# Example Based Pedagogical Strategies in a Computer Science Intelligent Tutoring System

by

NICHOLAS GREEN
BSc (Queen Mary, University of London, United Kingdom) 2001

Thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2017

Chicago, Illinois

Defense Committee:
Prof. Barbara Di Eugenio, Chair and Advisor
Prof. Ugo Buy
Prof. Thomas Moher
Prof. Davide Fossati, Emory University
Prof. Patrick Seeling, Central Michigan University

# ACKNOWLEDGMENTS

This thesis was made possible through the ChiQat-Tutor research project - an investigation into using an ITS in Computer Science - which was jointly conducted by the University of Illinois at Chicago, and Carnegie Mellon University in Qatar.

My deepest gratitude and thanks goes to the co-PI of this project, my committee chair, advisor, and mentor, Prof. Barbara Di Eugenio. The advice given to me over many years has most certainly shaped the way I approach research, and will influence decisions I make for the rest of my life. I am truly in debt to her for training me to be a better researcher, and person.

A special thank you goes to the rest of my committee - Prof. Ugo Buy, Prof. Davide Fossati, Prof. Tom Moher, and Prof. Patrick Seeling - to help me shape this thesis to become a better piece of scholarly work. Also, Prof. Mitchell Theys, and Prof. Patrick Troy, at the University of Illinois at Chicago, for aiding the project in allowing data collection to take place in their classrooms.

Finally, I have also been able to work with some truly fantastic collaborators and students over these years: Rachel Harsley, Lin Chen, Dr Omar AlZoubi, Mehrdad Alizadeh, David Randolph, Abhinav Kumar, Sabita Acharya, Paul Landes, Itika Gupta, and Natawut Monaikul. Working with a great group of people has made the process of working on this thesis even better.

This work was primarily supported by NPRP grant 5-939-1-155 from the Qatar National Research Fund (a member of Qatar Foundation), as well as other assistantships provided by the University of Illinois at Chicago's Department of Computer Science.

NG

# CONTRIBUTION OF AUTHORS

This thesis was carried out as part of a larger project that includes work previously published to several conferences. Much of this also includes contributions from multiple authors.

Chapter 3 includes work from (Di Eugenio et al., 2013), the copyright for this publication is given in Appendix F. I was third author is this publication, where I worked equally with Lin Chen in annotating a tutoring corpus and performing statistical analysis on the results.

Chapter 4 is largely based on (Green et al., 2015). Authorisation on using this has been given by the publisher (Appendix G). I was first author in this publication where I designed and developed the covered system. Feedback on design decisions was given by Mehrdad Alizadeh and Omar AlZoubi.

Chapter 5 includes work from both (Green et al., 2015) and (Green et al., 2016) - copyright for each publication is given in Appendix H and Appendix I. I was first author on both publications where I lead experimentation, grading, and analysis of results. Rachel Harsley, Sabita Acharya, Mehrdad Alizadeh, and Itika Gupta were also involved in data collection, as well as Rachel Harsley being involved with grading of experiment pre/post tests.

# TABLE OF CONTENTS

CHAPTER PAGE

## TABLE OF CONTENTS (Continued)

## TABLE OF CONTENTS (Continued)

# LIST OF TABLES

## LIST OF TABLES (Continued)

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| CS | Computer Science |
| ITS | Intelligent Tutoring System |
| WOE | Worked-out Example |
| CLT | Cognitive Load Theory |
| ANOVA | Analysis of Variance |
| UIC | University of Illinois at Chicago |
| ChiQat | ChiQat-Tutor |

# SUMMARY

Worked-out examples are a common teaching strategy that aids learners in understanding concepts by use of step-by-step instruction. Literature has shown that they can be extremely beneficial, with a large body of material showing they can provide benefits over regular problem solving alone.

This research looks into the viability of using this teaching strategy in an intelligent tutoring system specificity designed for the computer science domain. Here, we detail the developed tutoring system, ChiQat-Tutor, which is designed with scalability and experimentation at its core. The system is described demonstrating its powerful architecture that gives the flexibility to analyse different teaching strategies.

From the developed tutoring system, we focus on investigating the value in using the worked-out example teaching strategy in the system. Our investigation looks at human-human tutorial dialogues, and an implemented worked-out example module in ChiQat-Tutor for the linked list lesson. Multiple version of the worked-out example strategy is evaluated, with log data collected for each experimentation session that chronicles user behaviour. Various worked-out example based features are then identified that may be of used in such a system that could enhance student learning. We present a pipeline that may increase the effectiveness of an ITS that uses some of the example based features.

# CHAPTER 1

# INTRODUCTION

One-on-one tutoring is an effective means of teaching that allows a teacher to interact directly with their student, as well as allowing them to personalise the delivery of material in any means they see fit. This thesis looks at the use of intelligent tutoring systems - a computer application that takes the role of a human tutor. Our study focuses on the use of example based tutoring within the computer science domain, where we investigate the effectiveness of such a strategy in this setting. Next, we introduce the topic further and state our goals.

## 1.1 Human and Intelligent Tutoring

An effective method of learning for many students is via individualised tutoring (Shute and Psotka, 1994; Wood et al., 1976; Cohen et al., 1982; Bloom, 1984). Tutoring can take many forms, such as in a group or on a one-on-one basis. In either case, a tutor can provide a customised teaching plan to their students. Concepts can be introduced, examples can be given, and discussions can be made between all within the group. Rather than a didactic form of teaching, students can interact with the tutor, get feedback from them in the form of corrections, alternative lines of thought, and praise.

Even though tutors can be effective in inducing learning, there is a huge disadvantage; tutors come at an increased cost and reduced availability. Large lecture and classroom settings are popular due to them being cost effective as only one teacher is required for a large number of students. Lectures can be set in lecture halls with hundreds of students, or even be broadcast online for global reach. Having the same teacher interact with individual students is not possible. Another issue is that such teachers may not even be available in areas of need, for example, in the developing world or deprived communities.

One option in providing a similar learning opportunity on a cost effective basis with broader scope is to utilise virtual tutors, such as an *intelligent tutoring system* (ITS) (Sleeman and Brown, 1982; Corbett et al., 1997; Nwana, 1990; Graesser et al., 2012). An intelligent tutoring system recreates a similar experience as that of a human tutor by teaching a student skills via appropriate pedagogical strategies. Although a human tutor is a great example of an effective tutor, it is important to note that the goal of an ITS is to transfer knowledge and skills, therefore an exact translation of a human tutor is not necessary as there may be some deficiencies in a particular tutor's delivery of material (Shute and Psotka, 1994). Developing an effective ITS is not a trivial matter as it relies on multiple disciplines, these primarily being computer science, psychology, and education.

## 1.2 Learning Computer Science

As all seasoned Computer Science (CS) professionals know, starting off in the field is not easy. There are fundamental concepts and principles that must be taught to all students, many of which feel unintuitive and require a new way of thinking. Unfortunately, it is at these early steps that students may decide to choose a different path in their career. It has been suggested that attrition rates can be around 30%-40% at many institutions, coming from a variety of populations, such as freshman, sophomore, female, and students with weak mathematical backgrounds (Cohoon, 2001; Beaubouef and Mason, 2005; Rizvi and Humphries, 2012; Porter et al., 2013). This is not just impacting the number of professionals in the computing industry, but society as a whole, since there is now a significant worldwide skills shortage (Beaubouef and Mason, 2005). These issues may be alleviated by providing higher quality and more accessible CS education in the early stages.

Beaubouef, et. al. (Beaubouef and Mason, 2005) and Porter, et. al. (Porter et al., 2013) suggest that young computer scientists usually encounter problems in common classes, such as CS1 and CS2. Generally these fundamental classes can include material on computer languages, data structures, and computational theory. It therefore appears that such classes may be a source of students dropping computer science degrees. Henceforth, alleviating issues that may arise from such classes may reduce drop out rates.

### 1.3 Learning by Example

In 1985, Sweller and Cooper (Sweller and Cooper, 1985) researched effective learning strategies for teaching algebra in mathematics. What they found is that teaching a concept via step-by-step examples can be an effective scaffold for traditional problem solving. Such an example is broken down into three stages:

- Problem formulation
- Solution steps
- Final solution

This way, a student is given a full example of working through a problem, from start to finish. The benefits to the student are rooted in *cognitive load theory* (Sweller, 2011). It is thought that by teaching via such an example a student is able to assimilate information without the cognitive overhead that problem solving alone places on the student.

During traditional problem solving, students may have multiple concurrent cognitive tasks to accomplish - they have to learn the material and apply it at the same time. This may not always be the most effective means of learning. *Worked-out examples* (WOE) are a way of offering the student help in acquiring initial schemas associated with the problem domain without initially applying such knowledge. Once these have been acquired, the student should be able to apply these fundamental concepts in solving problems later on. This breakdown has been shown to help students in the early stages of knowledge acquisition, especially when algorithmic concepts are involved.

## 1.4 Research Aims and Contributions

In this work, we aim to answer a single overarching research question - Can features of worked-out examples support learning in a computer science intelligent tutoring system? In order to answer this question, we outline four tasks:

1. Investigate the effect of worked-out example on student learning with computer science concepts.
2. If human-human worked-out examples are effective, see if it can also be effective in an intelligent tutoring system.
3. Discover if there are any properties of worked-out examples, and if so what, may be of use to learners.
4. Investigate how such features can be used to enhance a computer science intelligent tutoring system.

In answering this question, we have provided several primary contributions. Firstly, we established the usefulness of worked-out examples in a computer science ITS in Chapter 3. For this, we extended upon previous work in ITS development (Fossati, 2008; Fossati et al., 2009; Fossati et al., 2010; Fossati et al., 2015) by analysing an already collected corpus of one-on-one human to human tutorial sessions that covers topics such as binary search trees, linked lists, and stacks. From this, we gained further insight into the usefulness of the WOE strategy. It had been observed that examples had been used fairly often in the tutorial dialogues, with indications that they can increase learning gains.

In order to perform experimentation, we developed a sophisticated ITS called ChiQat-Tutor, which we briefly cover in Chapter 4. ChiQat-Tutor provides a modular ITS framework that allows the creation of new lessons and strategies. For our investigation we created a linked list module and a worked-out example module.

Next, in Chapter 5, we show that worked-out examples may be of benefit to some students, some of the time, and cannot be considered a silver bullet. There, we look at the effectiveness of WOEs in an ITS. We conducted a study comparing the learning gains achieved by students who used WOEs against those that did not. Experiments were conducted over several points in time where we make our conclusion. In addition to this we see that a student's worked-out example behavioural patterns can offer insight into potential learning gains.

In Chapter 6, we uncovered that short, concise, worked-out examples may be more beneficial to advanced students than longer ones. Our investigation included the development of three extra types of examples - short, no exit, and time-out - where each group was analysed. The other two forms of examples do not appear to be more beneficial to either novice or advanced students.

Finally, in Chapter 7, we provide two concluding contributions. Firstly, we show that a level of initial student knowledge can be classified in an ITS, which can be improved via the use of WOE based features. Secondly, we describe an adaptive pipeline that can change the system's WOE delivery to improve a student's learning gain. This pipeline uses insights from prior chapters.

This work was carried out as part of the larger ChiQat-Tutor project, where several collaborators developed and evaluated three different fundamental computer science lessons, along with multiple strategies. The foundational work on the linked list module and data collection was primarily the work of Davide Fossati (Fossati et al., 2009). Work involving extending the corpus annotations and analysis (Chapter 3) was the joint effort of Lin Chen and myself.

The ChiQat-Tutor (Chapter 4) system was primarily developed by myself with collaboration from Omar Alzoubi, Mehrdad Alizadeh, and Rachel Harsley. Experiments were carried out over numerous laboratory sessions at the University of Illinois at Chicago (UIC) to evaluate various forms of intervention - these were carried out by myself, Rachel Harsley, Mehrdad Alizadeh, and Sabita Acharya. Following on from this, all documented analysis (Chapters 5 and 6), and proposed ITS enhancements (Chapter 7), were carried out by myself - valuable feedback was received from collaborators and peers throughout the process.

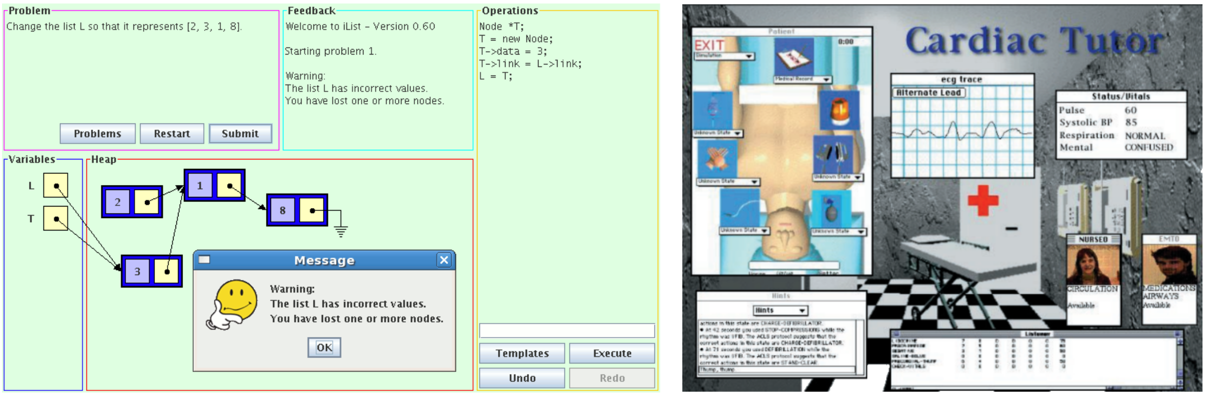
# CHAPTER 2

# RELATED WORK

Intelligent tutoring systems have evolved over the last several decades: their aim is to provide an educational environment on an artificial platform that is able to effectively teach individuals. Currently, human tutors are one of the most effective ways of doing this. Thus a goal of ITS literature is to emulate traits of human tutoring to yield similar learning performance.

(Corbett et al., 1997) suggests that an ITS is broken into four broad components: problem solving environment, domain knowledge, student model, and pedagogical model. The problem solving environment is the application interfaces that the student uses, such as a graphical or embodied interface. The domain knowledge model represents the specific knowledge that is being conveyed in an ITS lesson. Student modelling attempts to create a model of the student's behavioural states, trying to understand where the student is in terms of knowledge, with the aim of guiding the student to a goal state (enhanced learning). Finally, the pedagogical module aims at providing structure to the teaching of a topic. This may include the ordering of material delivered, or the way intervention is given to the student.

Many ITSs have been developed and deployed (Mills et al., 2004; Gadgil and Nokes, 2009; Smith, 2001; del Vado Vrseda et al., 2009; Wolfe et al., 2016; Aleven et al., 2016; Mostow and others, 2001), which cover numerous domains and objectives. Each of which may employ different teaching strategies, problem solving is a popular one, along with various types of feedback, on-demand hints, and worked-out examples.



Change the list L so that it represents [2, 3, 1, 8].

Welcome to iList - Version 0.60

Starting problem 1.

Warning:
The list L has incorrect values.
You have lost one or more nodes.

```
Node *T;
T = new Node;
T->data = 3;
T->link = L->link;
L = T;
```

Warning:
The list L has incorrect values.
You have lost one or more nodes.

| Status/Vitals | |
|---|---|
| Pulse | 60 |
| Systolic BP | 85 |
| Respiration | NORMAL |
| Mental | CONFUSED |

actions in this state are CHARGE-DEFIBRILLATOR.
• At 42 seconds you used STOP-COMPRESSIONS while the rhythm was VFIB. The ACLS protocol suggests that the correct actions in this state are CHARGE-DEFIBRILLATOR.
• At 71 seconds you used DEFIBRILLATION while the rhythm was VFIB. The ACLS protocol suggests that the correct actions in this state are STAND-CLEAR.
Thump, thump.

Figure 1: iList Interface (left) - Cardiac Tutor Interface (right)

## 2.1 Computer Science Intelligent Tutoring Systems

Related to computer science, the iList system (Fossati et al., 2008) aims to help students in computer science learn how to use the basic data structure of linked lists (Figure 1 left). The user is given several problems, which they have to solve by giving programming statements, in C++ or Java, to manipulate the list from an initial state to a goal state. It uses different feedback styles, reactive and proactive, the enhance learning gains for students. Our work is inspired and derived from the iList project.

There are intelligent tutoring systems that help novice programmers in various ways. (Choudhury et al., 2016) helps students in improving their coding style in languages such as Java. They employ an automated help generation scheme that utilises instructor authored guidance, and also leverages the stylistic features of fellow students. The author claims that 70% of students improve their coding style while using the tutor, while only 13% in the control group do so.

The JavaTutor system (Wiggins et al., 2015; Wiggins et al., 2016; Ezen-Can and Boyer, 2015) is an example of an ITS that aids students in learning the Java programming language. The application provides tasks to the student, such as printing a string to the console, or assigning a value to a variable. The student will then have to type Java code into the application while gaining feedback from the virtual tutor. The system does not just use the student's code to provide feedback, but can do so via facial expression recognition, physiological features, and affect (which is acquired through a depth map from a multi spectral camera).

Another ITS to aid students in learning Java is the JITS system (Sykes and Franek, 2003). JITS aid students in learning some of the fundamentals of the Java programming language, such as the use of variables, operators, and loops. The system can provide feedback to the student in the form of hints.

Aside from programming, (Hooshyar et al., 2015) outlines an ITS that helps students learn the fundamentals of problem solving and computer programming via the use of flowcharts. Their ITS engages students by allowing natural language to be used in the manipulation of such flowcharts that embody process flow for a given problem. They conclude that such a system can be of use to students of all levels, however, the biggest gains are in novice students.

ITSs for database design also exist, such as EER-Tutor (Mitrovic and Suraweera, 2016). The issue with design tasks is that there can be many ways of expressing a solution. The authors tackle this by using constraint satisfaction in identifying if a student is producing a design that conforms to a specification. Various forms of feedback were also experimented within the system.

## 2.2 Other Intelligent Tutoring Systems

Although our focus in this work is within the computer science domain, it is also worth mentioning that ITSs are not constrained to such a domain. A multi-domain ITS is AutoTutor (Graesser et al., 2005), which teaches computer literacy, physics, and critical thinking. This differs from most other ITSs due to its focus on natural language interaction.

In the medical domain, Cardiac Tutor (Woolf, 1996) is a simulator for teaching medical students about cardiac resuscitation. This provides students with the all important offline practise where mistakes are not costly. More importantly however, the tutor will customise problems for the student's level of competency, gives help when needed, will provide positive feedback for correct steps, and will also comment on incorrect steps. This ITS makes heavy use of graphics on screen to provide a rich multimedia learning experience (Figure 1 right).

### 2.3 Worked-out Examples

Worked-out examples, also called 'worked examples', have grown in popularity since the 1980's. They give a step-by-step example in solving a problem, such an example can be seen in Table I. First coined by Sweller and Cooper (Sweller and Cooper, 1985), they found it to be an effective teaching strategy, especially for novices. Sweller went further into the reasons for this in 1988 (Sweller, 1988). It is suggested that the foundations of domain specific knowledge lay in the form of schemas. These schemas are at the core of problem solving, and can be described as a cognitive structure, that allows problem solvers to recognise the type and state of a problem. From this state, and understanding the goal state, it is therefore possible to arrive at a solution via a number of steps. If a human does not hold a schema for a particular problem, it tends to be very difficult to identify the type of problem, or make rational decisions on how to arrive at the desired solution.

Based on this, Sweller claims that conventional problem solving may not give the best opportunity for learners to acquire these schemas. Therefore, problem solving alone may not be a wise choice for

| |
|---|
| let’s come here and say we searched for 9 |
| um we’d come down here we’d check 5 |
| 9 is greater than 5 so we go to its right child |
| um 9 is greater than 7 we’d check its right child |
| 9 is greater than 8 and we check its right child |
| 9 is equal to 9, would return it all the way up to 5 |

TABLE I: Worked-out Example for a Binary Search Tree Problem

someone to learn a new topic. The reasons for why schema acquisition is not optimal in problem solving is based on cognitive load theory. This theory is centred on the idea of working memory in humans. Working memory (Miller, 1956; Miller et al., 1960) is the theory that part of human cognition is devoted as a temporary store, much like a scratch pad. This store can receive information, manipulate it, and works with other cognitive subsystem, such as short term memory. Importantly, working memory is where humans process complex information, and therefore needed for learning. Unfortunately, working memory is not an infinite resource and differs between individuals. However, trying to perform too many activities at once may exhaust working memory, and therefore yield to cognitive overload, which may hinder learning. Sweller claims that learning instruction should be tailored towards the availability of resources.

All activities require some form of cognitive load. However, there are three distinct types of cognitive load, those being intrinsic, extraneous, and germane (Paas et al., 2003). Intrinsic load is related to the difficulty of the material being learnt - the more difficult the material is, the more cognitive resources are required to process it. Extraneous load are additional cognitive stresses on the learner that are not related to the core material to be learnt. This could be in the method that the material is being taught, or any noise that is distracting the learner. Germane load is the cognitive load associated with actually assimilating material to create knowledge. From these components, it can be deduced that to optimise working memory, cognitive resources should be placed into germane load where knowledge is being acquired. Fortunately, the latter two types of load are related to the instructional design of the material and can be controlled to a degree.

Intrinsic load is more difficult to control. Paas refers to a task as being an element of varying interactivity, where interactivity is how coupled the element is with other elements. For example, Paas describes a photo editing application, where a low interactive element would be how the function keys operate - each key performs a single function, independent from other keys. The learner can then gain knowledge from each, one at a time. A highly interactive element would be the image processing capabilities, where a single task is related to numerous intertwined features that need to be understood individually. However, it is sometimes possible to build elements from sub elements, therefore, intrinsic load may be reduced by breaking a difficult piece of knowledge into different components, and then reconstitute it to form the whole.

It is thought that worked-out examples may be more effective than problem solving alone due to their potential of reducing certain types of cognitive load. Problem solving naturally places a far higher load on the learner, since they have to understand new material, acquire new knowledge, commit it to memory, and solve a specific problem at the same time. Bearing in mind that working memory may only be able to process two or three novel concepts at once, any reduction may aid in learning. Examples, on the other hand, take away the additional extraneous overhead of problem solving and allow more cognitive resources to be allocated to germane cognition. This will allow the learner to build the appropriate schemas more efficiently. Once the schemas are built, the learner could reinforce this learning with additional problem solving, which no longer includes the cognitive overhead induced by the need to create these schemas.

Although examples appear to be a useful strategy in reducing cognitive load, there is the possibility of introducing *expertise reversal* (Kalyuga et al., 2003). As has already been shown, examples may be more effective at creating schemas which is ideal for beginners. For more advanced students, examples may not be beneficial since they already have acquired such base knowledge - a greater focus on problem solving may be a more beneficial. However, Kalyuga et al. noticed that advanced students may also be hurt by being given examples. This phenomenon suggests that not all student should be given examples and that student's knowledge should be evaluated before example deployment to make sure they learn efficiently.

To further complicate matters, working memory capacity varies between students. One study (Van Gerven et al., 2002) looked at the usage of WOEs across age groups. It is thought that working memory capacity degrades with age. Therefore, as Van Gerven et al. shows, WOEs are can be more effective for mature students than problem solving alone. Looking at it the other way, younger students may not need the cognitive reducing properties of examples to learn material. It is not just age, but learning difficulties such as dyslexia (Jeffries and Everatt, 2004) also contribute to deficiencies in working memory. Thus, it is not simply a matter of reducing cognitive load that may lead to more effective learning, rather than it may only be valuable to certain populations and individuals. Unfortunately, identifying such individuals is not a trivial task.

