\pdfoutput=1

\documentclass{l4proj}

%\graphicspath{{images}}

%

% put any packages here

%

\begin{document}

\title{How to Produce a Level 4 Project Report}

\author{Patrick Prosser}

\date{October 18, 2012}

\maketitle

\begin{abstract}

We show how to produce a level 4 project report using latex and pdflatex using the

style file l4proj.cls

\end{abstract}

\educationalconsent

%

%NOTE: if you include the educationalconsent (above) and your project is graded an A then

% it may be entered in the CS Hall of Fame

%

\tableofcontents

%==============================================================================

\chapter{Introduction}

\pagenumbering{arabic}

Teaching programming is inherently difficult. The way it is taught in schools in the UK 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, beginners often lack a good enough understanding of the domain from just the simple examples to be able to solve the problem~\cite{cooper2008, renkl2005}. This can lead to pupils struggling to find a 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"~\cite{cognitive\_apprenticeship}. 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 would not allow them personally to show their pupils many examples of what cognitive steps they should undertake in order to solve a problem.

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. These examples have proven to be effective ~\cite{sweller1985, tarmizi1988, ward1990, zhu1987}. It has also been found that they reduce the cognitive load when acquiring a skill~\cite{sweller98}. 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 \textit{Interactive Worked Examples (IWE)}~\cite{song-thesis}. 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 students 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~\cite{song-thesis}. 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~\cite{royalsoc2012}. 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 a 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.

This 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 \textit{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 pupils 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 pupils 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 pupils and enable them to help each other to become better in teaching Computing Science.

The rest of this dissertation describes more background for the context 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 in everyday teaching practice. The final chapter is dedicated to the future developments for the system which will be addressed shortly.

\chapter{Background}\label{background}

\section{Worked Examples}

\subsection{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"~\cite{clark2006}. Another definition for worked examples is given by Atkinson as “instructional devices that provide an expert's problem solution for a learner to study.”~\cite{atkinson2000}. 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~\cite{renkl2005}. 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~\cite{renkl2004} 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 will be able to effectively apply this understanding to solve other problems related to this concept.

\subsection{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~\cite{renkl2005} 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(reference?). However, the memory capacity of beginners should be used for building new knowledge instead. 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.

\subsection{Worked Examples in a School 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 pupils 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 Yulun Song developed software to facilitate the creation and viewing of worked examples~\cite{song-thesis}. The thesis statement for the research outlines the basic aims for the project. The system developed is such that it:

\begin{itemize}

\item{``delivers usable, best practice interactive worked examples to students in a computing science context;''}

\item{``enables teachers to create such interactive worked examples without bespoke programming, and to evolve them on the basis of feedback from the students.''}

\end{itemize}

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 and Bolstad (1991)(reference) claim that one example- which is the typical case in schools- might be insufficient. In his research, 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.

\subsection{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. Song's thesis~\cite{song-thesis}, however, raises some strong arguments to be taken into consideration, as shown below.

\begin{itemize}

\item 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.

\item 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 Song aims to address - is to adjust the worked examples depending on the teaching or learning needed.

\item 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.

\end{itemize}

\subsection{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 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:

\begin{itemize}

\item 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.

\item 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.

\end{itemize}

\section{Interactive Worked Examples Tool}

The following sections provide more information about the IWE tool.

\subsection{Intended Flow of Interaction}

There are two well-distinguished groups of users of the IWE tool – 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 Song's thesis, is shown on Figure 2.1. The original figure can be found in Song's thesis as Figure 2.6 (p.57). The following bracketed numbers correspond to the associated numbers in the Figure.

%\vspace{-7mm}

\begin{figure}

\centering

\includegraphics{images/IWE\_flow\_of\_interaction.png}

%\vspace{-30mm}

\caption{IWE Flow of Interaction.}

\label{iwe\_flow\_of\_interaction}

\end{figure}

\begin{itemize}

\item{Authors can create (1) and modify (5) interactive worked examples through the author interface. This interface also presents them with any student feedback (4) on these examples.}

\item{The student interface serves as a worked examples viewer (2) where students are given the opportunity to ask questions and write comments (3).}

\item{The interaction between teachers and students is direct rather than through the system.}

\end{itemize}

\subsection{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- \texttt{Documents.xml}, \texttt{Applications.xml}, and \texttt{Processes.xml}. The structure of those files is graphically represented on Figure \ref{xml\_files}. A description of each file can be found below.

