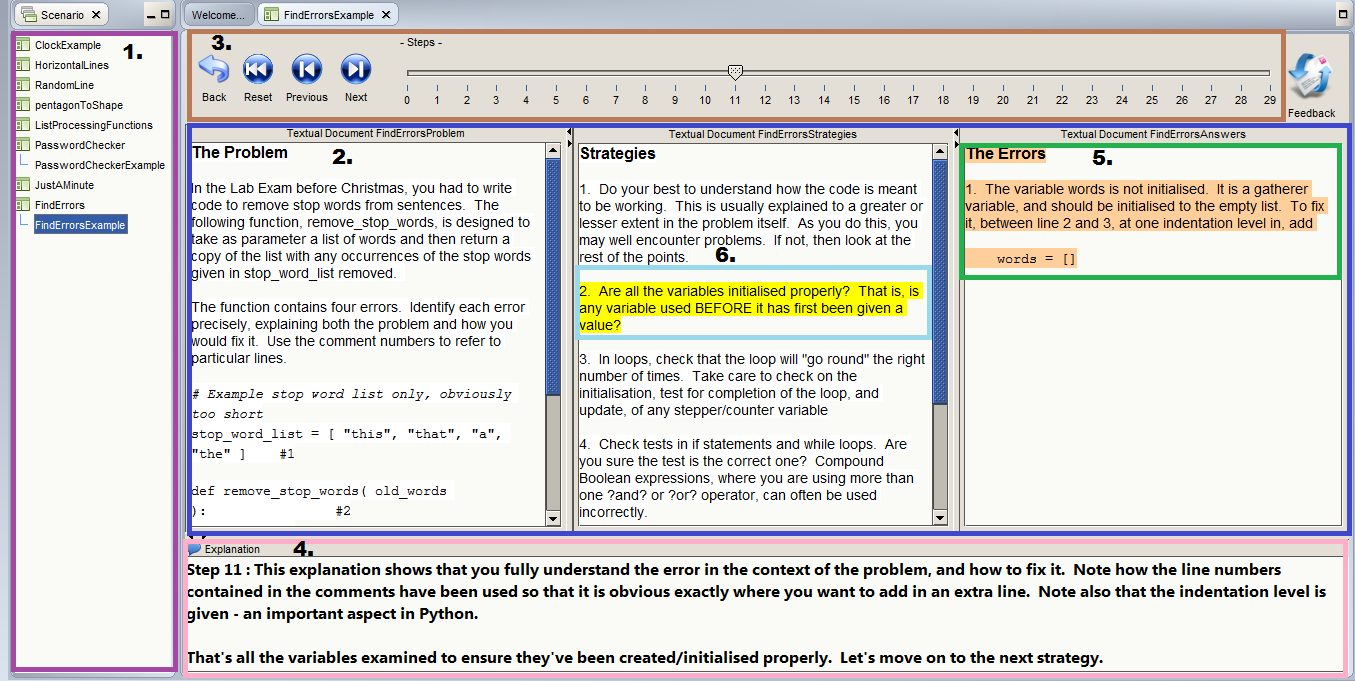


Figure 3.

1. An area for showing the worked examples installed on the system and enabling the user to choose an example to work on.

2. Panels showing different documents for a particular worked example.

3. An area for controlling transitions between steps.

4. An explanation area where the expert’s process of thinking involved on the current step is shown.

5. Highlighting of the newly appeared text at a particular step for drawing the user’s attention to the new content relevant for the current step.

6. Highlighting of fragments of interest for a particular step.

As the student uses the controls in area (3) to move through the worked example, the contents of the documents panels and the explanation area change to reveal the developing solution and the thinking process behind it.

Other features of IWE, which are not shown on the screenshot, are the ability of the tool to ask the user questions and to record data such as time spent at each step and answers to questions.

**2.2.4. Relation of IWE to this Level 4 project**

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 has demonstrated that it would bring benefit to both students and teachers.

The motivation for this Level 4 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. The focus of the project will be reconstructing the student interface of IWE into a web-based form as well as providing a new interface for teachers to separate them as a separate group of users. The authoring interface of IWE is beyond the scope of this project and no understanding of any of its aspects is needed by the reader to follow this dissertation.

For the rest of the dissertation, the reader’s attention will be drawn to the transition process of the Java standalone application IWE into a deployable online version called *Worked Examples Viewer (WEAVE)*.