## 2.4 Worked-out Examples in Intelligent Tutoring

Worked-out example have also been used in various ITSs, usually with positive results. (Najar and Mitrovic, 2013) utilised WOEs in their CS ITS for teaching SQL queries. They concluded that while useful and promoted learning, it was best to used with tutored problem solving. (McLaren et al., 2008b) integrated WOEs into an ITS that also includes tutored problem solving. No significant learning gains were observed over problems only, however students were able to learn faster. (Liu et al., 2016) also looked at using WOEs in an open-ended data-driven logic tutor. The authors showed that WOEs were effective for beginners, however had the same effect as hint-based systems. Advanced students did not benefit from the use of WOEs.

Other ITSs have used a hybrid approach. (Renkl, 2005) introduced the concept of faded examples. This type of learning involves students to start using examples, and then transition to problem solving by changing an increasing number of solution steps into problem solving steps. This level of interactivity counteracts the inherently passive properties of traditional examples. The way this occurs can differ, but has included adaptivity based on the correctness of answered problem steps - i.e. if all problem steps are correct, change an additional solution step into a problem step next time. (Najar et al., 2015) used this strategy in their ITS with either one or two solution steps being allowed to be turned into problem steps. This approach has been shown to provide benefits over traditional examples, but only on near transfer tasks (where problems have the same underlying structure, but different surface features) (Renkl, 2002).

Furthermore, (Najar et al., 2014) looked into an adaptive flow of material delivery method in their SQL ITS. Rather than using a static flow of problems or examples, their system included three types of activity; supported problem solving, WOE, and faded WOE. At each activity step, the system decides which of the three types of activity that should be given to the student. They claim that students using an adaptive strategy learnt significantly more than their static counterparts. More recently (Najar et al., 2016) concluded with a similar experimental setup that novice students benefit from an adaptive system, rather than a static strategy, in terms of learning efficiency. Advanced student benefit from an adaptive system in way of greater learning. (McLaren et al., 2016) echoes similar sentiment in their study of worked-out examples within the chemistry domain. The authors suggest that there were no significant learning gains between conditions (worked-out examples, erroneous examples, tutored problems, and untutored problems), but students who studied via examples spent 46%-69% less time to complete the activity.

# CHAPTER 3

# WORKED-OUT EXAMPLES IN HUMAN TUTORING

In order for us to determine if the use of worked-out example is a viable option in a CS ITS, we will firstly evaluate this teaching strategy in the real world, by evaluating human teaching in CS. Here, we detail our data collection and analysis of human tutoring dialogues that use this strategy.

## 3.1 Development of a Human Tutoring Corpus in Computer Science

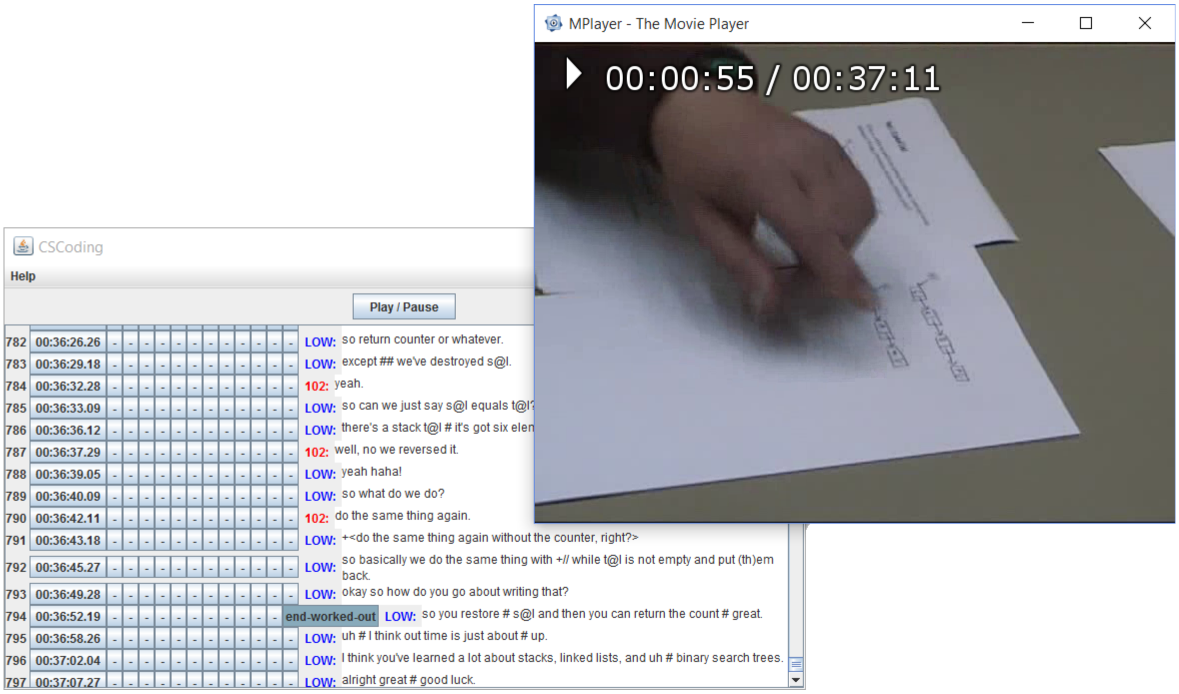
One-on-one, personalised tutoring is a very effective method of teaching, therefore, it should be possible to attain this high level of quality from an ITS if it were to be modelled on this type of sessions. Prior work (Fossati, 2009a) collected 54 sessions of human-to-human tutoring data within our domain of interest. Each tutoring session involved the tutors giving useful, relevant, feedback to students who were learning CS fundamental data structures. All sessions included one tutor and one student, and were approximately 40 minutes long. The sessions were split between two tutors, JAC being an advanced undergraduate student at UIC, and LOW begin an experienced lecturer.

Each of the 54 sessions consisted of three stages, a written pre-test, a tutoring session, and a post-test. Both tests included material on fundamental CS data structures: linked lists, binary search trees, and stacks. Once the pre-test was taken, the student participated in the tutoring session with one of two possible tutors. The tutor did not have access to the given tests in a way to avoid the tutor teaching to the test. Instead, the tutor was given a short assessment on the student's abilities demonstrated during the pre-test. This assessment enabled the tutor to individualise the session. After the tutoring session, the student took the post-test. Each of the tests were graded (out of a maximum of 15 points) and learning gain was calculated for each student.

All sessions were recorded with a camcorder and transcribed by human annotators. Each utterance spoken in the dialogue was tagged as either being spoken by the student or the tutor. Fossati developed an annotation tool called CSCoding (Figure 2), where the video timestamp of the tutor/student interaction was tied together with the transcribed utterances. Next, a predefined set of annotations were used to tag the utterances in each session. This tool was used in the past to annotate the dialogues with numerous features, including: student initiative (SI), prompts (PT), positive feedback (PF), negative feedback (NF), direct procedural instruction (DPI), and direct declarative instruction (DDI). The CSCoding tool is a flexible annotation tool, with the ability to add more annotation tag types as and when required.

The dialogues have already been used to show how various feedback strategies can be of benefit to students (Fossati et al., 2010; Fossati et al., 2015). We build on this work by looking for the use of worked-out examples in the dialogues.

During the annotation process, it was clear that another teaching strategy besides WOEs were occasionally being used, sometimes in conjunction with an example, that being the concept of *analogies*. Learning by analogy is a mechanism by which features of a known concept are mapped to features of an unknown concept (Gentner, 1998). Analogy plays a major role in learning, to the point that some researchers consider it the core of cognition (Hofstadter, 2001). It too can be effective in the early stages of learning especially when learners may lack appropriate prior knowledge (Gentner et al., 2003). An example of an entire analogy can be seen in Table II. Two collaborators, Mehrdad Alizadeh and Rachel Harsley, investigated the use of analogies in our tutorial dialogues (Alizadeh et al., 2015). They were



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 782 | 00:36:26.26 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | so return counter or whatever. |
| 783 | 00:36:29.18 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | except ## we've destroyed s@l. |
| 784 | 00:36:32.28 | - | - | - | - | - | - | - | - | - | - | - | - | 102: | yeah. |
| 785 | 00:36:33.09 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | so can we just say s@l equals t@l? |
| 786 | 00:36:36.12 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | there's a stack t@l # it's got six elen |
| 787 | 00:36:37.29 | - | - | - | - | - | - | - | - | - | - | - | - | 102: | well, no we reversed it. |
| 788 | 00:36:39.05 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | yeah haha! |
| 789 | 00:36:40.09 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | so what do we do? |
| 790 | 00:36:42.11 | - | - | - | - | - | - | - | - | - | - | - | - | 102: | do the same thing again. |
| 791 | 00:36:43.18 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | +<do the same thing again without the counter, right?> |
| 792 | 00:36:45.27 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | so basically we do the same thing with +// while t@l is not empty and put (th)em back. |
| 793 | 00:36:49.28 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | okay so how do you go about writing that? |
| 794 | 00:36:52.19 | - | - | - | - | - | - | - | - | - | - | - | end-worked-out | LOW: | so you restore # s@l and then you can return the count # great. |
| 795 | 00:36:58.26 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | uh # I think out time is just about # up. |
| 796 | 00:37:02.04 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | I think you've learned a lot about stacks, linked lists, and uh # binary search trees. |
| 797 | 00:37:07.27 | - | - | - | - | - | - | - | - | - | - | - | - | LOW: | alright great # good luck. |

Figure 2: CSCoding - Annotation tool for Tutorial Dialogue

involved in tagging analogical instances and analysing the data to find any link to growth. Since there appeared to be some overlap between WOEs and analogies during the annotation process (although Table II does not contain a WOE), we consider this link in our work.

Next, we take a look at how each of these strategies were annotated, and show some analysis that was performed on the annotated datasets.

## 3.2 Corpus Annotation

The tutorial dialogue corpus was annotated for worked-out example instances. Two annotators were assigned to the annotation task, which involved developing a coding manual, as well as labelling for the appropriate instances in all 54 sessions.

| | | |
|---|---|---|
| 219 | JAC | think of the stack as a bunch of Legos, okay? |
| 220 | JAC | and each time you put out a Lego... |
| 221 | JAC | okay, we'll call this, we'll just go a, b, c, d, e, and so forth. |
| 222 | JAC | okay? |
| 223 | JAC | so we're stacking our Legos up. |
| 224 | JAC | if we want to take a Lego off we can only take the Lego off that we just inserted. |
| 225 | JAC | right? |
| 226 | JAC | because we're building from the bottom up. |
| 227 | JAC | okay? |
| 228 | JAC | so we can only, take in, take off whatever we put in last. |

TABLE II: Analogy with Lego for Stacks

Before developing a coding manual, each annotator informally viewed several sessions to get a feel for the data available in the sessions. Both annotators contributed to the development of a draft manual. From here, five sessions were double coded by both annotators, which was then be compared via Kappa - a coefficient for intercoder agreement (Cohen, 1960; Carletta, 1996; Di Eugenio and Glass, 2004). Any differences between the two sets of annotations were discussed and resolved between the annotators. The coding manual was then modified to resolve any coding issues or ambiguity, that led onto another round of double coding (approximately another five sessions). Once a high degree of intercoder agreement was achieved during the double coding activity, the remainder of the sessions were split equally between the annotators.

The worked-out example annotations fell into three primary codes. For when an example only exists in a single utterance, the code 'single-woe' was used. If an example spans multiple contiguous utterances, the example begins with a 'begin-woe' and ends with an 'end-woe'. Further details can be found in the coding manual in Appendix A. After double coding seven sessions, a high degree of intercoder agreement was achieved with a Kappa of 0.82. This agreement was calculated based on the number of utterances that had the same WOE annotations. A full example of a WOE (with all annotations) can be found in Table III.

Similarly, analogies resolved down to three codes, those being ‘begin-analogy’, ‘end-analogy’, and ‘single-analogy’, all for the same reasons as given with the WOE codes. The analogy coding manual can be found in Appendix B. After annotating 15 sessions, we were able to reach an acceptable Kappa of 0.58 over the latter five sessions. Again, kappa was calculated on the number of utterances that have matching analogy based annotation codes.

## 3.3 Analysis of Tutoring Dialogues

### 3.3.1 Worked-out Examples

The analysis of the worked-out example in the tutoring dialogues uncovered some positive facts (Di Eugenio et al., 2013). Under all topics discussed in the tutoring sessions, WOEs were used throughout the sessions (Table IV). However, each of the topics used WOEs in different quantities and length. Stacks did not use WOEs in all sessions and WOEs were not frequently used. Instead the tutors tended to favour describing stack operations rather than using a problem to show their use. In binary search trees, tutors used examples very frequently, mostly to describe each of the cases that students may encounter while using them, e.g. deleting a node while one, two, or no child nodes. Examples were not used as often in linked lists as they were with trees, but would usually be longer. This confirms our hypothesis that tutors can use examples to teach some algorithmic CS concepts.

As in prior work (Fossati, 2008; Ohlsson et al., 2007; Chen et al., 2011), multiple linear regression was used on various features from the annotated corpus. The aim is to build a regression model that can explain any learning gain from a subset of available features. These models were build by brute-force,

| Line | Actor | Utterance | Annotation |
|---|---|---|---|
| 13 | LOW | um so <if we want to insert something such as here >[//] if we want to insert a new node containing g@1, and we want to insert after b@1, we need to be careful how we insert it. | |
| 14 | LOW | <because if we were to insert >[//] if we were to break this link first, between b@1 and f@1+... | |
| 15 | LOW | now nobody knows about f@l, right? | |
| 16 | Student | that would just kill it. | |
| 17 | LOW | it would just +// you wouldn't have anything that points to f@l, so there's no way for you to find f@1 xxx. | |
| 18 | Student | okay. | |
| 19 | LOW | okay? | |
| 20 | LOW | so if we were to insert g@l, and we want to insert f@l to b@l, the first thing that we do is draw it in here. | |
| 21 | LOW | this is for the pointer. | |
| 22 | LOW | it points to the next node. | |
| 23 | LOW | okay? | |
| 24 | LOW | first thing we would do is we would <check the >[//] find what it's pointing to, which is f@1, and we would set that equal to this. | |
| 25 | LOW | so that g@l now points to f@l. | |
| 26 | Student | okay. | |
| 27 | LOW | so now b@l and g@l both point to f@l. | |
| 28 | Student | okay and then you slide g@l in there b@l will point to g@l and g@l is pointing at f@l. | PF, SI |
| 29 | LOW | right. | PF |
| 30 | LOW | because if we were to just get rid of this now nobody is <pointing to f@l >. | |
| 31 | Student | right. | |
| 32 | LOW | and we don't know where f@l is. | |
| 33 | Student | f@1 just gets lost by doing this thing <its still >in line. | PF, SI |
| 34 | LOW | right. | PF |
| 35 | LOW | right, so then we would just come here and delete this. | PF |
| 36 | LOW | and now we have c@l k@l e@l g@l f@l. | |
| 37 | Student | okay. | |
| 38 | LOW | okay so we don't lose f@l. | |

TABLE III: Worked-out Example for a Linked List Problem with Annotations

| Topic | Sessions | Sessions $_{WOE}$ | Total WOEs | WOEs per Session | Words per WOE | Utt. per WOE |
|---|---|---|---|---|---|---|
| Lists | 52 | 50 | 180 | 3.5 | 498.3 | 48.3 |
| Stacks | 46 | 23 | 24 | 0.5 | 615.5 | 68.5 |
| Trees | 53 | 51 | 454 | 8.6 | 212.5 | 24.0 |

TABLE IV: Worked-out Example Statistics

including all permutations of features to see which is most accurate. The more accurate the model is (higher $R^2$), the greater confidence we have in being able to explain the model in terms of its component features, whether they are positively or negatively correlated with learning.

Prior work used this approach using the following features:

- Pre-test score
- Length of the tutoring session (number of dialogue moves)
- Count of each dialogue move type (SI, PT, PF, NF, DPI, DDI)
- Bigrams and trigrams of each of dialogue moves (consecutive and with gaps)

The base case of only using pre-test score in such models showed a negative correlation with learning, in that a lower pre-test score may indicate greater learning (Chen et al., 2011). Table V shows the correlation and the confidence given to the model in terms of $R^2$. Chen, et. al. goes one step further and created models for each of the three topics using combinations of the previously mentioned features (Table VI). In addition to including dialogue moves, bigrams and trigrams of moves were also utilised. The notion of a gap between elements in n-grams was used as a feature. This means that a feature with gap 0 would be all consecutive moves, while a gap of 1 skips a move after each element in the n-gram. Since this shows models having a greater $R^2$ than with pre-test alone, these models may be more descriptive of how to induce learning.

| Topic | Predictor | $\beta$ | $R^2$ | $P$ |
|---|---|---|---|---|
| Lists | Pre-test | $-.47$ | .20 | $<$ .001 |
| Stacks | Pre-test | $-.46$ | .296 | $<$ .001 |
| Trees | Pre-test | $-.739$ | .676 | $<$ .001 |

TABLE V: Regression Models that only Include Pre-test Score

| Topic | Predictor | $\beta$ | $R^2$ | $P$ | Gap |
|---|---|---|---|---|---|
| Lists | PT,DPI,FB<br>Pre-test<br>Length | .266<br>$-.463$<br>.011 | .415 | $<$ .01<br>$<$ .001<br>$<$ .05 | 0 |
| Stacks | DDI,FB,PT<br>Pre-test | $-.06$<br>$-.52$ | .416 | $<$ .01<br>$<$ .001 | 1 |
| Trees | +FB,SI,DDI<br>Pre-test | .049<br>$-.746$ | .732 | $<$ .01<br>$<$ .001 | 1 |
| Trees | FB,SI,DDI<br>Pre-test | .049<br>$-.746$ | .732 | $<$ .01<br>$<$ .001 | 1 |

TABLE VI: The most Explanatory Models Excluding WOE features

We repeated this regression model analysis with WOE features, which included the number of WOEs per session, length of those WOEs (number of words and utterances), as well as counts distinguishing the aforementioned dialogue moves that occurred within, and outside, of a WOE.

| Topic | Predictor | $\beta$ | $R^2$ | $P$ |
|---|---|---|---|---|
| Lists | Pre-test | $-0.442$ | .485 | $<.01$ |
| | WOE_Prompt | $-.0006$ | | $= 0.073$ |
| | WOE_#Utterances | .002 | | $= 0.092$ |
| | PF | .005 | | $= 0.099$ |
| Stacks | Pre-test | $-.37$ | .606 | $<.005$ |
| | WOE_PF | 0.077 | | $< .005$ |
| | WOE_Prompt | $-.021$ | | $<.05$ |
| Trees | Pre-test | $-.736$ | .737 | $<.0001$ |
| | WOE_[PF,SI,DDI] | .037 | | $< .005$ |

TABLE VII: The most Explanatory Models Including WOE Features

Table VII shows the top models generated from our feature set for each of our topics, where it suggests that WOE features may contribute to learning. Firstly, the contributions of WOE features is largely dependent on the topic they are used in, e.g. longer WOEs tend to be effective for learning linked lists, but not trees or stacks. Positive feedback is an important feature in these correlations when used within (as in lines 28-29, 33-35 of Table III) and outside of WOE regions. For stacks there is a slight correlation with positive feedback within examples. In addition to this, there is a strong correlation with learning for trees when positive feedback in part of the trigram [positive feedback (PF), student initiative (SI), direct declarative instruction (DDI)]. As for linked lists, there is also a correlation with positive feedback and learning. There are also some interesting negative correlations - prompts within WOEs were negatively correlated with learning for both lists and stacks.

### 3.3.2 Analogies

In (Alizadeh et al., 2015) we performed some in depth analysis of analogies . They were able to give some insight into when this strategy was used, as well as generating similar regression models as described in our worked-out example analysis.

Firstly, Alizadeh, et. al. did show that some topics favoured analogies over others. Table VIII gives some basic statistics of their usage. An interesting discovery here is that stacks tended to use the strategy more-so than lists and trees, and that stacks tended to use the strategy in most sessions (87%). Secondly, further regression models were built that distinguished between features within and outside of analogy regions. Worked-out example features were not included in these models. Table IX shows some of the best created models, where features that occurred within an analogy are prefixed with 'AN-'. It is suggested that distinguishing between features inside and outside of analogy can produce even more reliable models than ones without this distinction. This is most notable when used with linked lists. Thus, it indicates that analogies may be a promising feature to use within our target ITS.

| Topic | Sessions | Sessions $_{Analogy}$ | Total Analogies | Analogies per Session | Words per Analogy | Utt. per Analogy |
|---|---|---|---|---|---|---|
| Lists | 52 | 21 | 54 | 1.0 | 334.2 | 31.9 |
| Stacks | 46 | 40 | 63 | 1.4 | 121.2 | 10.8 |
| Trees | 53 | 16 | 22 | 0.4 | 54.5 | 5.5 |

TABLE VIII: Basic Analogy Statistics

### 3.4 Worked-out Examples and Analogies

Our results suggest that examples and analogies may contribute to learning. The question now is how these two strategies differ and if they can complement each other.