%\vspace{-7mm}

\begin{figure}[t!]

\centering

\includegraphics[width=\textwidth]{images/xml\_files\_structure.jpg}

%\vspace{-30mm}

\caption{Structure of the XML files storing the worked examples created in IWE.}

\label{xml\_files}

\end{figure}

\begin{itemize}

\item The \texttt{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.

\item The \texttt{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.

\item The \texttt{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.

\end{itemize}

There is one more type of XML file which is not shown on Figure \ref{xml\_files}. 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 for 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.

\subsection{Main Features of the IWE Student Interface}

The student interface aims to provide an effective worked examples viewer. A screenshot of the final version of Song's prototype can be seen on Figure \ref{iwe\_student\_interface}. The most important characteristics are labeled with numbers and are detailed below.

%\vspace{-7mm}

\begin{figure}

\centering

\includegraphics{images/iwe\_student\_interface.jpg}

%\vspace{-30mm}

\caption{A screenshot of the student interface of IWE.}

\label{iwe\_student\_interface}

\end{figure}

\begin{enumerate}

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

\item Panels showing different documents for a particular worked example.

\item An area for controlling transitions between steps.

\item An explanation area where the expert's process of thinking involved on the current step is shown.

\item 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.

\item Highlighting of fragments of interest for a particular step.

\end{enumerate}

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 the IWE student interface, 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.

\subsection{Relation of IWE to this Level 4 Project}

The evaluation on IWE conducted as part of 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 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 distinct group of users. The author 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).

\chapter{Requirements}

This chapter provides a detailed description of the functional and non-functional requirements for WEAVE.

\section{Preface}

As described in Chapter~\ref{background}, this Level 4 project builds upon an existing system for facilitating the use of worked examples in educational context. The evaluation of IWE clearly shows that such software would be a valuable asset contributing to the learning process of students. Due to the overly complicated procedure required to deploy IWE in schools while it is in the form of a Java standalone application, the need to turn it into a more easily deployable online version arose. Interviews with highly motivated teachers, who are part of PLAN C project\footnote{\texttt{http://www.cas.scot/plan-c/}. Last accessed March 24\textsuperscript{th}, 2015.}, have identified the need for one more interface to be used in schools. In order to improve their teaching practice and to be able to provide high quality feedback to their pupils, these teachers would benefit from knowing how pupils in their classes use these worked examples. Information that would be valuable for them includes:

\begin{itemize}

\item identification of which pupils interacted with which examples;

\item aggregated information on answers selected for multiple choice questions and the pupils that selected each answer;

\item information about the average time spent at each step of an example as per the whole class (or an individual); and

\item summary data of the total time spent at an example and the last step reached by each pupil in the class.

\end{itemize}

This project aims to achieve four goals.

\begin{enumerate}

\item[G1.] Build a web-based viewing system that is interoperable with the author interface of IWE, i.e. ensure that worked examples created using the old system can be viewed in the new system.

\item[G2.] Provide an interface for teachers that will help them gain more information on how the worked examples are used by their own pupils.

\item[G3.] Replicate as closely as possible the student interface of IWE.

\item[G4.] Ensure that worked example authors can view usage data in an anonymous manner, such that individual pupils, classes or schools are not identifiable.

\end{enumerate}

\section{Classification of Requirements}

The requirements are classified according to the \textit{MoSCoW}~\cite{jonasson2012} classification method. The categories considered are:

\begin{itemize}

\item \textbf{must-have}- requirements that are crucial for the achievement of the goal of this project and must be implemented

\item \textbf{should-have}- requirements that are considered to be important but not necessarily crucial for achieving the goal of this project and should be implemented

\item \textbf{could have}- requirements that have been identified as features that would add further value to the prototype but are thought of as stand-out ones rather than ones contributing to the correct functioning of the prototype and may not be implemented due to constraints

\end{itemize}

The \textbf{would-like} category coming from the \textbf{W} in \textit{MoSCoW} is not part of the classification methods used for this project due to the fact that all the requirements fit comfortably in the other categories.