An interesting discovery from the human tutoring data is the overlap of WOEs and analogy. Table X shows the utterance count over all sessions that are contained in a WOE or analogy region. It shows that

| Topic | Model | Predictor | $\beta$ | $R^2$ | $p$ |
|---|---|---|---|---|---|
| List | 1 | Pre-test | -0.466 | 0.202 | < .001 |
| | 2 | Pre-test | -0.466 | 0.353 | < .001 |
| | | PF | 0.011 | | < .1 |
| | | DPI | 0.003 | | < .1 |
| | 3 | Pre-test | -0.330 | 0.388 | < .001 |
| | | Length | 0.011 | | ns |
| | | PF | 0.015 | | < .1 |
| | | DPI | 0.005 | | < .1 |
| | | PT | -0.003 | | ns |
| | | AN | 0.145 | | < .1 |
| | 4 | Pre-test | -0.41 | 0.472 | < .01 |
| | | PF | 0.007 | | ns |
| | | DPI | 0.29 | | < .01 |
| | | DDI | -0.001 | | ns |
| | | AN | 0.298 | | < .01 |
| | | AN-Length | 0.002 | | ns |
| | | AN-PF | 0.002 | | ns |
| | | AN-PT | -0.025 | | ns |
| | | AN-SI | -0.070 | | < .1 |
| Stack | 1 | Pre-test | -0.462 | 0.296 | < .001 |
| | 2 | Pre-test | -0.495 | 0.332 | 0 |
| | | PT | 0.010 | | < .1 |
| | | DPI | 0.007 | | < .1 |
| | 3 | Pre-test | -0.408 | 0.348 | < .001 |
| | | DPI | -0.01 | | < .1 |
| | | PT | 0.007 | | < .1 |
| | | AN | 0.137 | | ns |
| | 4 | Pre-test | -0.443 | 0.459 | < .001 |
| | | PF | 0.025 | | ns |
| | | DDI | -0.001 | | ns |
| | | PT | 0.025 | | ns |
| | | SI | 0.004 | | ns |
| | | AN-DPI | 0.031 | | < .1 |
| | | AN-PF | 0.035 | | ns |
| Tree | 1 | Pre-test | -0.742 | 0.677 | < .001 |
| | 2 | Pre-test | -0.703 | 0.689 | < .001 |
| | | DPI | -0.002 | | < .001 |
| | | PT | 0.001 | | ns |
| | 3 | Pre-test | -0.755 | 0.696 | < .001 |
| | | Length | -0.004 | | ns |
| | | DDI | 0.001 | | ns |
| | | AN | 0.069 | | ns |
| | 4 | Pre-test | -0.756 | 0.729 | < .001 |
| | | AN-Length | -0.006 | | ns |
| | | AN-DDI | 0.038 | | < .1 |
| | | AN-PT | -0.117 | | < .1 |

TABLE IX: Multiple Linear Regression Models

the use of analogies are far lower than WOEs, especially with respect to trees where there are about 100 times more utterances during a WOE than an analogy. Once again, the results differ drastically between topics, however, the topic of lists shows an interesting relationship between WOEs and analogies. Many of the WOE and analogy utterances do occur together; even more striking is that 60.8% of all analogy utterances also occur within WOE regions. This may suggest that lists may benefit from analogies, but these analogies are mostly used within an example. It also shows that about 12% of WOE content contains analogical concepts, showing that while potentially useful, not all WOE content should refer to analogies.

| Topic | Total Utt. | WOE Utt. | Analogy Utt. | WOE + Analogy Utt. | % of Analogy in WOE | % of WOE in Analogy |
|---|---|---|---|---|---|---|
| Lists | 12148 | 8692 | 1723 | 1047 | .608 | .120 |
| Stacks | 4929 | 1645 | 680 | 7 | .010 | .004 |
| Trees | 18514 | 10883 | 122 | 67 | .549 | .001 |

TABLE X: Presence of Examples with Analogies

Table XI shows the correlation between pre-test score and usage of the individual strategies. Here, the correlation is performed using the number of utterances in a session marked as the strategy, which is then correlated with the pre-test score of the session. This gives some idea of how much the strategy is used given prior knowledge of the student's abilities. The analysis shows that in all cases, except analogy use with the topic of trees, a lower pre-test score increases the chance of the strategy being used. However, each topic does have a different distribution from all others, which suggests that strategies are fairly dependent on the topic. For example, stacks show a drastic different between WOE and analogy usage with a low pre-score strongly correlating with the chance of an analogy being used. This is not the case with example usage with a near zero correlation. Lists on the other hand tend to increase the chance of analogies and examples being used given a lower pre-score with a similar negative correlation. Furthermore, the correlation for example usage when teaching lists is slightly stronger when analogies are introduced, suggesting that analogies are more likely to be used during a WOE episode when there is a lower pre-test score.

| Topic | Analogy | WOE | WOE + Analogy |
|---|---|---|---|
| Lists | -.311 | -.247 | -.296 |
| Stacks | -.446 | -.016 | -.117 |
| Trees | .119 | -.292 | .174 |

TABLE XI: Correlation Between pre-test and Analogy/Worked-out Example Utterance Counts

## 3.5 Summary of Findings

Based on our analysis of an annotated human tutoring corpus on introductory data structures, our hypothesis is confirmed that worked-out examples are regularly used. Furthermore, based on our pre/-post testing, there is an indication that such strategies may correlate with learning gains - a summary is given in Table XII. Additionally, analogies tend to be used in many cases, where a tutor introduces a concept using familiar constructs and entities, such as building a stack out of a stack of trays at a cafeteria. There are interesting variations though, as in the case of WOEs: linked lists tend to use WOEs extensively with long examples, however, teaching stacks does not use examples on a regular basis. Instead, tutors teaching stacks seem to prefer to use analogies in the majority of sessions. Interestingly, analogies tend to be used more-so with beginners with linked lists and stacks using analogies more when the pre-test score was lower (Table XI). Also it appears that in some cases, such as with the topic of linked lists, WOEs can use analogies at times, especially for those students with less knowledge of the topic.

| Topic | $R^2$ | Model Type |
|---|---|---|
| Tree | .737 | WOE |
| Tree | .732 | Standard |
| Tree | .729 | I/O Analogy |
| Tree | .696 | Analogy |
| Stack | .606 | WOE |
| List | .485 | WOE |
| List | .472 | I/O Analogy |
| Stack | .459 | I/O Analogy |
| Stack | .416 | Standard |
| List | .415 | Standard |
| List | .388 | Analogy |
| Stack | .348 | Analogy |

TABLE XII: Summary of Model Correlations

These results strengthen our hypothesis that WOEs may contribute to learning in our target domain. Also, analogies alongside WOEs may be a valuable strategy, especially for beginners, which is where this type of example may shine the most.

In the next chapter, we will present an overview of a computer science intelligent tutoring system, ChiQat-Tutor, that was built with experimentation in mind. The overview describes the system architecture, available lessons and strategies, and how it has aided us in conducting this research. Following the system overview, Chapter 5 continues our investigation into the use of worked-out examples.

# CHAPTER 4

# THE ARCHITECTURE OF CHIQAT-TUTOR

*This chapter discusses the architecture of the ChiQat-Tutor system's reusable framework, which was originally published at the 7th International Conference on Computer Supported Education. The content of this chapter is a shortened version of the publication - full details can be found in (Green et al., 2015).*

Since we have evidence of the importance of tutoring, and that particular strategies in the form of worked-out examples and analogies may be useful, the next step is to develop an ITS to utilise our findings. Here, we briefly outline the ChiQat-Tutor system as a whole which utilises three major building blocks; *lessons*, *teaching strategies*, and *utilities*, where each can be considered as system modules, or plug-ins. An overview of the system is given, showing a rich modular architecture which is designed for ease of adding new lessons and strategies, as well as encouraging reuse of system components across all modules.

## 4.1 Background and Motivation

ChiQat-Tutor builds on previous work at UIC, specifically, the iList ITS (Fossati, 2009b). iList had been shown to be effective in teaching the basics of the linked list data structure, yielding learning gains similarly to that of a human tutor. In iList, students were given a series of problems to be solved. Each problem presents the student with a linked list, as well as a task to complete. For example, the student may need to insert a node between the first and second nodes. This is accomplished by the student typing in Java-like commands, either one-by-one or as a block of code, that will modify the data structure until the desired structure has been constructed. The true power behind this system is its teaching strategies, in this case its various feedback models (Fossati et al., 2009; Fossati et al., 2010; Fossati et al., 2015). If the ITS sees the student making an error, or feels that the student is going down a potentially wrong path, it provides natural language feedback to the student to help them get back on track as soon as possible.

iList contains a single lesson type and primarily uses feedback (positive or negative) to aid the user. Our new system takes the lessons learnt from this work and expands on its successes. ChiQat-Tutor is able to handle multiple types of lessons, extend the use of teaching strategies, and include system utilities that may help in experimentation and evaluation for researchers.

### 4.2 Architecture

In developing the new system, we listed fundamental requirements that the system must possess. Full details on the system requirements can be found in (Green et al., 2015), however with respect to this research, the key is for flexibility for us to experiment with different variations of the system. Common base functionality that will aid in the running of such experiments was also important to smooth execution of the investigation. These requirements indicated that the system architecture would be key. For our own development, software reuse is important in order to help develop new components at speed and with a degree of consistency.

Given these basic requirements, we decided to develop a highly modular desktop application, with a sparse communication with a central server. Figure 3 shows an abstract overview of the system's major components for both client and server applications. Given our desire to keep as much processing as possible to be local to the user, the majority of the architecture is concentrated on the client side application. The server side is minimised to act as a system coordinator and data store.

Our architecture mirrors the approach taken by other researchers in recent times. One such example is the Generalized Intelligent Framework for Tutoring (GIFT) (Ososky, 2016), which attempts to provide a standard modular framework for ITS applications. The motivation by the authors was the lack of any reusable computer-based tutoring system (CBTS) framework. Although their framework appears promising, and is still receiving updates as of January 2017, this was not available in a stable form at the start of this project. The first release of GIFT was at the start of ChiQat-Tutor research, where using an unproven and immature framework may introduce unjustified risk to the project.

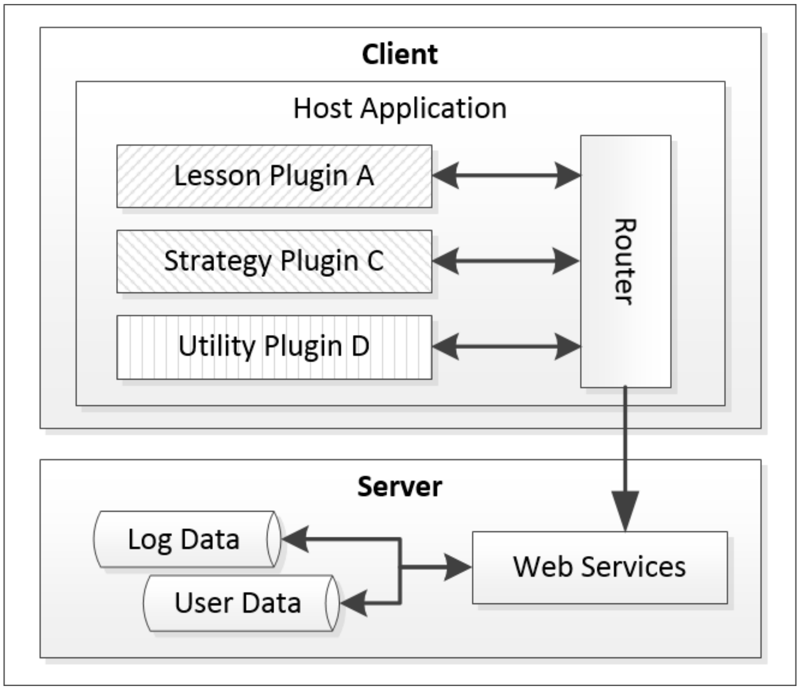


Figure 3: System Block Diagram.

### 4.2.1 Client

The core of the system is in the client side application, whose strength is in its modular architecture, where components can be created independently, however, can communicate together for a high degree of software reuse. This is valuable for experimentation among multiple researchers working on components for the project.

There are two main components in the client application; plug-ins and a host application. The host application acts as the entry point and glue of all client components. On execution, it will dynamically find all applicable plug-ins, automatically load, and initialise them. This process of plug-in selection will be invisible to the end-user, yet is configurable within the system. Application specific functionality is encapsulated in *plug-ins*. All plug-ins are derived from a common plug-in type, however, broadly speaking there are three types of plug-ins; lesson, teaching strategy, and utility plug-ins. Each plug-in can work with other plug-ins via an internal messaging API.

#### 4.2.1.1 Client Host Application

The host application provides a framework for the client side functionality to reside within. Its main function is to: (1) host plug-ins, (2) act as a container for a plug-in's user interface, (3) route messages between various components of the system.

Upon creation of the host application, it will load applicable plug-ins and initialise them as defined in the configuration file. An application can load any plug-in which exposes a set interface and has been defined within the system's blueprint configuration file. When plug-ins wish to communicate with each other, they can send a message through this interface to the application's router, which is hosted in the host application. The application router can either route the message to the destination plug-in, broadcast the message to all plug-ins, or handle the message itself. An example of the host application handling a message could be an application change of layout event.

A plug-in can expose various interfaces that the host application can access. The application can query for these interfaces, and update its own user interface. An appropriate case of this is when a plug-in exposes a view that it wants shown to the end user when a button is clicked. The host application can choose how it displays the view, for example, as a modal dialogue box or within the applications main window. This layer of abstraction exists to handle potential future interfaces, such as touch screen displays or multi-screen setups. The key feature of this style of application is that it can easily be rewritten, for example when porting to a new device.

#### 4.2.1.2 Client Plug-in

Plug-ins contain all ITS specific business logic and user interfaces for a well defined piece of functionality. Each plug-in is decoupled from all others and will only be able to communicate via an internal messaging system. A plug-in contains a queryable user interface which can be exposed to the host application, as well as a messaging interface which other plug-ins may pass messages through.

The major requirement on the part of the plug-in developer is to fulfil the plug-in's interfaces, which can be done by deriving from the abstract class *BasePluginInstance*. In terms of the message passing interface, this will require overriding of the *OnMessage* method to receive messages. Sending messages can be achieved by either calling the *PostMessage* or *BroadcastMessage* methods.

Also, the plug-in can pass back a set of queryable user interfaces, which are JavaFX (Topley, 2010) containers, by deriving from the *PluginViewFx* class and registering it with the BasePluginInterface. The interface, which will be displayed within the application will be wholly operated within the plug-in and changes cannot be made to the application in order to use new interface features. To complement the interface, another abstract class, *PluginController*, can be implemented to provide back-end implementation details.

#### 4.2.1.3 Lesson Plug-in

When plug-in developers wish to develop a new plug-in, they will have to create a new plug-in and implement the generic plug-in interface. However, lesson developers can bypass much of the standard work by deriving from a specialised lesson controller and view. This plug-in provides an additional framework where individual lessons can be created within. Many of the lesson basics which have been developed have been abstracted to this level, including a standard user interface with abstracted components for further specialisation, along with integration of existing teaching strategies, such as the worked-out example plug-in.

This is completely optional, yet should provide a consistent look and feel to the application, while speeding up the development process.

### 4.2.2 Server

The next major component of the system is the server, which has the role of acting as a centralised data manager within the systems ecosystem. The primary role of the server is to provide an online user authentication and profile store, repository for log messages, and binary asset data storage (such as plug-ins and computational models).

In order to improve the system, it is very important for us to know how users are using the system. Thus we include a very flexible logging system, where the client can send entries to the server for persistent storage. This data can then be mined for user behaviour that may help during offline analysis.

The system will also be able to log data while offline by storing entries locally until online connectivity can be obtained.

## 4.3 ChiQat-Tutor with Worked-out Examples

From the ChiQat-Tutor framework, we created a working ITS with numerous facets to demonstrate some of the strengths of the architecture in a practical setting. Here, we focus on the modules of interest; the linked list plug-in, worked-out example plug-in, and supporting utilities. Figure 4 gives an overview of the full system and the relations between components. An in-depth view of a larger selection of plug-ins can be found in (Green et al., 2015).

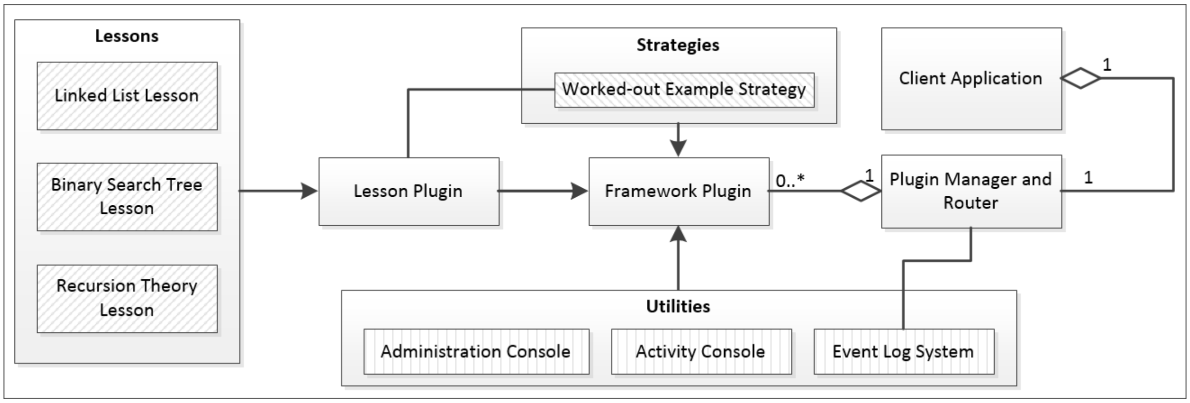
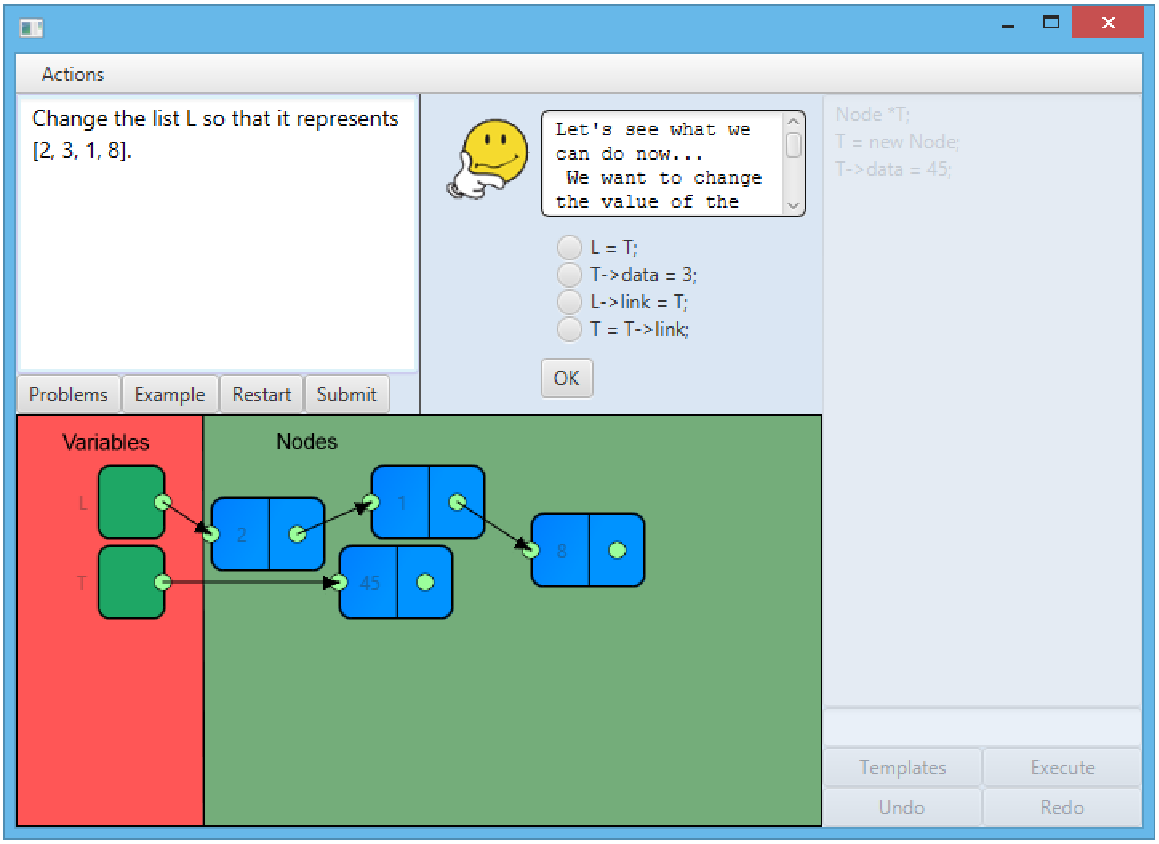


Figure 4: ChiQat-Tutor Plug-ins.

### 4.3.1 Linked List Lesson

Based on iList, this plug-in has been developed within the new system's framework (Figure 5). This lesson gives students the opportunity to manipulate linked lists in code, along with being presented the list graphically. The tutor also provides various hints, feedback, and corrections as needed. Students may experiment within a sandbox or work on one of the 7 provided problems. Each of the 7 problems exercise different techniques that must be used to achieve the desired outcome. Problems 1 to 5 tasks the student to manipulate a single linked list with one command submission at a time. For example, the student may submit a command to create a new node, then another to assign its next pointer to another node. Each step made by the student is then shown graphically in a graph in the user interface. Once the desired goal state is achieved, the student submits the solution to the system for final checking. Problems 6 and 7 varies from the previous problems in that the student has to write a block of code that is to manipulate multiple lists. The concept here is for the student to write a generic program to perform a task, such as find a node in the list with value ‘8’ by using a while loop. The student then submits the whole block of code at once, which is then executed. The student is given feedback from the tutor, and a graphical result for each of the manipulated lists.

This lesson leverages the systems common framework, services, teaching strategies, and common user interface. Worked-out examples in this lesson have been designed to be contextual - there is a relevant example for each of the seven problems available in the system. An example is played back to the student once they click the ‘Example’ button during a problem (Figure 6). They are allowed to play through each step of the example by clicking the ‘Next’ button. It is possible for students to play through the example multiple times, and they are allowed to terminate it at any time.



Change the list L so that it represents [2, 3, 1, 8].

```
Let's see what we
can do now...
 We want to change
the value of the
```

○ L = T;
○ T->data = 3;
○ L->link = T;
○ T = T->link;

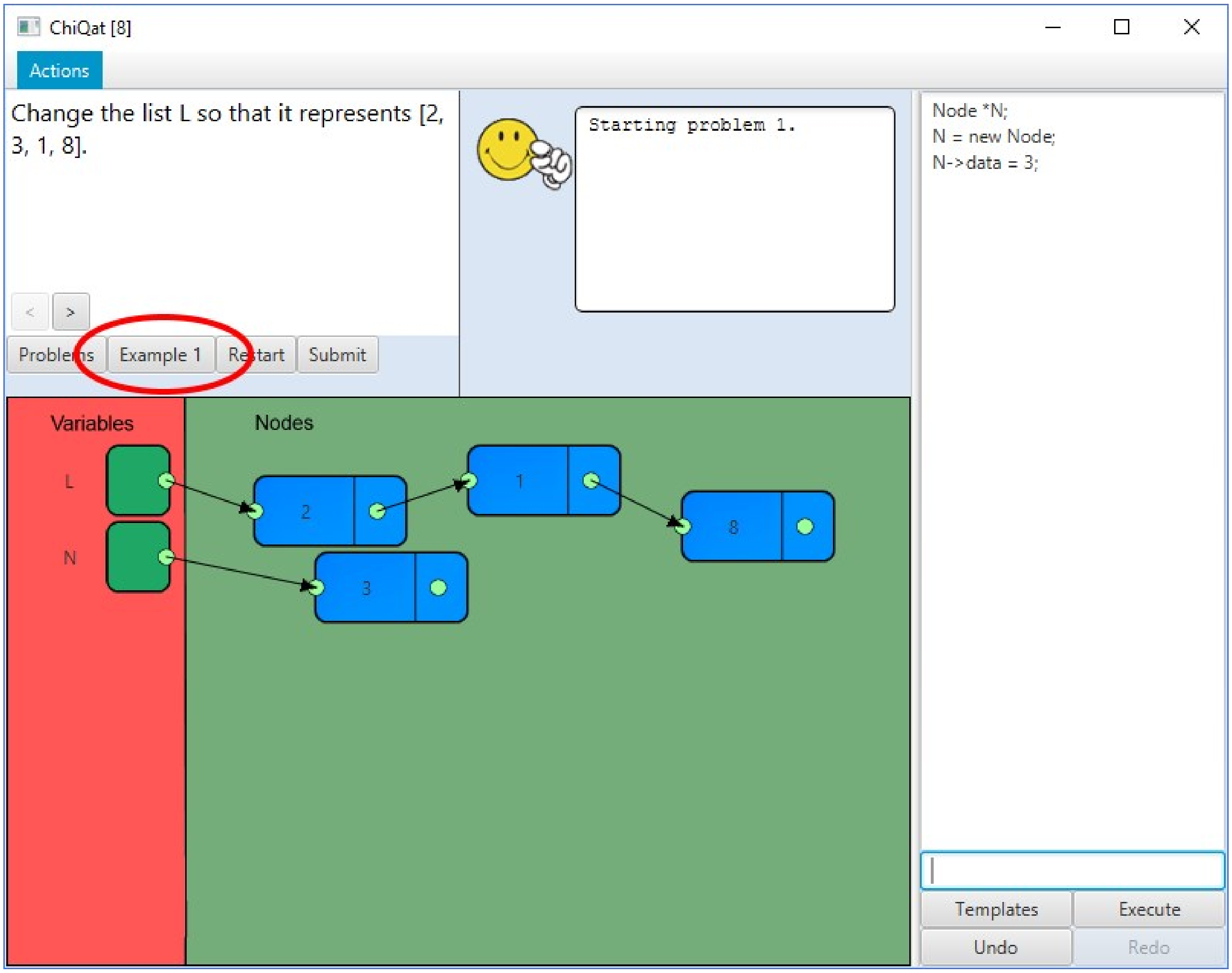
```
Node *T;
T = new Node;
T->data = 45;
```

Figure 5: Linked List Tutorial

### 4.3.2 Worked-out Examples

The Worked-out example plug-in gives the ability for any system plug-in to support this teaching strategy by including a well-defined interface via the system's messaging system. The plug-in is split into three major components; WOE engine, definition asset, and an editor for the definition asset.

At its core, a WOE is described via a definition asset (in XML) which is based on a cyclic directed graph, where each node is considered a WOE step. Each step is related to an action, such as the virtual tutor giving an explanation for the next solution step. The steps are then connected to other steps,



Change the list L so that it represents [2, 3, 1, 8].

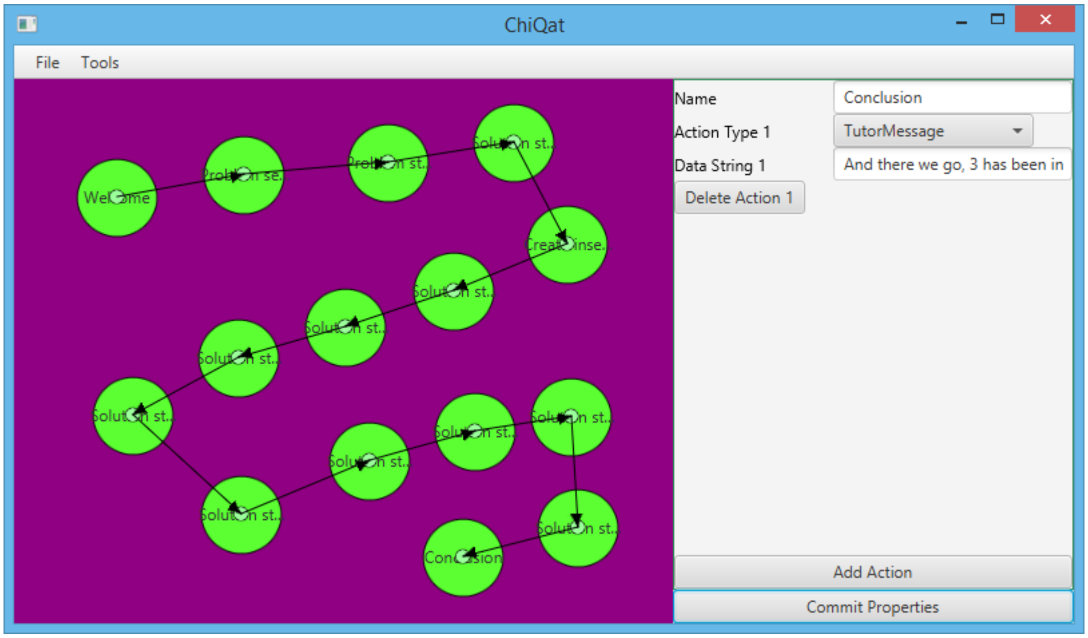
```
Starting problem 1.
```

```
Node *N;
N = new Node;
N->data = 3;
```

Figure 6: Linked List Tutorial with Example Button Highlighted

with the simple case being a series of steps leading from problem to solution in a serial fashion. The configuration file is given to the engine in order for processing. To ease the creation of such assets, a graphical editor is also provided to the lesson designer (Figure 7).

Finally, the WOE engine is used to play out a WOE session to the student. This works by executing actions at each step while making appropriate transitions to other steps in the WOE graph. Such

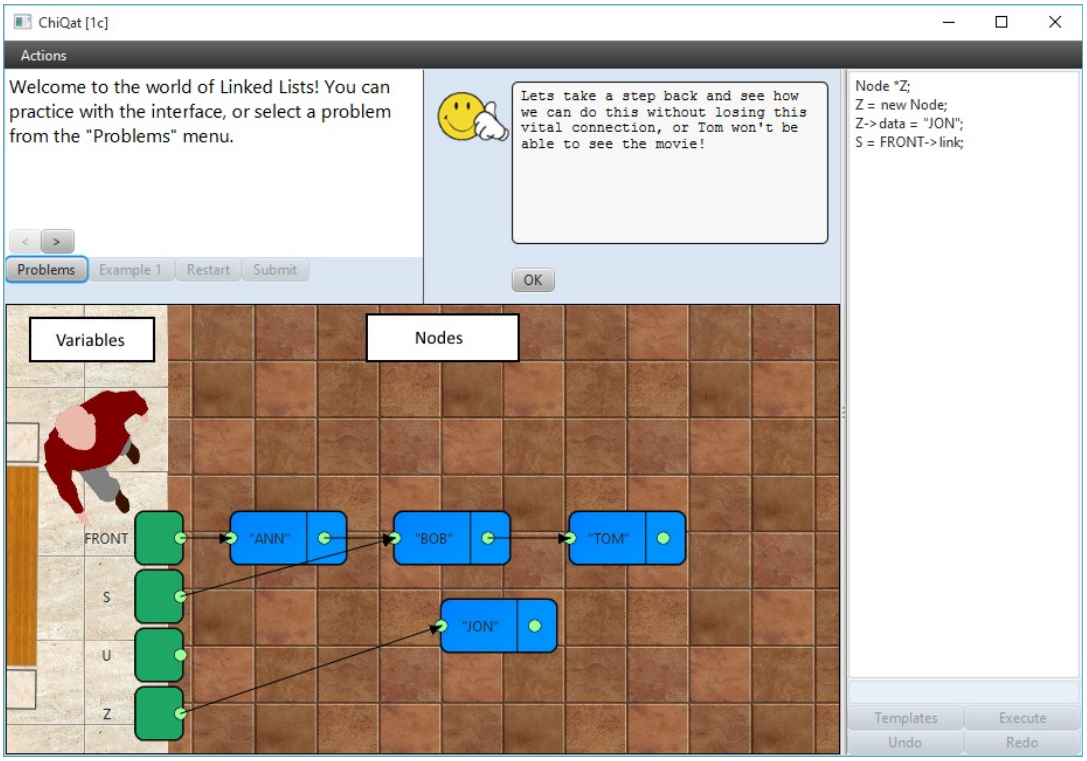


Name: Conclusion
Action Type 1: TutorMessage
Data String 1: And there we go, 3 has been in

Figure 7: Worked-out Example Editor

actions include printing to the console, displaying a tutoring message, and executing a command for the problem. Lessons can integrate this strategy by acting upon messages sent from the engine.

Since the engine has been designed to be generic, it is also possible for the WOE scripts to inject a command as a string into the calling module instance. This is how analogies can be defined in the system. The engine can make a request to the lesson to perform operations such as changing the background of the user interface, to altering the surface characteristics of structures being operated upon. This way, a lesson, such as the linked list lesson, may be driven by injecting a change of background to that of a movie theatre, while the nodes in the list could be given as ‘Alice’, ‘Bob’, and ‘Chris’ (Figure 8). A WOE graph could be created to give an analogical example, one that is presented in a familiar domain, that is then mapped onto the target domain.



Welcome to the world of Linked Lists! You can practice with the interface, or select a problem from the "Problems" menu.

```
Lets take a step back and see how
we can do this without losing this
vital connection, or Tom won't be
able to see the movie!
```

```
Node *Z;
Z = new Node;
Z->data = "JON";
S = FRONT->link;
```

Figure 8: Linked List Analogy Worked-out Example

### 4.3.3 Utilities

Although students learn via lessons, and are taught via teaching strategies, another important aspect are the general system utility plug-ins. Such utilities can be used to perform background tasks or add some additional usability to the system.

The entry point of the system is through a plug-in that exposes an automatic starting property. Here, we have made an *Activity Console* that is shown on startup. The console handles user log in, account creation, as well as shows all possible lessons and lesson options.

ChiQat-Tutor supports global event logging via the *Logging Plugin*, where text can be logged to the server's log database for future analysis. Such messages may include when a lesson is started, stopped, completed, and when user interface (UI) components are moved by the user. To ease development, generic ChiQat-Tutor messages such as starting and stopping a lesson, are handled within the framework.

The ChiQat-Tutor framework supports the notion of user privileges, whereby a user may be given standard user or administrative privileges. An administrator will have access to the *Administration Console*, which allows them to edit system properties, such as individual user settings and control parameters. This is useful while setting up experiments.

## 4.4 Summary

In this chapter we presented a flexible, modular, intelligent tutoring system framework, with the ability to extend to new domains and pedagogical strategies. Several modules have been developed and put together via its configuration file to provide a working ITS. Of importance to this research are the linked list, worked-out example, and utility plug-ins. This application has greatly aided us in our research objectives by providing a sound foundation for executing experiments, and creating new functionality that can be easily deployed for further experimentation. An ITS comprising this configuration was foundational to conducting experiments for this work.

Next, we continue our investigation by describing our experiments with ChiQat-Tutor and the WOE teaching strategy.

# CHAPTER 5

# THE EFFECTIVENESS OF WORKED-OUT EXAMPLES IN AN INTELLIGENT TUTORING SYSTEM

Thus far, we have shown that there is promise in using worked-out examples in the domain of computer science fundamental data structures. We have deduced this from our first task - Investigate the effect of worked-out examples on student learning with computer science concepts - where there can be a positive effect on students if WOEs are used. We have also outlined a system that is able to teach various lessons in the computer science domain.

The next step is to move onto our second task - If human-human worked-out examples are effective, see if it can also be effective in an intelligent tutoring system. Therefore, we detail the deployment of the system while observing the effect of worked-out examples on students in our linked list tutorial. Our hypothesis is that worked-out examples can yield significant learning benefits over traditional problem solving alone in a computer science intelligent tutoring system.

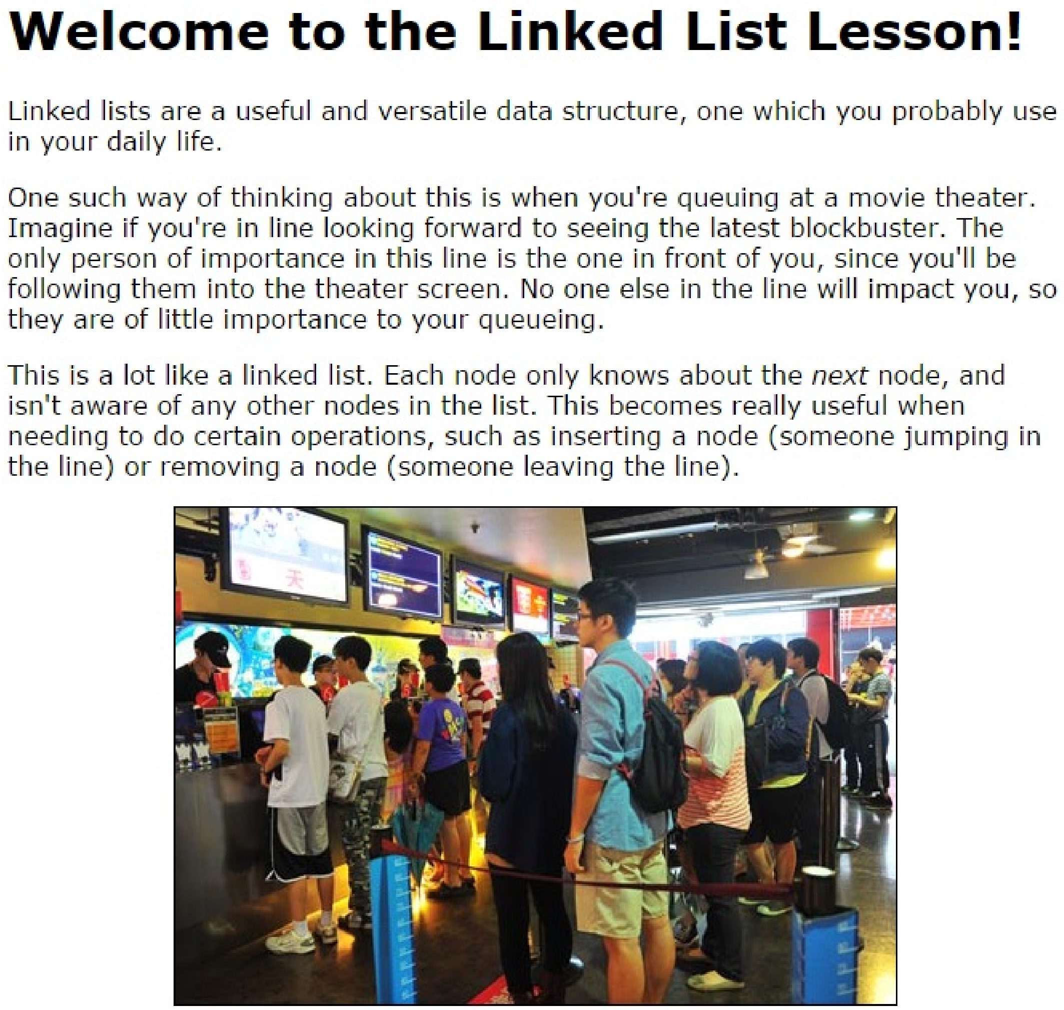
There will be four conditions of interest in our experiments:

- Control group - No examples, students have access to all seven problems. This is considered the baseline group that other groups will be compared against.
- Worked-out examples - Along with the 7 problems, each problem also has an on-demand example. This group will test the hypothesis that worked-out examples may be of benefit to students.

- Analogical worked-out examples - Contains the same seven problems, but has a unique on-demand analogical example per problem. This draws on our analysis that analogies are sometimes used in teaching the topic of linked lists. More specifically, half of all analogy utterances occur during WOE episodes, thus a WOE with analogical features may be effective.
- Worked-out examples with an analogy start screen - This will be referred to as *WOE + pop-up analogy*. Same as the worked-out example condition, with the addition of the student being shown a pop-up window with an analogy for linked lists on first run. This analogy is of queuing at a movie theatre (Figure 9). This variant has been introduced in addition to the prior group as around 40% of analogy utterances did not occur within a WOE episode. Thus, this group will evaluate the case of using analogy outside of a WOE.

## 5.1 Experiment Setup

Two experiments were conducted over a total of eight laboratory sessions (four laboratory sessions per experiment) as part of a second year class in computer programming (CS211 - Programming Practicum), which is a required course for CS majors, where it follows on from a class in program design (CS141), and is also a co-requisite of CS251 (Data Structures) and CS261 (Machine Organisation - basics of computer hardware). This class covers topics in C and Java programming, including some basic CS data structures. Although experiments were conducted on approximately the fifth week of the course, students had already been given an introduction to linked lists prior to experimentation. These laboratory sessions would serve the dual purpose of aiding us in experimentation, as well as giving students the ability to further exercise linked list usage.



# Welcome to the Linked List Lesson!

Linked lists are a useful and versatile data structure, one which you probably use in your daily life.

One such way of thinking about this is when you're queuing at a movie theater. Imagine if you're in line looking forward to seeing the latest blockbuster. The only person of importance in this line is the one in front of you, since you'll be following them into the theater screen. No one else in the line will impact you, so they are of little importance to your queueing.

This is a lot like a linked list. Each node only knows about the *next* node, and isn't aware of any other nodes in the list. This becomes really useful when needing to do certain operations, such as inserting a node (someone jumping in the line) or removing a node (someone leaving the line).

Figure 9: Popup Analogy

The two experiments were conducted on different cohorts of the same class. One cohort took the class in September 2014 (Sep14) and the other in February 2015 (Feb15). Other than the time of year, there were no major differences between the cohorts, e.g. same major, point in semester, and professor. Both classes included material on linked lists prior to our investigation.

Participating students completed a pre and post test (12 minutes each), while using the system in between tests for approximately 40 minutes. Both tests were identical, and were the same ones as used in iList evaluations (Appendix C). The activity was a solo activity where they were able to use the lesson in any way they wished within the constraints of the system. The students were randomly assigned to a single condition, with a rough split between applicable conditions in each session. Table XIII shows the available conditions in each of the two sets of experiments.

| Experiment | Sep14 | Feb15 |
|---|---|---|
| Control | ✓ | ✗ |
| WOE | ✓ | ✓ |
| Analogy | ✗ | ✓ |
| WOE + Pop-up Analogy | ✗ | ✓ |

TABLE XIII: Conditions over Experimental Conditions for Sep14 and Feb15

Participating students had their usage of the system recorded, with all actions performed being logged to the centralised system log. Pre and post tests were randomised and anonymised, which were then graded between two or three independent graders. Two graders were assigned to each test, and in the event of a difference in grades, the third grader would also grade that test.

Since the pre/post tests are the same as in Fossati's work, this allows us to also make some comparisons from these experiments with relative ease. However, six years have elapsed between the two sets of experiments; iList was evaluated around 2008 and experiments on our new ChiQat-Tutor were conducted from 2014 onwards. This six year difference is a variable to bear in mind as students behaviour may have changed over time, e.g. the pervasiveness of touch-screen technology may have changed the way students use a user interface.

## 5.2 Measuring Progress

Measuring progress in such a trial can be done in many ways, such as seeing how many problems a student successfully completes, to timing how long it takes for a student to complete a particular problem. Another option is to use the learning gain of the student, which is derived from the pre and post-test scores. Although this may appear to be a simple measure, there are various ways of calculating learning gain.

The simplest way to calculate learning gain is via *absolute learning gain*, which is the delta between a student's pre-test and post-test scores. With this calculation, all gains are considered equal, i.e. all students who score an extra point on the post-test will have an equal learning gain. It can be calculated using the formula in Equation 5.1.

$$\langle g \rangle = \langle post \rangle - \langle pre \rangle \tag{5.1}$$

An issue with absolute learning gain is when samples are collected across multiple student populations where pre-test scores wildly vary. Hake et al. devised an alternative measure called *normalised learning gain*, whereby learning gains are relative to how much a student can gain after intervention (Hake, 1998). It can be calculated using the formula in Equation 5.2.

$$\langle g \rangle = \frac{\langle post \rangle - \langle pre \rangle}{1 - \langle pre \rangle} \tag{5.2}$$

However, normalised gain does have some fundamental flaws, such as not being able to cope with negative learning. *Normalised change*, which was introduced by Marx and Cummings (Marx and Cummings, 2007), improves upon normalised gain by taking into account of negative gains, and handles edge cases, such as gaining a perfect score. Normalised change can be calculated with the formula in Equation 5.3.

$$c = \begin{cases} \frac{\langle post \rangle - \langle pre \rangle}{1 - \langle pre \rangle} & \text{post} > \text{pre} \\ drop & \text{post = pre = (100} \cup \text{0)} \\ 0 & \text{post = pre} \\ \frac{\langle post \rangle - \langle pre \rangle}{\langle pre \rangle} & \text{post} < \text{pre} \end{cases} \tag{5.3}$$

Using a more sophisticated learning gain method may not always be best. For either of the normalised variants, it gives a far greater weight to those students who scored very highly in the pre-test. The key to deciding whether absolute learning gain is sufficient depends on how much the pre-score of students varies between classes. If pre-scores do vary considerably, then normalised variants would be more suitable than absolute gain.

To test if the pre-score of students varied significantly, we conducted an ANOVA using the pre-test scores for all students grouped by experimental condition. This test showed no significant differences in condition pre-test scores with $[F(4, 128) = 1.6788, p = .1589]$. Also to check for significance between the two classes, we conducted an unpaired t-test using the pre-scores for all students in each of the two experiment sets - that is the pre-test of all students in Sep14 was tested against all students in Feb15.

The t-test reported $p = 0.6118$, which suggests that the two populations do not differ significantly. The means for each condition can also be seen in Table XIV. Since the pre-score tests do not wildly differ, absolute learning gain will be used for analysis.

## 5.3 Student Learning Gains

Table XIV shows the results of each condition in all preliminary experiments. The first striking observation is the vast difference in performance between the two groups in the Sep14 experiment set. The group without WOEs performed close to that of being given a human tutor (when compared to Fossati's experiments), while the WOE condition achieved *negative growth*. This in itself suggests that either WOEs are not effective in our particular ITS, or perhaps, the WOEs in the ITS were badly designed and inhibited student growth.

| Intervention | N | Pre-test | | Post-test | | Gain | |
|---|---|---|---|---|---|---|---|
| | | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| None | 53 | .34 | .22 | .35 | .23 | .01 | .15 |
| Human | 54 | .40 | .26 | .54 | .26 | .14 | .25 |
| iList-1 | 61 | .41 | .23 | .49 | .27 | .08 | .14 |
| iList-2 | 56 | .31 | .17 | .41 | .23 | .10 | .17 |
| iList-3 | 19 | .53 | .29 | .65 | .26 | .12 | .24 |
| iList-4 | 53 | .53 | .24 | .63 | .22 | .10 | .16 |
| iList-5 | 30 | .37 | .24 | .51 | .26 | .14 | .17 |
| ChiQat without WOE (Sep14) | 35 | .51 | .19 | .63 | .21 | .12 | .23 |
| ChiQat with WOE (Sep14) | 32 | .55 | .20 | .52 | .20 | -.02 | .23 |
| ChiQat with WOE (Feb15) | 23 | .41 | .18 | .56 | .22 | .14 | .17 |
| ChiQat with Analogy WOE (Feb15) | 21 | .50 | .17 | .61 | .15 | .11 | .18 |
| ChiQat with WOE + Popup Analogy (Feb15) | 22 | .52 | .21 | .60 | .23 | .09 | .21 |
| ChiQat (Sep14) | 67 | .53 | .19 | .58 | .21 | .05 | .24 |
| ChiQat (Feb15) | 66 | .47 | .19 | .59 | .20 | .12 | .19 |

TABLE XIV: Learning Gains of Students

However, the gains made from the second round of experiments, in Feb15, differ from the prior round by yielding similar learning gains as that from a human tutor. This suggests that the WOEs used in the system do not strictly inhibit growth in students, although there may be differences in the two groups that made the use of WOEs effective or ineffective. Surprisingly, the analogy groups performed similarly, yet were not as strong as the WOE only group in the Feb15 experiment set.

We performed an ANOVA on the five groups from both Sep14 and Sep15 experiments. Running an analysis for each group, based on absolute learning gain, showed significance in 4 conditions $[F(4, 128) = 2.83, p = .0273]$ (Figure 10 shows a box plot from the ANOVA).

From here, a Tukey post-hoc test was performed. For absolute learning gain (Figure 11 shows an interval plot), one pair of groups came out as significant, that being the difference between WOE groups in Sep14 and Feb15 with $p = .0314$. There was also significance between the WOE and no-WOE groups in the Sep14 experiments with $p = .0516$. Although puzzling, this unexpected significance does give way to some interesting lines of further investigation.