\section{Functional Requirements}

\subsection{Interoperability with the Existing Author Interface}

The existing system uses XML files to store the worked examples. The structure of these files is shown on Figure \ref{xml\_files} in Chapter~\ref{background}. The web-based system will need to read in worked examples from these data files. Furthermore, feedback from pupils and teachers will inevitably lead to changes being required in some of the worked examples. The existing authoring tool supports editing of worked examples, and it is expected that it will still be used to make such changes. WEAVE will need to be able to support these changes. Due to the fact that the update model of IWE is destructive- no versioning of the examples is supported- and that WEAVE does not provide means for modifying examples, the update model will follow the one of the old system.

The prototype:

\begin{itemize}

\item \textbf{must} be able to parse an XML file containing the fragmented problem specifications of the worked examples and their solutions.

\item \textbf{must} be able to parse an XML document containing information about individual steps of the worked examples (e.g. which fragments of a document must be shown/hidden/highlighted, the explanation associated with a step and a question if one was provided).

\item \textbf{must} be able to parse an XML document containing information about the layout of worked examples (e.g. number of panels for the example, their order and problem solutions associated with each panel).

\item \textbf{must} be able to parse an XML document containing information about the styling associated with each example (e.g. font style, font size, etc.).

\item \textbf{must} be able to support easy addition of new worked examples created using the old authoring tool.

\item \textbf{must} be able to incorporate new versions of worked examples already installed in the web-based system.

\end{itemize}

\subsection{Teacher Interface Requirements}

A major part of the contribution of WEAVE is to enable teachers to receive information about how their pupils worked with these examples, while authors of such examples and Computing Science researchers must receive such data in an anonymised way. The desired effect is for teachers to be able to see usage data for their classes as well as individuals in these classes. However, protecting the privacy of both teachers and pupils is a major issue. The authors of worked examples will be able to see any usage data for the examples they created. If this data is informative enough for them to identify the person standing behind this data, this would be highly unethical and would violate privacy.

In this section, the requirements for the teacher interface are outlined. The next chapter will describe how the privacy issues mentioned above are resolved by the system and will discuss in further detail these requirements.

The teacher:

\begin{itemize}

\item \textbf{must} be able to choose whether to track their pupils' progress or not.

\item \textbf{must} be able to view how the students in a particular class interacted with the examples.

\item \textbf{must} be able to view how a particular student interacted with the examples.

\end{itemize}

\subsection{Replication of the IWE Student Interface}

One of the goals for this Level 4 project (G3) is to replicate the student interface of IWE as closely as possible. The reasoning behind the requirements for the student interface, as well as the positive conclusion from their evaluation, are described in detail in Song's thesis. These were found acceptable for the purpose of this project. In short, the research literature describes the qualities of successful worked examples (reference), and these qualities were included in Song's design. Their implementation in his software system was shown to be suitable for use by students in the extensive evaluation carried out.

The prototype:

\begin{itemize}

\item \textbf{must} enable the pupil to select a worked example from a list of existing examples.

\item \textbf{must} support multiple panels for the different parts of the problem solution.

\item \textbf{must} contain a dedicated area for the explanation.

\item \textbf{must} support showing/hiding/highlighting of fragments.

\item \textbf{must} support the option to ask pupils questions.

\item \textbf{must} enable the pupil to go back and forwards through steps.

\item \textbf{must} highlight the newly introduced fragments at each step.

\item \textbf{should} record time spent at a step.

\item \textbf{should} record answers to questions

\item \textbf{should} enable the pupil to reset the example there are working on.

\item \textbf{could} provide a means for drawing the pupil’s attention to the newly introduced fragments .