## 5.4 Log Analysis

One of ChiQat-Tutor's modules, the Event Log Utility, gives the ability to log a vast array of messages in the system. A list of messages, and arguments that may be attached to each message, is shown in Table XXXIX. These log events may be used to get a better understanding of what occurred during experimentation. All log entries primarily consist of:

- Time stamp (in milliseconds)
- Event type
- User ID

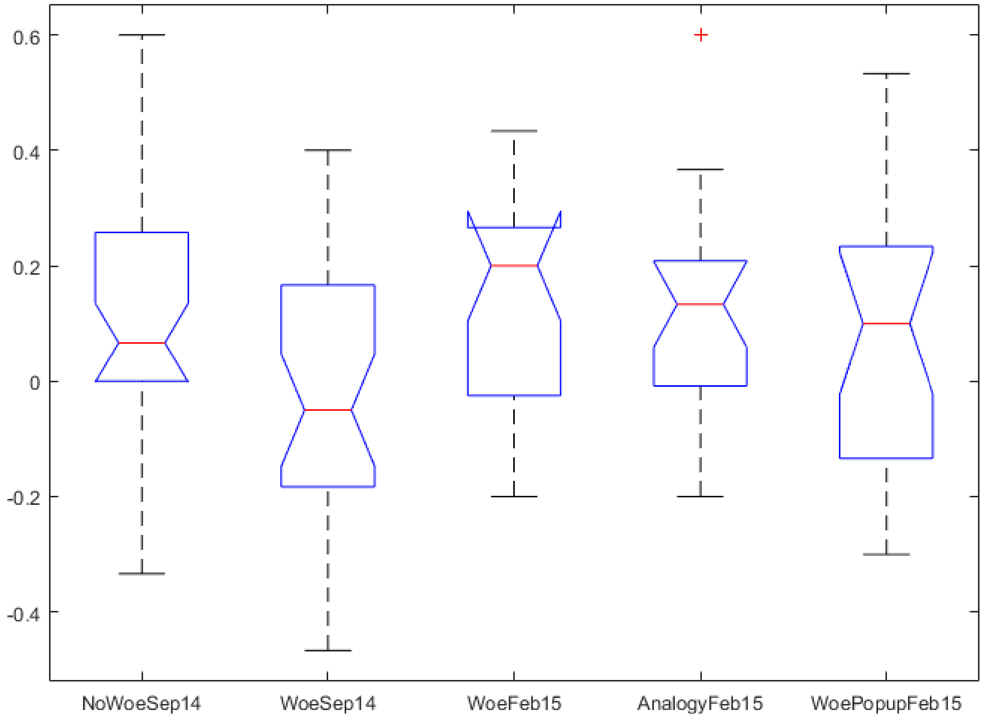


Figure 10: ANOVA for Sep14 and Feb15 Learning Gain

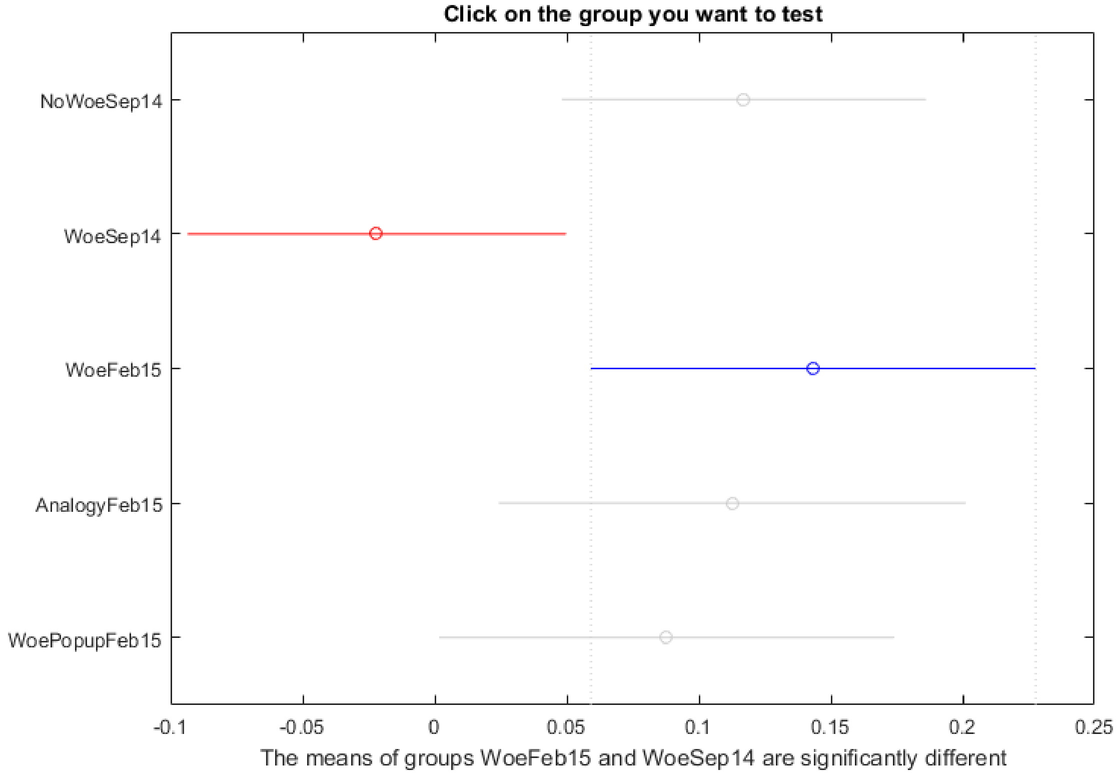


Figure 11: Tukey Post-Hoc Analysis on Learning Gain ANOVA

We cleaned and processed the logs, which involved the removal of some outlying samples. Outliers included sessions that contained no useful log entries, i.e. students who exited the system without attempting a problem. Some time was spent on cleaning some session data, such as aggregating sessions where a student restarted the system. After preprocessing, we extracted features to investigate the interaction of students with our system (Green et al., 2015). Two graphs give insight into some differences between the groups on a problem by problem basis. Firstly, Figure 12 shows the problem success rate for each of the five conditions. The clear worst performing group was the Sep14 WOE condition with a lower success rate for all problems. The Feb15 analogy condition does show a greater success rate for the first few problems, but drops after problem 4. The no-WOE condition has lower success on initial problems, but has high degrees of success over the mid-problems, with most students being able to complete problem 4 and 5 if problem 3 was completed. This may be due to the additional time consumed by the students while studying examples, not giving them as much time to progress on more advanced problems.

Figure 13 shows the amount of time students spent on each problem. It is important to note that each duration also includes the time spent on examples, if any were used. All the conditions show that students become more efficient at problem solving as they work through the problems. Interestingly it does not appear that viewing examples reduced the amount of time to solve problems, contrary to previous work that suggests WOEs may make students more efficient (Najar and Mitrovic, 2013).

Analysis of log files also showed achievement differences (Table XV). These basic statistics do show some differences in how Sep14 WOE users differed from all others. They tended to solve fewer problems, despite having the same amount of time as the no-WOE group, as well as making a similar

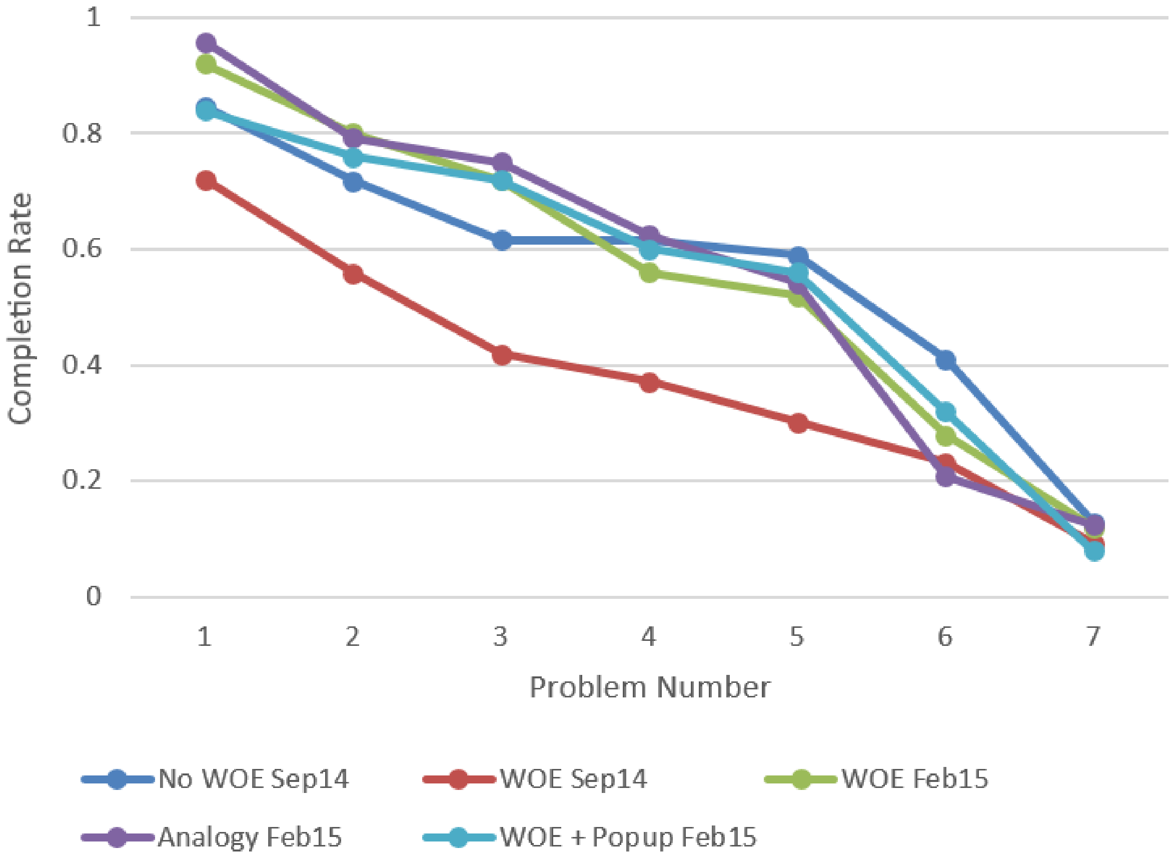


Figure 12: Percentage of Students Completing Problems

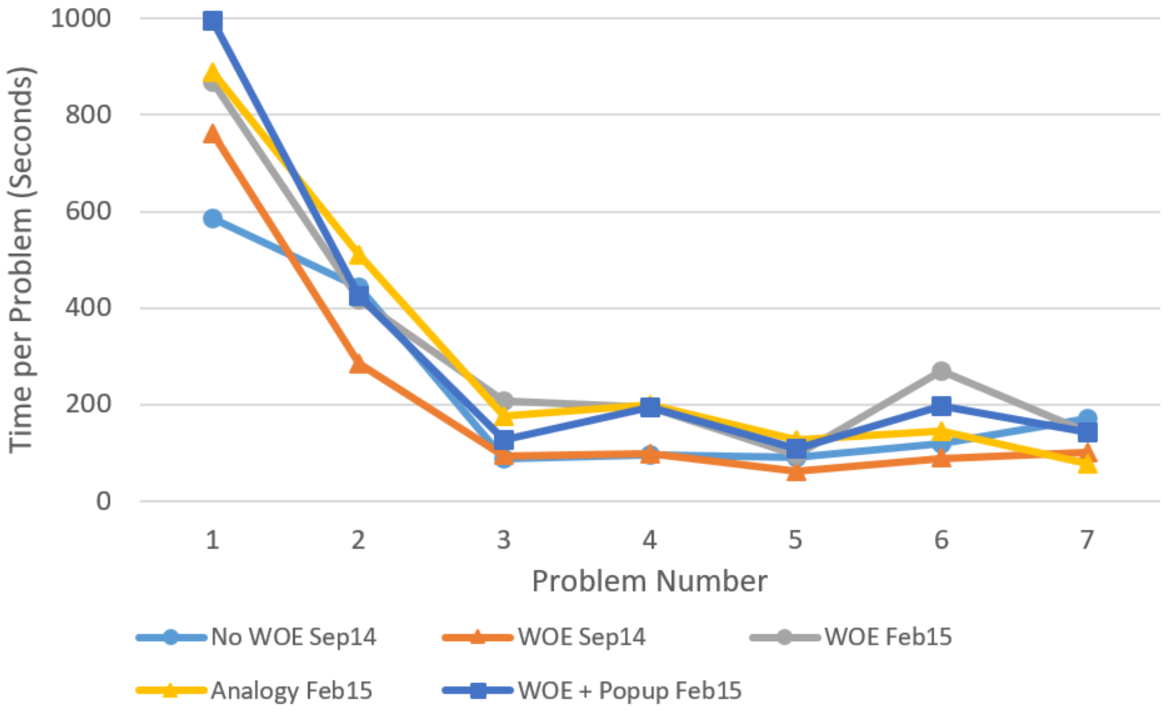


Figure 13: Time Spent on Problems

number of attempts. The Feb15 students spent about 25% longer on the system which may account for the previously analysed time graphs where they spent more time on earlier problems, but solved as many problems as the no-WOE group.

| Feature | Sep14 | | Feb15 | | |
|---|---|---|---|---|---|
| | No WOE | WOE | WOE | Analogy | WOE + Popup |
| Avg. problems solved | 3.923 | 2.698 | 3.92 | 4.0 | 3.88 |
| Avg. problems attempted | 6.487 | 6.628 | 10.36 | 9.33 | 8.96 |
| Avg. unique problems attempted | 4.69 | 3.81 | 4.72 | 4.79 | 4.64 |
| Avg. system usage (in seconds) | 1799.8 | 1702.2 | 2297 | 2261.6 | 2366.8 |

TABLE XV: Logged Statistics for all Groups

### 5.5 Worked-out Example Analysis

Prior analysis showed some significance, mainly due to the poor results achieved from the Sep14 WOE condition. While surprising, and not conforming to other work where WOEs usually bring some value to student growth, it does give some evidence that WOEs may not be a silver bullet. A deeper analysis of the differences between the known positive and negative WOE conditions may shed some light on where the shortfalls, and merits, of WOEs may lie.

Figure 14 shows the breakdown of learning gain between the two WOE conditions (Sep14 WOE, and Feb15 WOE). These show a striking difference between the benefits gained in the two conditions with the Feb15 condition scoring consistently higher than in Sep14. It is interesting to see that the Sep14

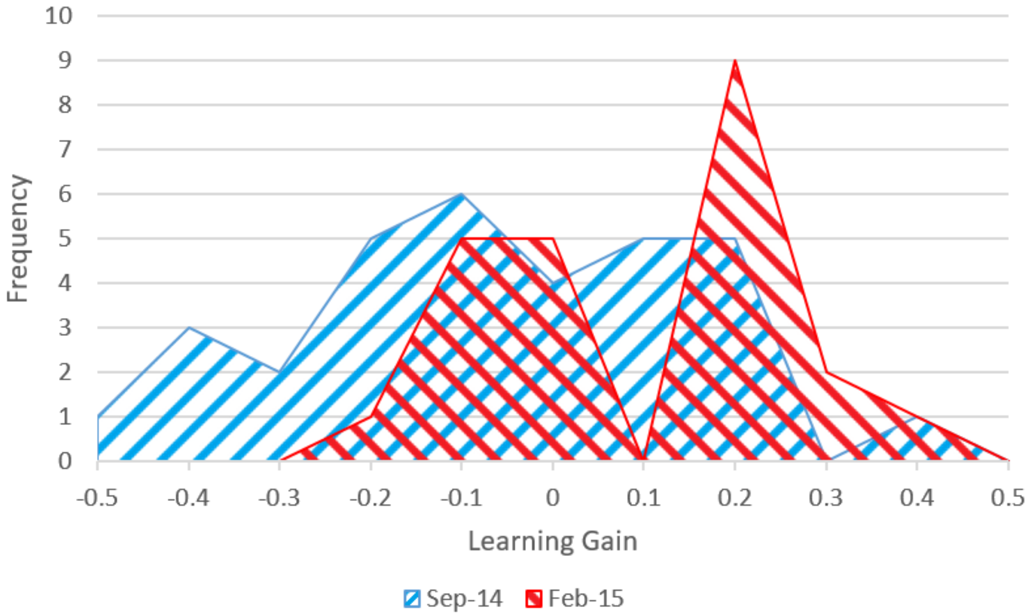


Figure 14: Learning Gain for WOE Students in Sept14 and Feb15 Conditions

condition shows far more variation with no clear common score, whilst the Feb15 condition has half of the students gaining in the region of .2 or -.05.

There are also some interesting variations of WOE usage over the seven problems. Figure 15 shows a breakdown of WOE usage per problem where both conditions show a decline of example usage as problems progress. However, there is a large increase of example usage on problems 6 and 7. This makes sense as problems 6 and 7 differ substantially from prior problems as they require the student to write and execute an entire block of code that manipulates several lists at once, rather than by giving the usual step-by-step list of operations. Also, the latter problems do not include as sophisticated feedback as the prior problems do. Interestingly, there is a very large difference between the WOE retention rate over problems in both conditions, whereby the Feb15 group appears to utilise the feature far more

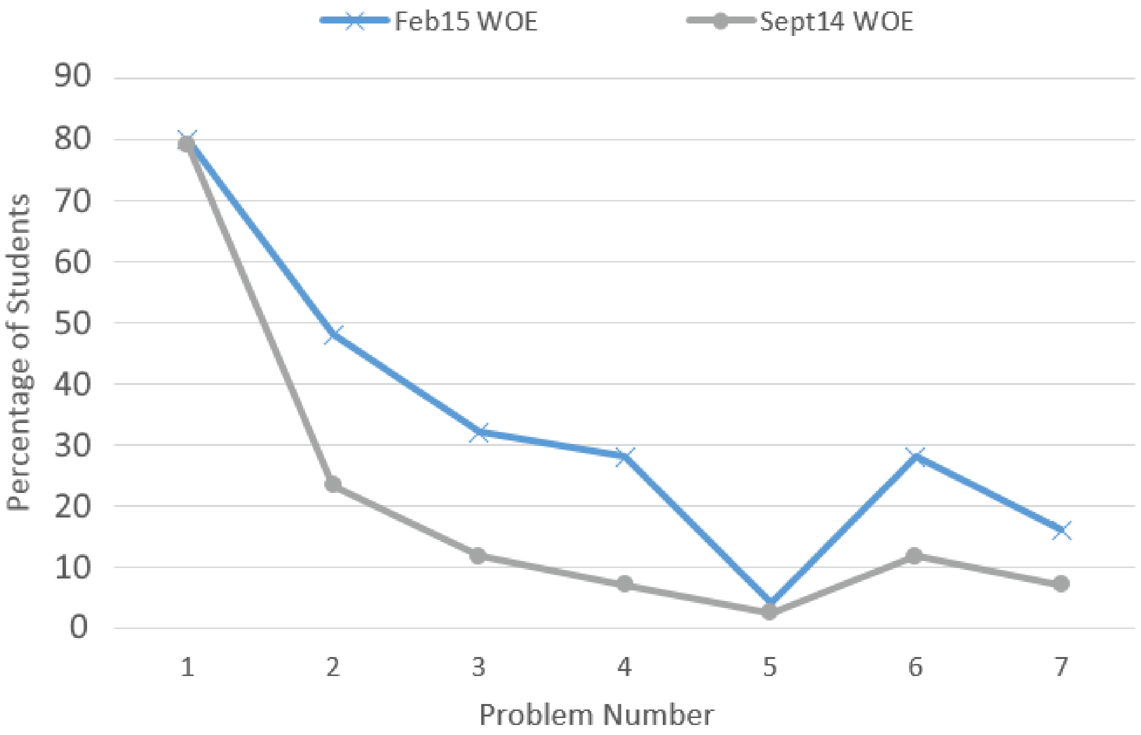


Figure 15: Students using WOEs

than the Sep14 group after the first problem. This may indicate that higher WOE usage in subsequent problems may increase learning gains. The time spent using WOEs (Figure 16) further reinforced the idea that the Feb15 students utilise the feature more-so than Sep14 students by showing that students tend to, on average, spend more time studying WOEs. The similarities here indicate that while more students use WOEs they spend a similar amount of time studying them.

### 5.6 Behavioural Analysis

Next we took a more in-depth look at student behaviour while using examples (Green et al., 2015). Due to the high uptake of WOEs on problem 1, and then the rapid degradation of usage on subsequent

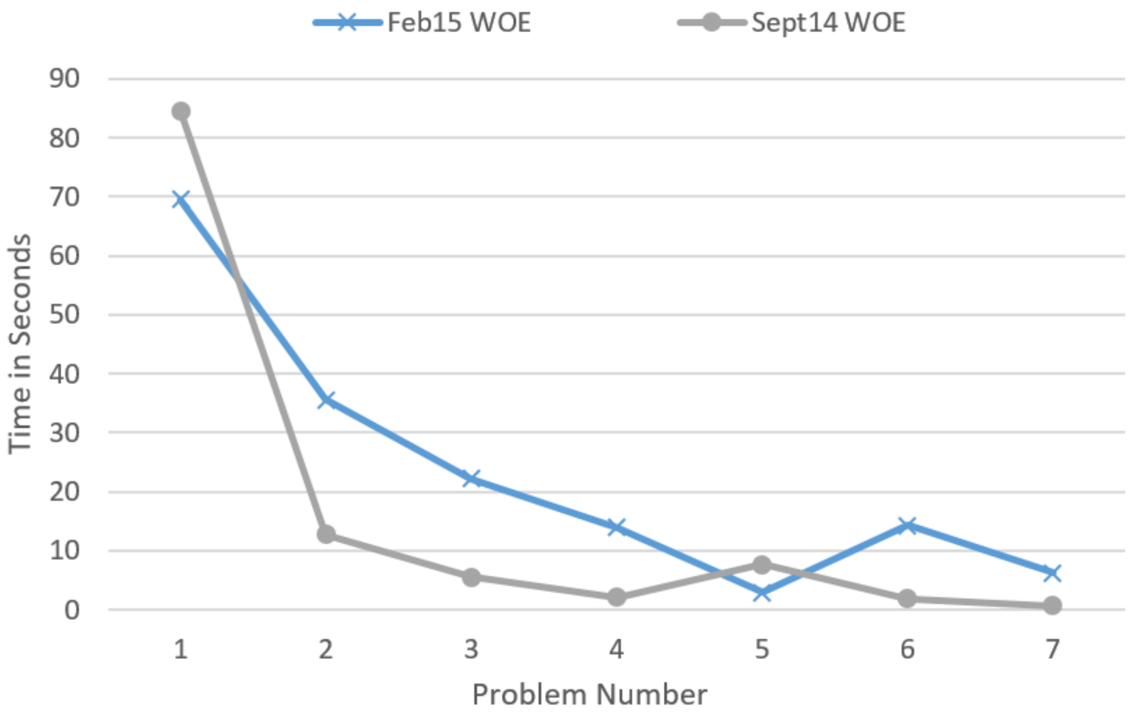


Figure 16: Time Spent Studying WOEs

problems (especially in the Sep14 condition), we chose to focus on examples used exclusively in problem 1 for both Sep14 and Feb15 WOE conditions. We performed behavioural analysis in two ways. Firstly, we created visualisations of student behaviour using timeline plots so we can gain some intuition of potential interesting features in the system logs. From there, we were able to refine our search and use statistical methods to solidify any theories that we may have.

### 5.6.1 Visualisation Analysis

From the log files gathered during the experiments, we extracted several event types for all users (regardless of them using WOEs or not) during the first problem. The events of interest revolve around WOE usage; starting the WOE, stepping through the WOE, terminating it (quitting or completing), as well as important events including the correct or incorrect submission of solutions for problem 1. These had been cleaned and augmented with the students pre/post-test score, and learning gain.

Figure 18 shows a plot of these events for all students in the Feb15 group (described in the key in Figure 17) in a series of timelines. The x-axis represents time through the problem (on a linear scale), and each row contains events associated with an individual student (discrete categorisation with no scale). The ordering of the timelines starts from the highest gainers at the top, to the lowest at the bottom. This graph shows some common themes between users, some regardless of their achieved learning gains. Users who utilised the WOE feature of the ITS tended to follow a pattern of starting a WOE early on in the problem, but would then quickly quit the example and start working on the problem again. However, they would often return to the example again soon after. This may be where students may have only used the example up to the point that they felt they needed to exercise a skill, and then

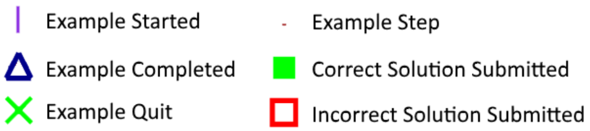


Figure 17: Key for Timelines

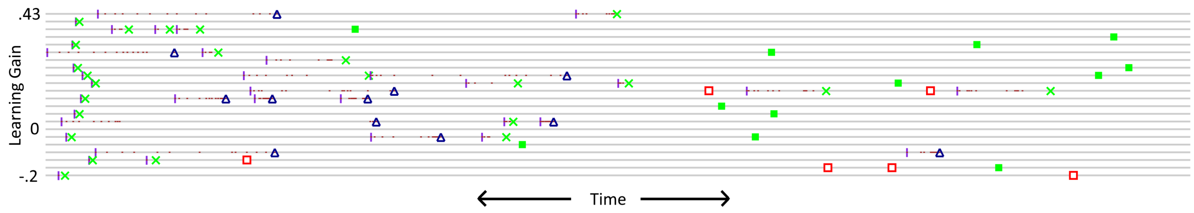


Figure 18: Event Timeline for all Feb15 WOE Users (ordered by learning gain)

returned to the example if they became stuck. Students tended to finish using the example functionality early on, with very few occurrences nearing completion of the problem.

Figure 19 contrasts with that of Figure 18 by showing events from the WOE group in Sep14's experiments. There is clearly different behaviour in this experiment as examples appear to be used, and completed, at all stages of the problem, and there does not appear to be a particular point that students

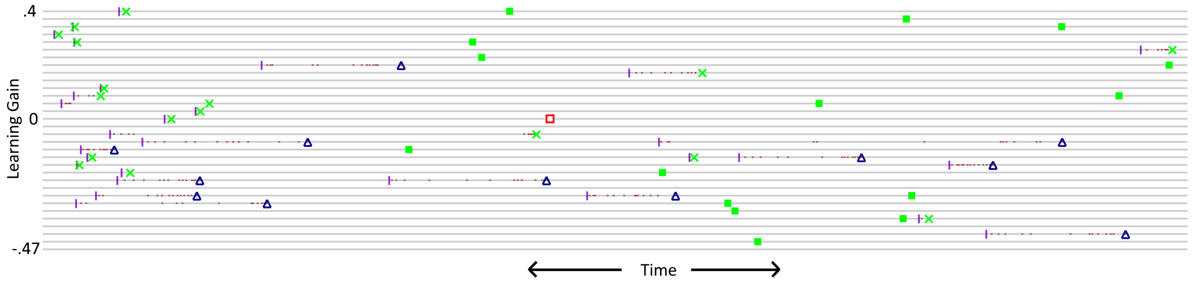


Figure 19: Event Timeline for all Sep14 WOE Users (ordered by learning gain)

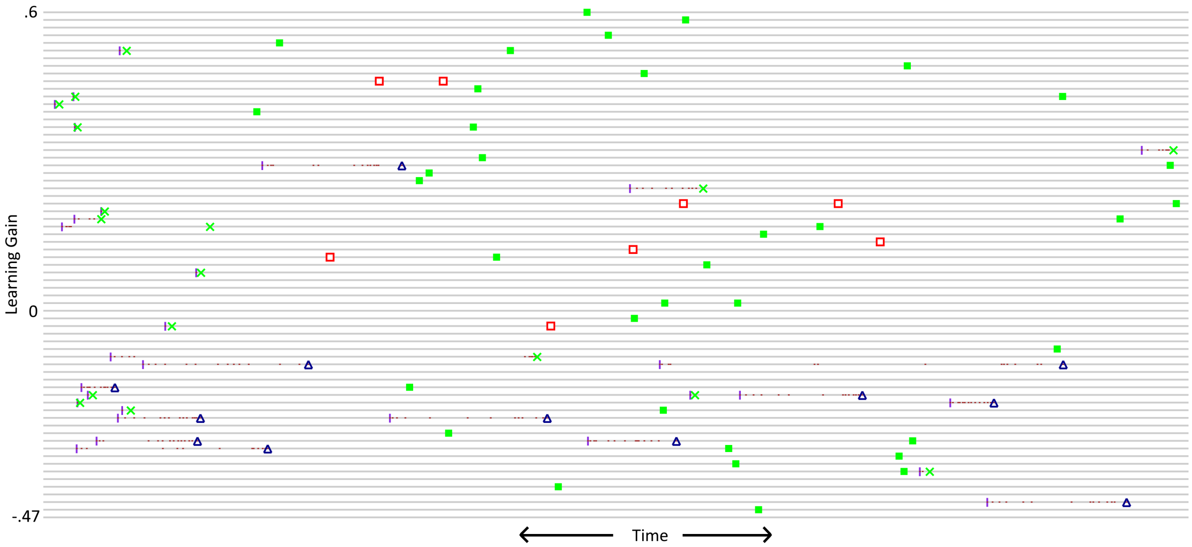


Figure 20: Event Timeline for all Sep14 Users (ordered by learning gain)

tended to use the example feature. The discrepancies here may help in understanding what constitutes good and bad example usage, or may identify students that may need extra structure while learning.

Furthermore, Figure 20 plots **all** students in the Sep14 conditions (Sep14 WOE, and Sep14 No-WOE), once again ordered from high to low gainers. There is a clear pattern relating to WOE usage as all but one of the students that *completed* a WOE ranked in the bottom third in terms of learning gain. Although WOE students in Feb15 benefited from using WOEs in a certain manner, Sep14 students seem not to.

### 5.6.2 Statistical Analysis

The previous time-line plots do give some interesting insights into how students used the tutoring system, and how WOE usage may contribute to learning, or even lack of it. We use some of this insight to perform statistical analysis to gain some objective evidence into the validity of such insights (Green et al., 2016).

Upon visual inspection, it appears that the behaviour of the two groups are different with respect to the data points of starting, completing, quitting, and stepping an example. It appears that the students in the lower gaining group tend to start an example, and follow it through to completion. However, the higher gainers appear to start an example, step through some of it, quit, and maybe return to it later. In this system, the example can be restarted from the beginning at any time. It is important to note that doing so will cause the student to lose any progress they have made on the current problem, therefore they must repeat any steps made on a previous attempt.

Since the graphs, visually at least, indicate a pattern in use, we extracted and derived features from the log data to help us uncover any potential links to learning gain. This way, we may be able to hypothesise whether certain WOE behaviours may be of benefit to students, or if such behaviour may be a useful indicator for student modelling. The extracted features (Green et al., 2016) were:

- **First WOE Completed:** 1 if the first viewing of a WOE ended in completion, 0 if terminated by the user.
- **Completed WOEs:** The number of WOEs that played through to completion.
- **Incomplete WOEs:** The number of WOEs that were terminated by the user.
- **Total WOEs:** The total number of WOEs played (Complete + incomplete).
- **Proportion Completed WOEs:** The proportion of complete vs incomplete WOEs (completed WOEs / total WOEs).

- **Total Duration:** Total duration of all WOEs in seconds.
- **Average WOE Length:** Average length of all played WOEs in seconds (complete and incomplete).
- **Average WOE Steps:** Average number of steps made in a WOE.
- **Standard Deviation Steps Used:** The standard deviation of steps used over all WOEs.
- **Standard Deviation Duration:** The standard deviation of all WOE durations in seconds.
- **All Complete WOEs:** 1 if all WOE played were completed.
- **All Incomplete WOEs:** 1 if all WOE played were incomplete.

Since pre-score has been found to be the most explanatory feature in our previous work (Di Eugenio et al., 2013), we also include this in our analysis. The features used in this analysis are solely from the first problem of the linked list lesson, and only students that used a WOE in this problem were included the analysis (42 students).

So far, we have looked at students in two groups; Sep14 and Feb15. The general view of the visualisations was that the behaviour was different between the two groups. More importantly, however, was that in both cases the behaviour changed among students as their gains increased. The behaviour between the two groups may be more pronounced due to the significant learning gain difference between the two groups. Thus, a more interesting path is to investigate the difference between high and low gaining students, rather than two near equivalent experimental sessions. Therefore all students were aggregated into a single dataset, which was then partitioned into groups of high and low gainers. The dataset was ordered by learning gain, and then divided into two groups.

We performed analysis on several features which were acquired, directly or derived, from the experiment logs. Table XVI shows the average WOE features for both of these sets. Values of statistically significance, as calculated via an unpaired t-test, are highlighted: $^*$ indicates significance ($p < 0.05$), and bold values indicate a trend towards significance ($p < 0.1$).

| Group | Low | High |
|---|---|---|
| First WOE Completed | .381 | .238 |
| Completed WOEs | .952 | .667 |
| Incomplete WOEs | 1.0 | 1.333 |
| Total WOEs | 1.952 | 2.0 |
| Proportion Comp. WOEs | .389 | .206 |
| Total Duration | 123.052 | 71.295 |
| Avg WOE Length | **57.19** | **27.93** |
| Avg WOE Steps | 7.421 | 5.794 |
| Stdev Steps Used | 2.22 | 1.558 |
| Stdev Duration | **29.25** | **12.18** |
| All Complete WOEs | .19 | .048 |
| All Incomplete WOEs | .429 | .667 |
| Learning Gain | **-.079**$^*$ | **.244**$^*$ |

TABLE XVI: Mean WOE Based Feature Counts from Problem 1

In general, at a surface level, none of the tested features were significantly different. However, there were two features which trended towards significance, those being the average WOE length in seconds ($p = 0.0801$), and the standard deviation of WOE durations ($p = 0.0842$). In such cases it seems that higher gainers tend to spend less time on examples that low gainers, as well as high gainers showing far more variation in time spent on examples than high gainers. While not being under the $p < 0.01$

target, it does suggest that such features may be useful in distinguishing between good and bad WOE behaviour.

From here, to get a better sense of what features may contribute to learning gain, we systematically built regression models using subsets of these features where the independent variable was absolute learning gain. Table XVII shows the top three regression models (with respect to adjusted-$R^2$), with a fourth model using a baseline of pre-score alone. From this we see that WOE features do improve the adjusted-$R^2$ over solely using pre-score. The most significant features are ones related to the student quitting the example or letting it run to completion. This centres on ProportionCompletedWOEs, AllCompleteWOEs, and AllIncompleteWOEs, where a negative correlation is shown for completing WOEs, and positive for completing them. This correlation strengthens our prior conclusions suggesting students who prematurely terminate WOEs are higher gainers. The average WOE length feature also appears in two of the three models, which trended towards significance in our t-test analysis.

## 5.7 Summary of Worked-out Example Experimentation

We described our experimentation into integrating worked-out examples into a computer science intelligent tutoring system. Our work was evaluated by carrying out two similar studies in September 2014 and February 2015. Experimentation revealed inconsistencies between the two WOE groups from both experiments, where one performed very well, while the other made no mean learning gains whatsoever.

Furthermore, we looked in-depth at how students used WOEs on problem 1. There were behavioural differences between the two WOE groups; the lower gaining group tended to use examples throughout the problem, whereas the higher gaining group appeared to finish with examples early in the problem. Interestingly, the higher gaining group had double the retention of using examples on subsequent prob-

| | Predictor | $\beta$ | $R^2$ | $P$ |
|---|---|---|---|---|
| Model 1 | Pre-score | $-.518$ | .212 | $< .01$ |
| Model 2 | IncompleteWOEs | .07 | .364 | $= .165$ |
| | ProportionCompletedWOEs | .59 | | $< .1$ |
| | TotalDuration | $-.0004$ | | $= .127$ |
| | AllCompleteWOEs | $-.315$ | | $< .1$ |
| | AllIncompleteWOEs | .357 | | $< .05$ |
| | Pre-score | $-.586$ | | $< .001$ |
| Model 3 | TotalWOEs | .046 | .366 | $= .105$ |
| | AvgWOELength | $-.001$ | | $= .134$ |
| | AllIncompleteWOEs | .175 | | $< .02$ |
| | Pre-score | $-.555$ | | $< .001$ |
| Model 4 | IncompleteWOEs | .054 | .366 | $= .279$ |
| | ProportionCompletedWOEs | .499 | | $< .1$ |
| | AvgWOELength | $-.001$ | | $= .118$ |
| | AllCompleteWOEs | $-.258$ | | $< .1$ |
| | AllIncompleteWOEs | .351 | | $< .05$ |
| | Pre-score | $-.577$ | | $< .001$ |

TABLE XVII: The most Explanatory Experiment Models with WOE Features.

lems. This may indicate that the high gaining group was able to learn effectively given a minimal structure while using the examples feature, while the opposite may apply to low gainers. The importance of this is further emphasised by the fact that examples were not just ineffective on the lower gaining group, but appeared to be damaging with no mean learning gain, and virtually all students that utilised this feature ranked in the bottom third percentile in terms of learning gain when compared with all other students in the Sep14 round of experiments.

The features mined for in the log files do show some potentially interesting correlations between the use of WOEs and learning gain. Averages of these features among various groupings do suggest that students may benefit from partial WOE usage rather than studying them in their entirety. Furthermore, we created some models using multiple linear regression that more-so back this claim up. These features have also been compared using a t-test between high and low gainers, with some features trending towards significance. Thus, it may be advantageous to use WOEs in a CS ITS, but keeping in mind that students may gain more from the ITS if they do not aim to complete all of them, at least not with the content that is given to the users in these experiments.

# CHAPTER 6

# INVESTIGATING PROPERTIES OF WORKED-OUT EXAMPLES

Our research so far has uncovered interesting insights into the usage of worked-out examples in a computer science intelligent tutoring system. Rather than confirming the general belief that worked-out examples aid students in learning, we observed a mix of results that show examples can be of use. This provides evidence for our hypothesis that examples do affect learning. Furthermore, we have identified behavioural traits that may make the learning experience more, or less-so, effective. The next step we took was to expand our analysis of the use of WOEs in our application. The next task is: Discover if there are any properties of worked-out examples, and if so what, may be of use to learners.

## 6.1 Properties

Our research has identified that higher gaining students tended to use a WOE for a short time, and then terminate it prematurely. Other desirable behaviour also includes the restarting of examples at a later stage. From these features, we identified two potential properties of WOEs that may affect their usefulness.

### 6.1.1 Regulation of Example Duration

It appears that students who exhibit one type of behaviour (frequent starting and stopping of examples) achieve higher learning gains over others (full example execution). Furthermore, higher gaining students appear to spend less time on an individual example over their counterparts. Perhaps, this behaviour is conducive to learning and is something that should be encouraged of students. Employing regulation on example usage to promote this ’good’ behaviour may show if forcing students to use particular behaviour is beneficial to learning.

### 6.1.2 Variable Length Examples

Current experiments have included two types of WOEs: standard, and analogies. The results of these do differ, however it is not known if the impact is through the use of analogies or with differing content. One possible variable feature could be the length of the example. Examples used in the discussed experiments have been fairly verbose, with additional text that does not add to the core information being conveyed. For example, a verbose step may be “What we will do now is create a new variable, and lets assign the value of ‘3’ to it.”, while a more concise step may be “Create a variable with value 3”. Additional experiments with examples of different length may give additional insight into how WOE content could affect learning.

## 6.2 Additional Experimentation

Some avenues of investigation required additional data, which led to the need for further experimentation. This also meant that some additional features were required in ChiQat-Tutor. The two lines of research that required new features are ones that look at variable length examples, and look at the regulation of examples.

To investigate the role of example length in an ITS (6.1.2), we introduced a new set of examples called *Short WOEs*. These examples are based on the original WOEs, but are substantially more concise. The key with these short examples is that they are to convey the same core information as the original examples, but remove superfluous language and example steps. For example, the WOE for problem 1 includes a reflection step, whereby an error is introduced into the example, and the tutor then corrects it with an explanation for the error. Details such as this were removed from Short WOEs. A Short WOE was created for each problem in the same manner. During experimentation, each registered user was assigned to either a Short or Standard WOE, thus a student will only ever see the same type of example during their time using ChiQat-Tutor. A comparison of Standard and Short WOEs is detailed in Table XVIII, with the full standard example for problem 1 shown in Table XIX, and the corresponding short example in Table XX.

| Problem | Feature | Standard WOE | Short WOE |
|---|---|---|---|
| 1 | Number of Steps | 14 | 6 |
| | Number of Words | 212 | 56 |
| 2 | Number of Steps | 12 | 4 |
| | Number of Words | 173 | 56 |
| 3 | Number of Steps | 12 | 4 |
| | Number of Words | 192 | 50 |
| 4 | Number of Steps | 11 | 4 |
| | Number of Words | 164 | 64 |
| 5 | Number of Steps | 10 | 5 |
| | Number of Words | 132 | 64 |
| 6 | Number of Steps | 10 | 5 |
| | Number of Words | 128 | 92 |
| 7 | Number of Steps | 11 | 4 |
| | Number of Words | 167 | 93 |

TABLE XVIII: Comparison of Standard and Short Examples

In addition to the newly created WOEs, all WOE steps were annotated for their step type. This may be useful in analysis as the Short WOEs have been abbreviated and omit some types of steps, such as pausing/reflection steps. Annotations include:

| |
|---|
| OK, (USERNAME), we’re going to take a look at how we can insert items into a linked list |
| Take a look at this list |
| What we want to do is to insert a ’3’ inbetween ’2’ and ’4’ |
| The very first step is simple, lets create a new node with a value of ’3’, and why don’t we call it ’Z’ |
| Now we need to insert a node AFTER ’2’, the one that’s flashing |
| Firstly, we should get access to the second node. This can be done by going through the root, T, - and getting its next node. This can be assigned to a variable, S |
| From here, we could assign 2’s next pointer to the node containing 3, like so |
| However, there is a problem here, think about it... |
| How do you reattach the node containing 4? The connection now has been lost! |
| Lets take a step back and see how we can do this without losing this vital connection |
| Lets connect the node containing 3 to the node containing 4 |
| Now lets do what we done before and connect 2 to 3 |
| and then tidy up some of the references, this being S and Z |
| And there we go, 3 has been inserted into the list between 2 and 4! |

TABLE XIX: Standard Worked-out Example for Problem 1

| |
|---|
| Here is how you insert ’3’ inbetween ’2’ and ’4’ |
| First create a node ’3’, pointed to by ’Z’ |
| Next we need to find the node ’2’ and assign it to a new variable, S |
| Connect ’3’ to the ’4’ |
| and then ’2’ to ’3’ |
| Remove variables that are not needed, and we are done |

TABLE XX: Short Worked-out Example for Problem 1

- Intro - Introduces the example.
- Definition - Outlines the problem to be tackled.
- Operation - Performs an operation towards the solution.
- Explanation - Elaborates on a step.
- Pause - A reflection step.
- Conclusion - Concluding steps of the example.

Experimenting with this new type of WOE may shed some light on how varying example length can affect learning by comparing with the standard WOE experiments previously conducted. Annotating both steps in the Standard and Short WOEs will also give us some additional insight into how the construction of such WOEs may also affect learning.

The investigation into regulation of examples is another task that required additional features (6.1.1). In this investigation, we focused on the termination of WOEs. There are two angles that we looked at: automated termination of examples, and the inability of students to prematurely terminate an example. Although both of these appear to control the same feature, they look at termination from two different angles, which makes both methods of regulation worth exploring. Since completing examples in their entirety appears to reduce learning, we hypothesise that automated termination may yield higher learning gains than forcing WOE completion. If this is true, it may add additional evidence to this assertion and give some insight into the question whether restarting of examples affects learning.

WOEs in ChiQat-Tutor had been enhanced to allow both of these methods of regulation to be specified on each example. While it is quite straightforward to set a flag stating that early termination of WOEs is not possible by disabling any button that could lead to a WOE termination, automated termination requires knowing when to terminate an example. Prior analysis showed that students who spent less time on WOEs, on average, performed better than other students. Therefore, we added a time-out attribute on each example, whereby the example will terminate after a specified number of seconds. Each example has a custom time-out point that was derived from the experiment logs that we had previously collected. This figure was calculated using the median of high learners' WOE durations.

Additional experiments were conducted as before, along with three further conditions: Short WOE, Auto Exit WOE, and No Exit WOE. These experiments were performed over several classes - a breakdown of each condition and experiment is given in Table XXII, which indicates the number of subjects in each condition. The seven experiments were conducted over two different courses at UIC, the original CS211 course (Programming Practicum) and CS201 (Discrete Maths and Data Structures). CS201 is not a required class for computer science majors, but does have requisites of MATH180 (Calculus 1) and an introductory class in computer programming (CS102 or CS107). Experiments in the CS201 class took place at various points in the semester. However, experiments always took place after the topic of linked lists had been introduced - this is the same as with experiments conducted in CS211. A mapping of experiments to courses are shown in Table XXI.

| Date | Course |
|---|---|
| Sep-14 | CS211 |
| Dec-14 | CS201 |
| Feb-15 | CS211 |
| Apr-15 | CS201 |
| 17th Sep-15 | CS211 |
| 27th Sep-15 | CS201 |
| Feb-16 | CS211 |

TABLE XXI: Courses used for each Experiment

Since we had over 300 individual samples in total after these experiments, all sample groups were reorganised to remove any potentially outlying samples. Examples of an outlier are students who had multiple session samples (e.g. restarting the system half way through an experiment), or a student who posted very little activity in the system (e.g. starting and exiting the system). This was not undertaken as aggressively as in previous analysis due to having a small sample size for the first two experiments.