\end{itemize}

\subsection{Additional Features Needed for the Student Interface}

This section describes the requirements for satisfying goals G2 and G4 – to support identifiable usage information for the teacher, and anonymous usage information for worked examples authors and for Computing Science education researchers. It is important that each teacher is able to link usage data to their groups/pupils, while authors of such examples must not be able to identify by any means what the group or who the pupil is due to the privacy issues discussed previously.

The prototype:

\begin{itemize}

\item \textbf{must} be able to record usage data for an example.

\item \textbf{must} be able to record the usage data in such a way that the true identity of the pupil associated with this data is not revealed when this information is sent to authors of worked examples and Computing Science education researchers.

\item \textbf{must} allow the pupil to use the system without any identifying information.

\item \textbf{must} be able to connect the usage information stored for a pupil to their teacher and the class they belong to.

\end{itemize}

\section{Non-Functional Requirements}

The non-functional requirements for WEAVE are guided mostly by the web-based nature of the system and by the context it is intended for. In order for pupils to be able to study the worked examples effectively, and also due to the small workstation screen sizes found in schools, the area showing the worked example should be maximised. Furthermore, due to the step-by-step nature of worked examples, some steps may put more emphasis on the explanation while others might be more intensive in the problem specification areas so the system must be able to deal with such situations accordingly. In addition, features which make the interactions with the examples more convenient and which would minimise effort, such as shortcuts and appropriate fitting of the whole system on the screen, are highly desirable.

The web-based nature of WEAVE poses a possible problem when uploading modifications to existing examples because pupils might be working on these examples at the same time. Consistency must be ensured in such cases, meaning that the pupil should be able to see either the old or the new version of the example, rather than a mixture of both.

Since pupils may have not worked with such a system before, they may benefit from a brief guide on how to use WEAVE in an optimal way.

These considerations form the following requirements:

\begin{itemize}

\item The prototype \textbf{must} be easy to use.

\item The worked examples \textbf{must} fit the entire screen.

\item The size of the area showing the worked examples \textbf{must} not exceed the size of the screen.

\item A modification to a worked example \textbf{must} not affect pupils doing the same example.

\item The student interface \textbf{should} include a tutorial on how to use the system

\item The teacher interface \textbf{should} provide information on how to use each feature.

\item The panels showing the problem content \textbf{should} be resizable.

\item The explanation area \textbf{should} be resizable.

\item Shortcuts for easier transition between steps \textbf{could} be added.

\end{itemize}

The following chapter will describe the design decisions which were constructed based on these requirements.

\chapter{The Fox and Dog}

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

\section{The Fox Jumps Over}

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over Uroborus (Figure \ref{uroborus}).

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

%\vspace{-7mm}

\begin{figure}

\centering

\includegraphics[height=9.2cm,width=13.2cm]{uroboros.pdf}

\vspace{-30mm}

\caption{An alternative hierarchy of the algorithms.}

\label{uroborus}

\end{figure}

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over \cite{ckt} the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

\section{The Lazy Dog}

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox \cite{am97} jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog.

%%%%%%%%%%%%%%%%

% %

% APPENDICES %

% %

%%%%%%%%%%%%%%%%

\begin{appendices}

\chapter{Running the Programs}

An example of running from the command line is as follows:

\begin{verbatim}

> java MaxClique BBMC1 brock200\_1.clq 14400

\end{verbatim}

This will apply $BBMC$ with $style = 1$ to the first brock200 DIMACS instance allowing 14400 seconds of cpu time.

\chapter{Generating Random Graphs}

\label{sec:randomGraph}

We generate Erd\'{o}s-R\"{e}nyi random graphs $G(n,p)$ where $n$ is the number of vertices and

each edge is included in the graph with probability $p$ independent from every other edge. It produces

a random graph in DIMACS format with vertices numbered 1 to $n$ inclusive. It can be run from the command line as follows to produce

a clq file

\begin{verbatim}

> java RandomGraph 100 0.9 > 100-90-00.clq

\end{verbatim}

\end{appendices}

%%%%%%%%%%%%%%%%%%%%

% BIBLIOGRAPHY %

%%%%%%%%%%%%%%%%%%%%

\bibliographystyle{plain}

\bibliography{bib}

\end{document}