The sample count in Table XXII does vary between experiments. Firstly, the April-15 experiment only contains a total of eight students. Although the class was larger than this, approximately 25 students, many of the them had already used the system in a previous class. These student were therefore eliminated from analysis. Also, there is a single student in the No WOE condition for the experiment on the 17th September 2015. The aim of this experiment was to test the WOE and Short WOE conditions, therefore each student who registered for an account on that day was assigned to either one of these conditions. This lone No WOE user had created an account previously without having used the system. It is deemed that this student is still a valid sample for the No WOE condition and therefore kept.

| Date | Num Samples | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|
| | No WOE | WOE | Short WOE | No Exit WOE | Time Out WOE | Pop-up Analogy | Analogy | |
| Sep-14 | 32 | 28 | | | | | | 60 |
| Dec-14 | 16 | 11 | | | | | | 27 |
| Feb-15 | 8 | 27 | | | | 21 | 19 | 75 |
| Apr-15 | | 3 | 5 | | | | | 8 |
| 17th Sep-15 | 1 | 31 | 23 | | | | | 55 |
| 24th Sep-15 | | 18 | 17 | | | | | 35 |
| Feb-16 | | | | 28 | 32 | | | 60 |
| Total | 57 | 118 | 45 | 28 | 32 | 21 | 19 | 320 |

TABLE XXII: Count of Samples over all Conditions

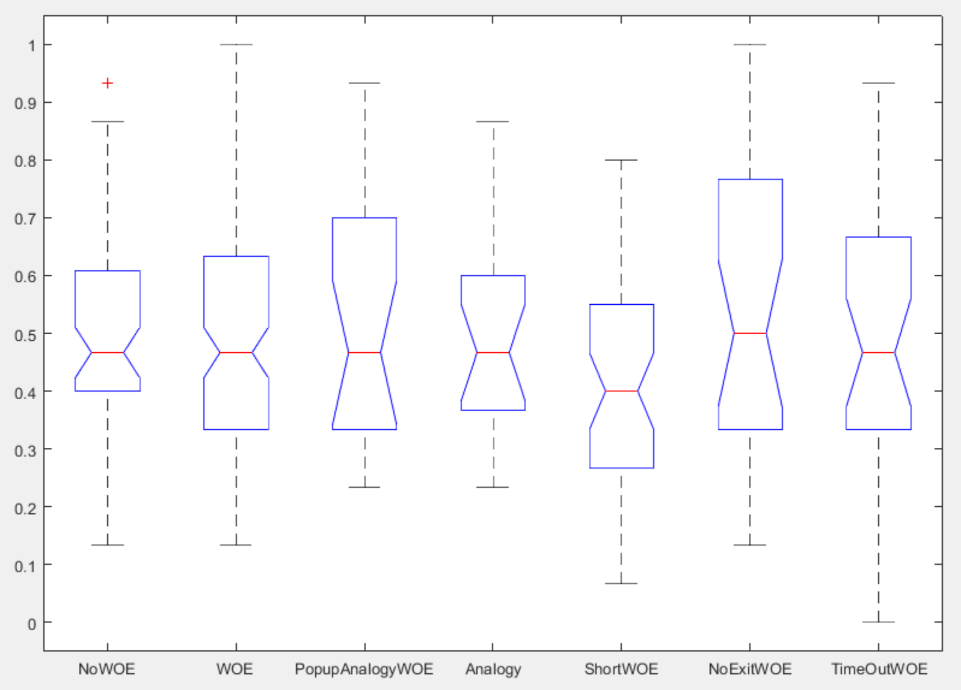


Figure 21: ANOVA for all Pre-Test Scores by Experiment Condition

An ANOVA was run on a condition basis for student pre-test scores, which showed no significant differences between groups with $[F(6, 313) = 1.16, p = .3265]$ (Figure 21 and Figure 22).

### 6.3 Learning Gain Analysis

As with prior experiments, all participating students completed a pre and post-test with their ChiQat-Tutor session being used as the intervention. All tests were graded, with each student receiving a learning gain score based on absolute learning gain. Table XXIII shows a cumulative learning gain table for all experiments conducted. Note that the learning gain for previously analysed experiments does differ here due to the additional removal of outliers.

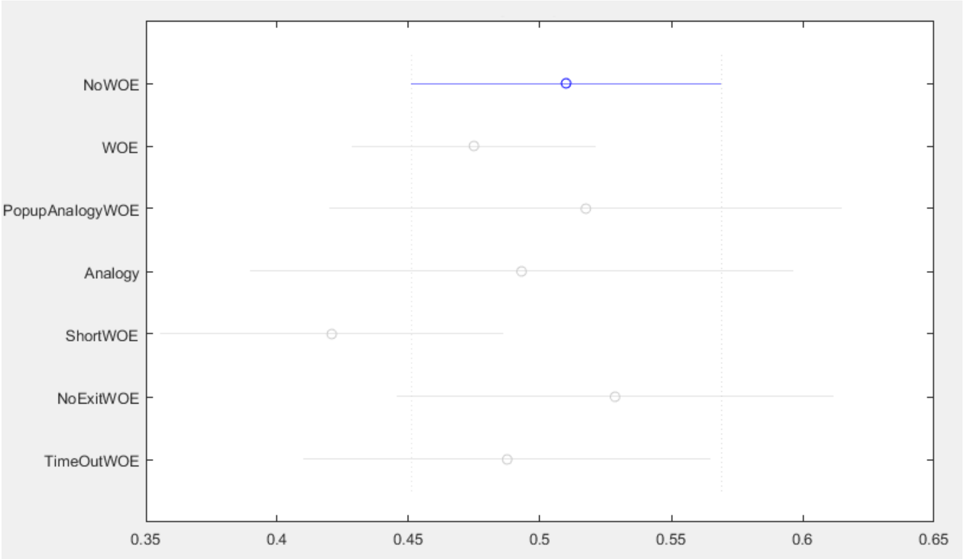


Figure 22: Tukey Post-Hoc Analysis on all Pre-Test Score by Experiment Condition

Table XXIII suggests students learnt during the exercise with the mean of all conditions showing growth. Even though there does appear to be some variation in the means for each condition, an ANOVA using each condition's sample learning gain (Figure 23) does not show any significant differences. Post-hoc analysis (Figure 24) reports $[F(6, 313) = 1.111, p = .3556]$.

Thus, the learning gain alone for each condition does not give any interesting insight into the use of the various WOE conditions that have been explored, given our analysis.

## 6.4 Log Analysis

As in prior experimentation, full logs were collected from a student's use of the system. Logs had been processed and expanded upon by creating derived features. In earlier analysis, we had looked at feature changes over all problems. Here, we once again look at these differences in more depth, and

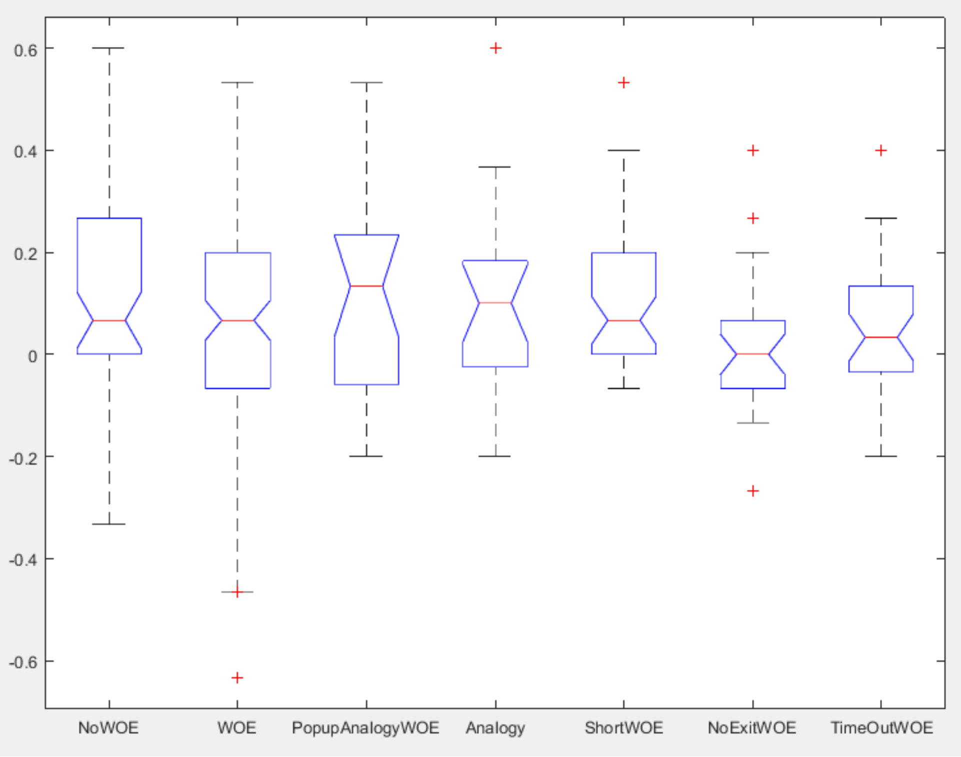


Figure 23: ANOVA on Learning Gain for all Conditions

| Date | Gain per Condition | | | | | | |
|---|---|---|---|---|---|---|---|
| | No WOE | WOE | Short WOE | No Exit WOE | Time Out WOE | Pop-up Analogy | Analogy |
| Sep-14 | .0667 | -.0429 | | | | | |
| Dec-14 | .1375 | .2424 | | | | | |
| Feb-15 | .1625 | .1074 | | | | .1064 | .1053 |
| Apr-15 | | 0 | .1467 | | | | |
| 17th Sep-15 | 0 | .0753 | .1333 | | | | |
| 24th Sep-15 | | .1519 | .1059 | | | | |
| Feb-16 | | | | .0238 | .0542 | | |
| All | .0988 | .0799 | .1244 | .0238 | .0542 | .1064 | .1053 |

TABLE XXIII: Overall Learning Gain per Group and Condition

also among the various conditions. There are two perspectives that we looked at: by condition, and additionally the difference between high and low gaining students in the standard WOE condition.

### 6.4.1 All Conditions

Firstly, Figure 25 shows the problem's completion rate for each condition. The data points here are not cumulative - a student who does not even attempt a problem will be considered a 'did not complete'. In prior analysis, it appeared that students who completed more problems are more likely to grow on the post-test. This is not the case here when comparing the Short WOE and No Exit WOE conditions. Figure 25 shows that students in the Short WOE condition consistently complete fewer problems than the No Exit WOE condition. This would typically imply that students in the No Exit WOE condition would learn more. However, Table XXIII shows the opposite where students in the Short WOE condition learnt more (gain = .1244) than those in the No Exit WOE condition (gain = .0238). This difference between problem completion is also significantly different - problem 3 has a 69.6% and 95.7% success rate in their respective conditions ($\chi^2$, $p < 0.005$).

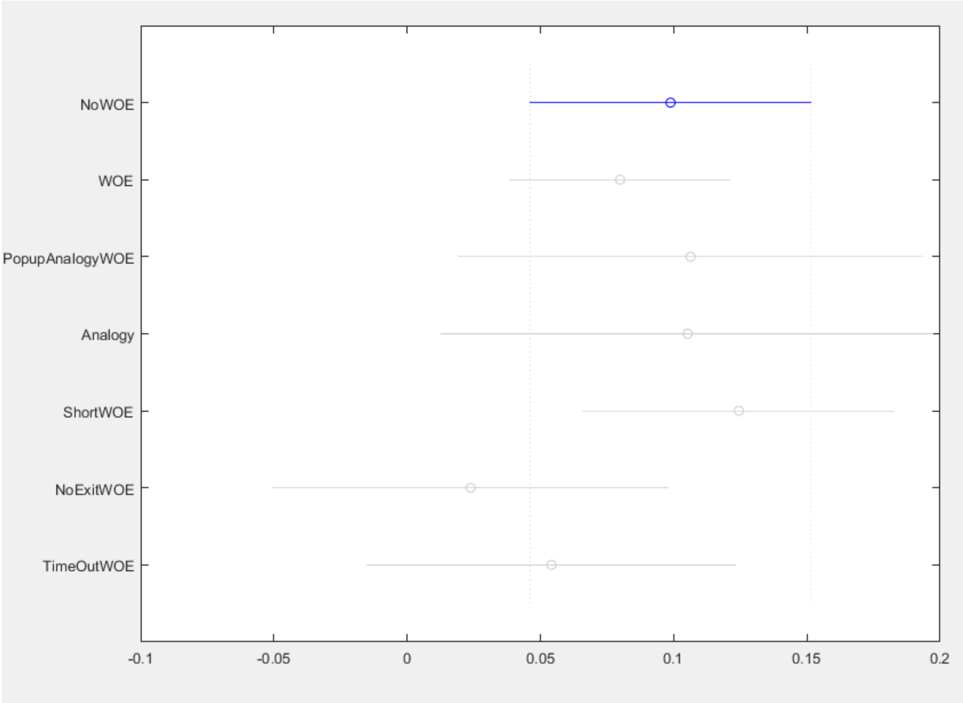


Figure 24: Tukey Post-Hoc Analysis on all Learning Gain ANOVA

Another way to look at the logs is through the mean time spent on each problem. This only takes into account students that actually attempted a problem. Figure 26 plots the amount of time (in seconds) that students spent on each problem. There is some difference on the initial problem, as hypothesised from our knowledge in previous experiments. To counter that hypothesis the amount of time spent per problem appears to normalise somewhat on subsequent problems, barring any inconsistent spikes during the Analogy and No Exit WOE conditions. This may suggest that the first problem may uncover behaviour that cannot be seen on subsequent problems. It could be the style of problem, or perhaps simply because it is the first problem encountered in the system.

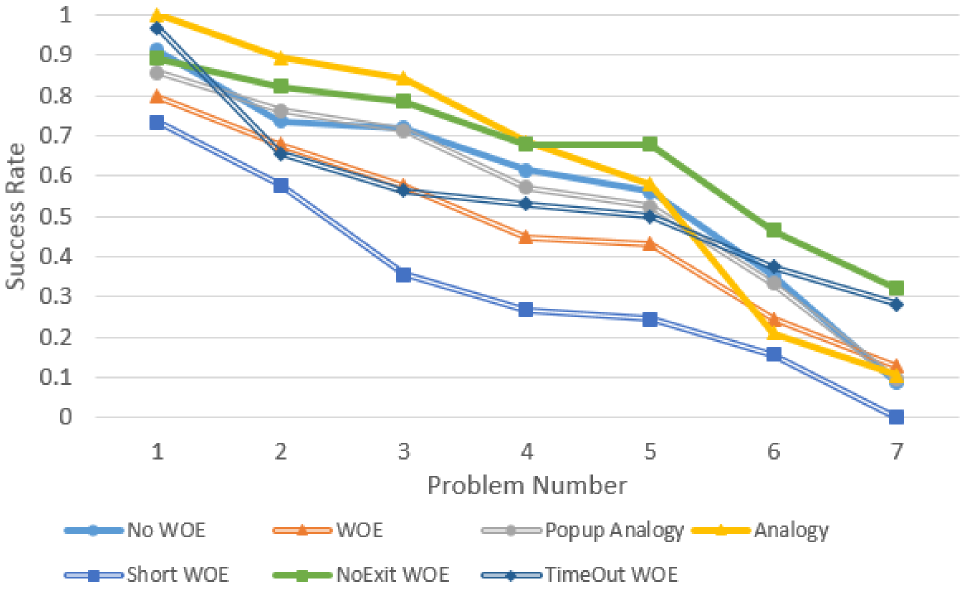


Figure 25: Problem Success Rate (by Condition)

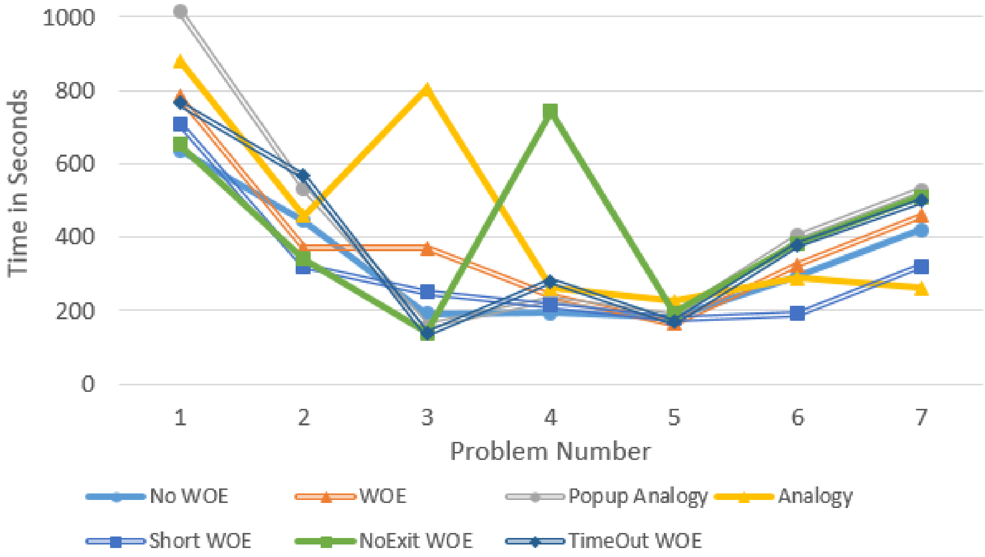


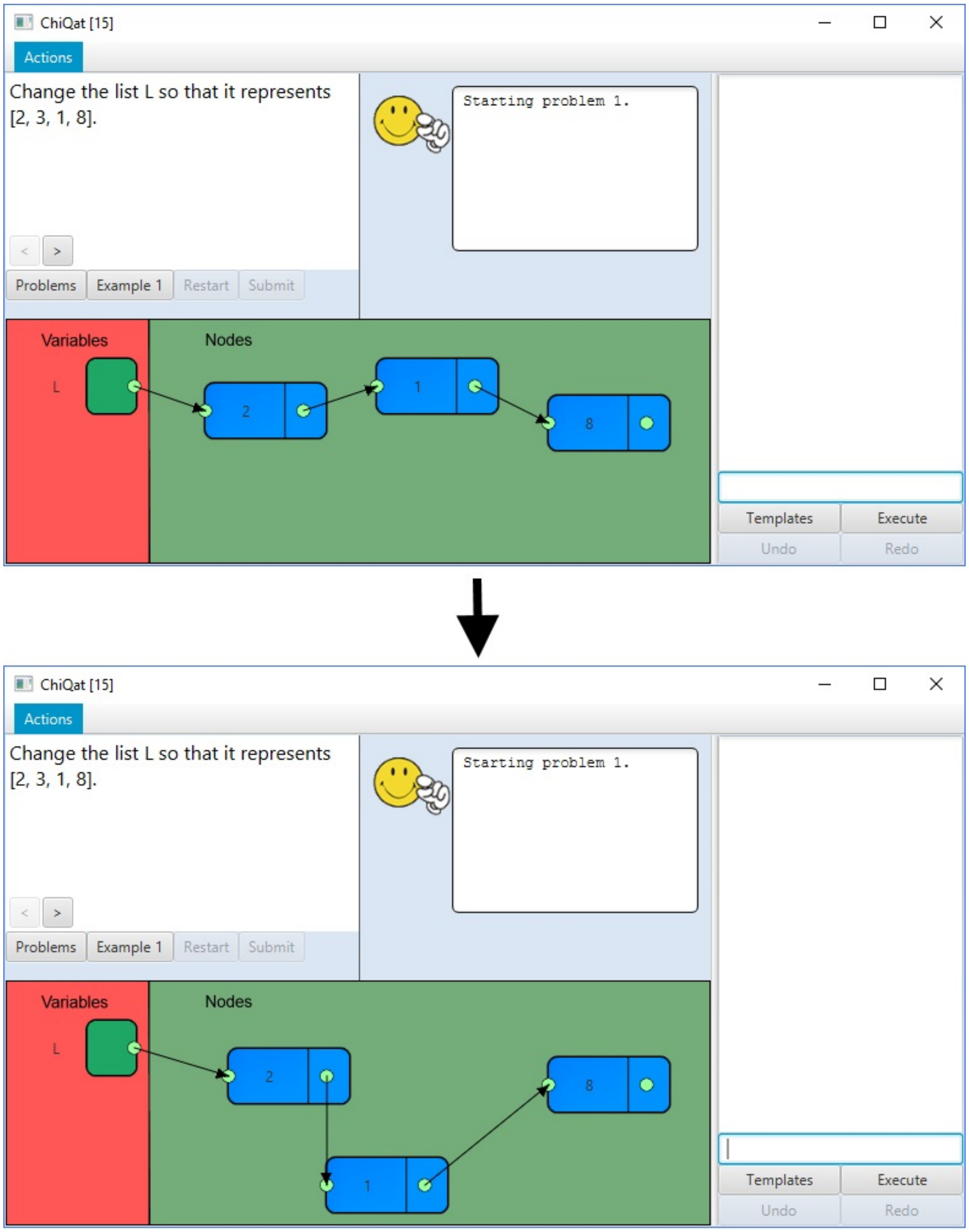
Figure 26: Mean Time Spent on Problem, Grouped by Condition

Finally, another interesting trend is that of the number of clicks that occur on nodes in the user interface. These are the nodes in the graphical user interface that represent variables and linked list nodes that can be moved around the interface via drag and drop (Figure 27). Similarly to the prior assertion of problem 1 being a special case in behaviour, a similar, albeit more extreme, trend can be seen in Figure 28. The number of clicks on anything after problem 2 is very similar, however, problem 1 does show a lot of variation, with Analogy based conditions scoring at around 24.5 clicks on average per student, while Short WOEs record only 10.2 clicks. Three out of the four remaining conditions are similar.

It is unclear why this may be the case. Two possible theories is that the use of analogies could increase interactivity with the system, prompting students to move nodes around more. Alternatively, there may be some link between the number of steps in a WOE and the number of student node movements made. To shed some light on this, linear regression models were built using permutations of WOE features against the number of node clicks performed on problem 1 (Table XXIV). The top three regression models do show that two of the 16 features used occur in all three models. Furthermore, 98.9% of the top 1000 models included the 'First WOE Complete' feature, and 95.8% included the 'StDev WOE Duration' feature. Interestingly, the First WOE Complete feature is negatively correlated with node clicks, meaning that students who complete the first WOE do not click on the nodes often, perhaps linking node clicks to the engagement level of the student.

### 6.4.2 High and Low Gainers

Aside from looking at problem trends over conditions, another interesting aspect is by looking at log differences between high and low gaining students in the Standard WOE condition. Following from



Change the list L so that it represents [2, 3, 1, 8].

```
Starting problem 1.
```

Change the list L so that it represents [2, 3, 1, 8].

```
Starting problem 1.
```

Figure 27: Moving a Node in ChiQat-Tutor

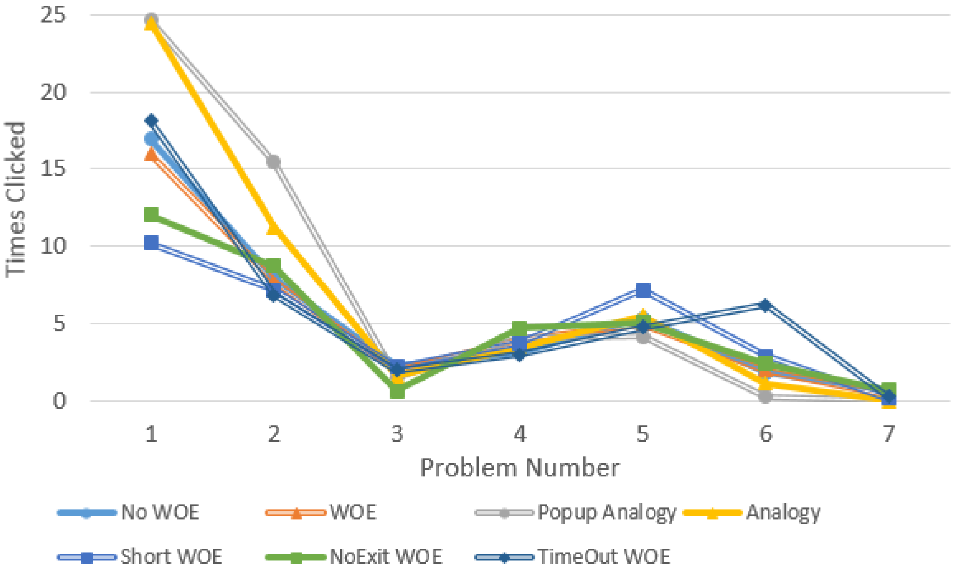


Figure 28: Mean Number of Nodes Clicked on the User Interface per Problem, Grouped by Condition

our analysis of all conditions, Figure 29 shows the success rate of high and low gaining students over all seven problems. Contrary to what we have just seen, high gainers consistently complete more problems than low gainers, up to a point - the two groups do converge eventually on problem 7. It is unclear why this is the case, but may include a small number of low gainers who are cheating and not attempting to complete as many problems as possible rather than learning the material

Moving onto WOE based features, it appears that the number of WOEs requested per problem does not differ much after the first problem (Figure 30). Low gainers do originally request slightly more, but this appears to level out. This goes against the hypothesis that was made earlier that high gainers use WOEs partially and frequently.

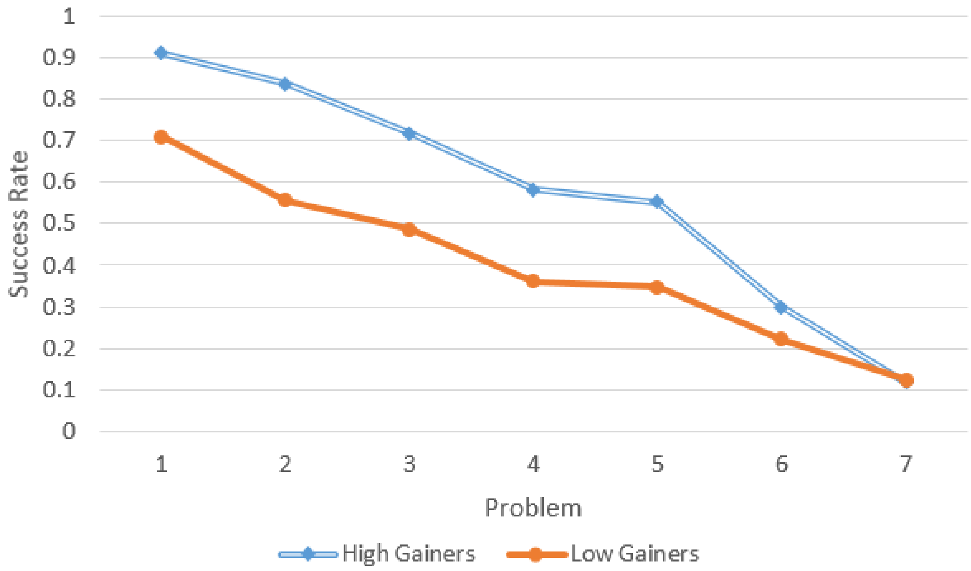


Figure 29: Problem Success Rate for Standard Worked-out Example Students

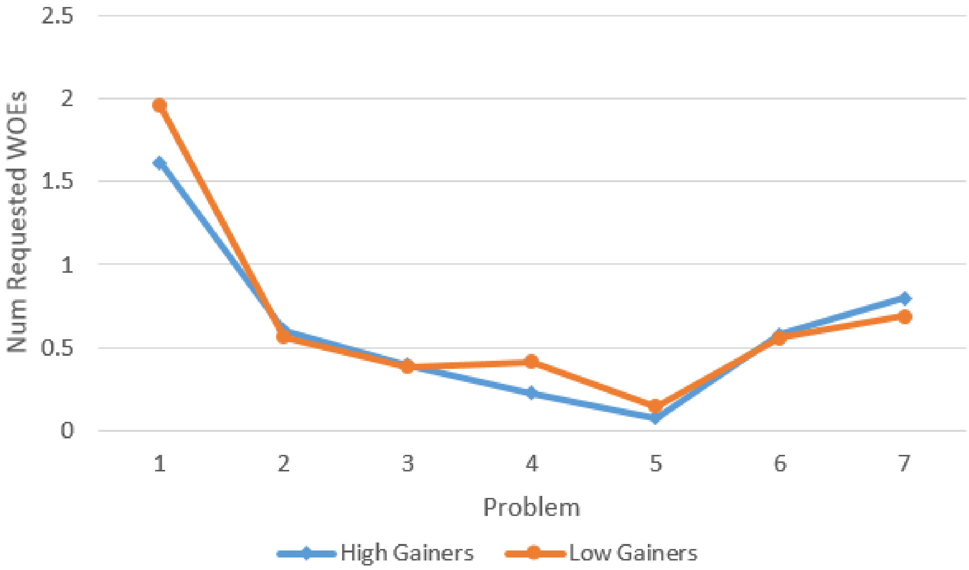


Figure 30: Mean Number of Worked-out Examples Requested per Problem

| Predictor | $\beta$ | $R^2$ | $P$ |
|---|---|---|---|
| All WOEs Complete | $4.7106$ | .0324 | $ns$ |
| Avg Step Duration | $-.6417$ | | $< .02$ |
| First WOE Complete | $-6.5036$ | | $< .1$ |
| StDev WOE Duration | .08 | | $< .05$ |
| Avg WOE Steps | .3397 | .032 | $ns$ |
| First WOE Complete | $-5.9146$ | | $< .1$ |
| StDev Step Duration | $-.4583$ | | $< .02$ |
| StDev WOE Duration | .0692 | | $< .1$ |
| Avg Step Duration | $-.6071$ | .0316 | $< .02$ |
| Avg WOE Steps | .3373 | | $ns$ |
| First WOE Complete | $-5.7012$ | | $< .1$ |
| StDev WOE Duration | .059 | | $ns$ |

TABLE XXIV: The most Explanatory Models Correlating with Worked-out Example Node Clicks

In terms of the hypothesis of high gainers using WOEs partially yet regularly, two of the other metrics are the amount of time spent using WOEs (per step and whole WOE), which are detailed in Figure 31 and Figure 32. The partial, yet frequent, line of thought does show itself slightly when looking at the number of steps taken, with high gainers taking on average fewer steps than low gainers, with the exception of problems 3 and 4. The average duration of examples does not bring into light any other striking observations either, with what looks like noise between two conditions with the same result. There is a large spike on problem 5 for the low gaining group, which may be due to sampling error as this data-point represents one single student. This is surprising since all three of these WOE features were prominent in our prior analysis. The only intuition we have here is that the first problem appears to be fairly unique in telling us the difference between high and low gainers, but other problems offer little insight.

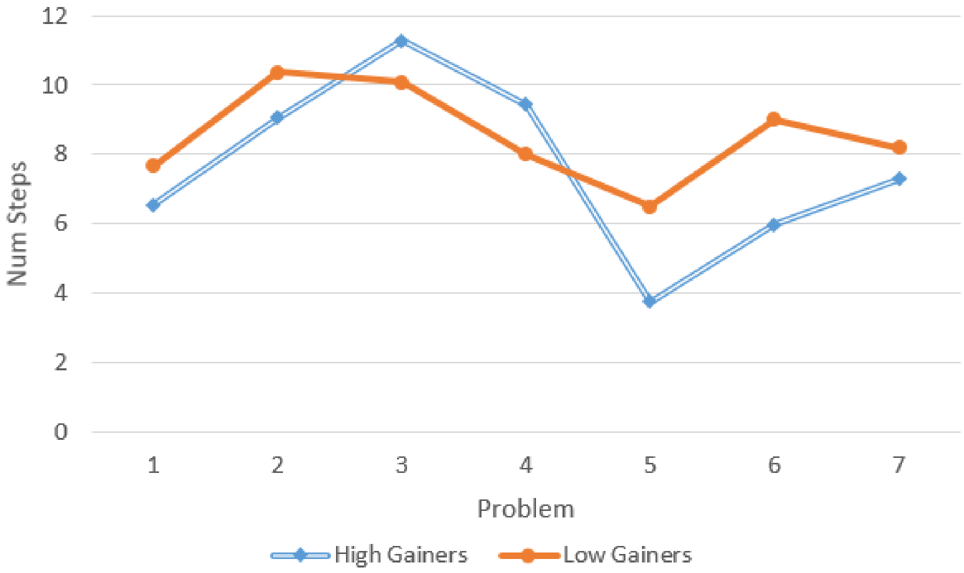


Figure 31: Mean Number of Steps Taken in a Worked-out Example

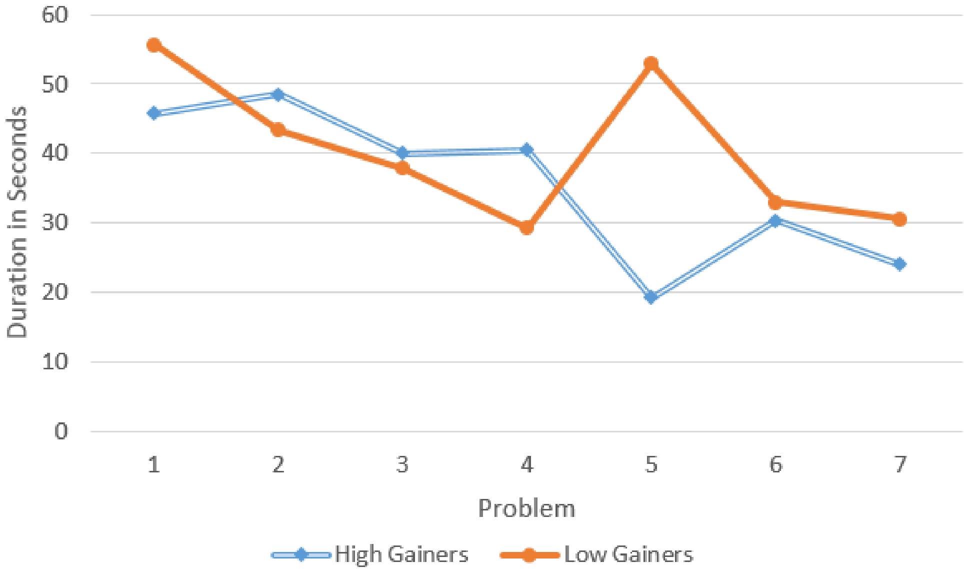


Figure 32: Mean Worked-out Example Duration in Seconds

## 6.5 Other Aspects of Student Knowledge

The majority of work so far has centred around learning gain - do students learn more or less in various conditions. Another aspect that we have looked at was the concept of high/low gainers as a binary classification. In this section, we take a detour and look at other ways to view student knowledge. This overview is based only on standard WOE samples.

### 6.5.1 Initial Student Knowledge

An aspect of student knowledge that has not been covered so far is initial student knowledge - that of a student being a *beginner* or *advanced* student at the start of the exercise. This distinction may be important since pre-test scores (a good measure of existing knowledge) is a useful feature in regression analysis. Also, prior literature has shown that beginners tend to benefit more from WOEs than advanced students. (Najar et al., 2016) uses a similar concept of novice and advanced students to draw conclusions that novice students become more time efficient with WOEs, yet advanced students learn more. Therefore, we introduce the concept of a beginner and advanced student into our analysis.

Firstly, a beginner student is defined as a student who scored in the bottom half of all students in the pre-test, while advanced students are all other students. Due to the coarse grain grading system that we had in the pre-test (0 to 15 in increments of 1) multiple students gained a median score. Therefore, the boundary of the two groups of students was shifted to a well defined boundary; thus, there is not an equal number of students in both categories.

Table XXV shows a matrix of initial knowledge vs learning gains for all students (covers all conditions). It was originally hypothesised that high gainers would mostly consist of students who were beginners since they had much more potential to grow. Although the data does show a skew towards that hypothesis, the lean towards that particular group is not as pronounced as originally thought. Although it is expected that beginners are high achievers, surprisingly 22% of advanced students were also deemed high gainers, showing that such an ITS is not just for beginning students, but also for students with some experience of the material. This in itself is an interesting observation, even though not related directly to WOEs at this point, this does show that advanced students do also benefit from the exercise.

| | Count | | Percentage | |
|---|---|---|---|---|
| | Low Gainer | High Gainer | Low Gainer | High Gainer |
| Init Beginner | 64 | 88 | 20% | 27.5% |
| Init Advanced | 98 | 70 | 30.6% | 21.9% |

TABLE XXV: Initial Knowledge vs Gains

### 6.5.2 Resulting Student Knowledge

Another metric is looking at the end knowledge state of a student. This differs from learning gain as gain shows how much a student has grown after intervention. End knowledge is directly related to the post score achieved by a student, classifying a student into a beginner or advanced student at the end of the exercise. This term is used loosely as one that denotes a student in the bottom or top half of scores achieved in the post test. Once again, due to multiple median scores, students were unevenly distributed between the two classifications.

Table XXVI shows the matrix of all students regarding their beginning and end states. Interestingly, although not as high a proportion, many beginners did make the transition to advanced students after the exercise. This suggests that not all students will make relatively equal gains and that some students will ultimately learn much more, or less, than others.

| | Count | | Percentage | |
|---|---|---|---|---|
| | End Beginner | End Advanced | End Beginner | End Advanced |
| Init Beginner | 109 | 43 | 34% | 13.4% |
| Init Advanced | 55 | 113 | 17.2% | 35.3% |

TABLE XXVI: Initial Knowledge vs End Knowledge

Such a metric suggests that the ITS in its current incarnation is of potential benefit to all students, not just based on their initial level of knowledge. If the goals of the ITS were to target specific groups of, e.g. beginners, such a matrix could be used to evaluate the benefit to such groups after modifications had been made to the ITS.

## 6.6 Analysis of Variable Length Worked-out Examples

Learning gain analysis so far has yielded no significant differences in learning gain for different types of worked-out examples (Table XXIII). Even though there are differences between the mean learning gains for each condition, this does not provide any evidence for our hypothesis that worked-out example length may be a factor in learning.

Prior literature has shown that WOEs work differently for students based on numerous factors, one of which is prior knowledge. Beginners are thought to learn more from WOEs than advanced students.

Therefore, we partitioned our learning gain table into two, one for beginner (Table XXVII), and one for advanced (Table XXVIII) students.

| Date | Gain per Condition | | | | | | |
|---|---|---|---|---|---|---|---|
| | No WOE | WOE | Short WOE | No Exit WOE | Time Out WOE | Pop-up Analogy | Analogy |
| Sep-14 | .1692 | .1091 | | | | | |
| Dec-14 | .181 | .225 | | | | | |
| Feb-15 | .22 | .15 | | | | .2 | .2042 |
| Apr-15 | | | .2 | | | | |
| 17th Sep-15 | 0 | .1795 | .1385 | | | | |
| 24th Sep-15 | | .1949 | .1333 | | | | |
| Feb-16 | | | | .0974 | .1128 | | |
| All | .1756 | .1696 | .1413 | .0974 | .1128 | .2 | .2042 |

TABLE XXVII: Learning Gain for Initially Beginner Students

Upon further analysis, there appears to be a significant difference in learning for *advanced students only* between those who used short vs standard WOEs. Advanced students appeared to make significant gains from using Short WOEs over Standard WOEs - as observed in column Short WOE and WOE in Table XXVIII. A t-test reports significance with $p = 0.0355$. The same cannot be said for beginners where they appear to benefit more from the standard WOEs, although a t-test shows this difference is not significant ($p = 0.4927$). This significant difference suggests that shorter WOEs may be of greater benefit to advanced students rather than longer WOES.

| Date | Gain per Condition | | | | | | |
|---|---|---|---|---|---|---|---|
| | No WOE | WOE | Short WOE | No Exit WOE | Time Out WOE | Pop-up Analogy | Analogy |
| Sep-14 | -.004 | -.1412 | | | | | |
| Dec-14 | .1037 | .2889 | | | | | |
| Feb-15 | .0667 | .0733 | | | | .0212 | .0333 |
| Apr-15 | | 0 | .1111 | | | | |
| 17th Sep-15 | | 0 | .1267 | | | | |
| 24th Sep-15 | | .04 | .0667 | | | | |
| Feb-16 | | | | -.04 | .014 | | |
| All | .0344 | -.0038 | .1033 | -.04 | .014 | .0212 | .0333 |

TABLE XXVIII: Learning Gain for Initially Advanced Students

In the beginners category, there is also a very strong result for students within the analogy condition with a learning gain of 0.204. It is important to remember that this result can only be compared against statistics within the beginners category and not with the overall results. This result is the highest that we have obtained in our experiments, although non-significant with any of the other conditions (t-test reports $p = 0.6631$ against standard WOE). The non-significance of this result may be due to the low sample count of beginners in the analogy condition - there were only eight students. Even so, it may be worth gathering more samples for this condition to see if similar learning gains can be achieved. We have previously seen in Chapter 3 that analogies are advantageous for beginners, so this may be a promising WOE type to deliver to beginning students.

### 6.7 Analysis of Example Termination Regulation

Prior analysis gave evidence that early termination of examples contributed to learning. Whether that was a good strategy to employ or if that is a good trait of a high learner was unknown. We experimented with this notion in two ways: we made examples that time out after a specified duration, and another than cannot be terminated. The hypothesis is that students in the early termination group may benefit from restarting examples at times. Conversely, the No Exit WOEs should be detrimental to the student as they cannot use the positive strategy of early WOE termination.

Learning gain results (Table XXIII) showed a non-significant difference between the learning gain for No Exit and Time Out WOEs, (unpaired t-test $p = .3957$). Comparing within ability groups also showed no significant difference ($p = .1543$ for advanced, $p = .7913$ for beginners). There is also no significant difference between either of these regulations and the Standard WOE grouping.

It may be possible that the significance tests fail due to the very small sample size collected for both regulation styles - there were only 28 (No Exit) and 32 (Time Out) samples per condition. When taking into account initial knowledge level, these numbers become small and will be susceptible to noise. Further experimentation may be advantageous since the results do show a much smaller mean over the use of no regulation (such as .1696 for beginners in the standard WOE group to .0974 in the No Exit condition). Also, we do see that for all samples in each condition, students in the Time Out condition do perform roughly twice as well as those in the No Exit condition. Here, we can only conclude that regulation cannot be deemed beneficial from our dataset.

### 6.8 Summary

In this chapter, we furthered our investigation of worked-out examples in a computer science intelligent tutoring system by conducting additional experiments in order to gain further insight into potentially useful WOE properties. We have seen that WOE usage can still be advantageous when deployed in different forms. One such highlight is that Short WOEs can yield greater learning gains when used by advanced students, but this is not the case for beginners.

In the next chapter, we will look at how these findings could be used to enhance ChiQat-Tutor.

# CHAPTER 7

# AN ADAPTIVE INTELLIGENT TUTORING SYSTEM USING WORKED-OUT EXAMPLE FEATURES

Previously, we performed analysis on log data and pre/post test information to get an idea of what potential WOE properties may be of use, along with possible ways we may be able to improve an ITS such as ChiQat-Tutor, to use WOE based features. The most significant improvement that has been uncovered so far is with regards to WOE length affecting advanced students. Here, we use prior analysis to start building modifications to ChiQat-Tutor that may enhance learning. The key here is that not all students should be treated equally, as was have seen with the type of WOE that should be delivered, thus a view will be taken to ensure we cater for some without sacrificing others. Therefore, we move onto our final task: Investigate how such features can be used to enhance a computer science intelligent tutoring system. We go about this modification by creating an pipeline to adapt ChiQat-Tutor content to an individual student.

## 7.1 Adaptive Pipeline for ChiQat-Tutor

The main insight that we have got so far has been that not all students learn equally, and that behaviour plays a part in learning. From this, and prior insights, we developed a three stage pipeline (Figure 33) to be used in the linked list tutorial of ChiQat-Tutor that can adapt WOE content to a student. This is done by evaluating the student using system features, including WOE based features, and serve them appropriate WOE content.

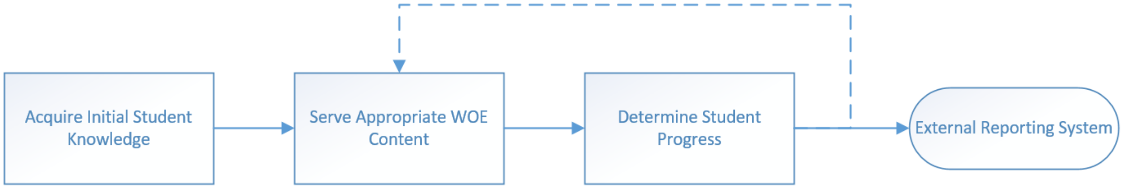

Acquire Initial Student Knowledge
Serve Appropriate WOE Content
Determine Student Progress
External Reporting System

Figure 33: Outline of a Pipeline to Enhance Learning via Worked-out Example Features

In the remainder of this chapter, we will detail how each of these stages will be achieved. Some insight will also be given into what features are important in these models, which will therefore test our hypothesis of whether WOE based features can be used in a CS ITS.

## 7.2 Developing Models

In order to achieve the goals of the pipeline, we will use machine learning techniques to build models. Building computational models for an ITS such as this centres on *Educational Data Mining* (EDM) (Romero and Ventura, 2007). A typical strategy used to model such data is via supervised learning, whereby a set of samples are used to build a model given a known independent class feature. The model is then evaluated for quality by testing it either on a test set (a set of samples not used to build the model), or by cross-validation. Multiple models can be built using different machine learning algorithms, such as Naive Bayes (Rish, 2001) and SVM (Chang and Lin, 2011), where the evaluation results can then be compared. The model that shows the highest level of accuracy or F-score are typically used as the one that best models the data given.

Out of the many tools and methods, we have chosen to use the open source machine learning toolkit, WEKA (Hall et al., 2009). WEKA provides the researcher with access to numerous machine learning algorithms, and allows easy evaluation of each on a given dataset. This toolkit is often used in EDM research, such as (Janning et al., 2016), who uses WEKA to classify perceived task difficulty from student log files. Another example is in (Harley et al., 2013), where the author uses WEKA clustering to identify three different types of student profiles that can be used to dynamically adapt ITS strategies to enhance learning.

Rather than just evaluating various algorithms on our dataset, it may also be useful to evaluate models built with subsets of features. The reasoning for this is two fold. Firstly, adding more features into a training session may reduce the quality of the model, as some of the features may not be inherently related to the class attribute - this is called over-fitting (Elkan, 2001). Therefore the accuracy of the prediction will be lower than it could possibly be. Secondly, finding high accuracy subsets with common features may indicate that the specific feature may be very much related to the outcome, thus give us a feature worthy of further investigation. Such a feature could also then be used to influence further ITS development to enhance learning gains even more.

Unfortunately, the applicable feature set that we have for this task could contain up to 559 different features. It is implausible to use an exhaustive approach and test each combination of features for the best model. *Feature selection* is a method that can reduce the number of features (Guyon and Elisseeff, 2003) to a subset of potentially important features. There are different methods of feature selection (e.g. wrapper and filter methods), and of these multiple different algorithms to select features. Each type of feature selection algorithm does have its advantages/disadvantages, such as some not removing redundant features. Even though reducing the feature space is desirable, it will most likely not result in the optimal set of features, and perhaps the feature selection process may remove useful features.

Therefore, we devised a bootstrapping method by getting an extended set of desirable features by use of multiple feature selection algorithms. We apply four different feature selection algorithms (Table XXIX) onto our dataset, where up to the top 20 highest rated features from each are put into a short-list of candidate features - the short-list would therefore include at most 80 features. If the short-list includes over 20 unique features, we further prune the list of features by removing ones that only show up in one of the algorithms. The remaining features would then be evaluated in an exhaustive approach on 16 different machine learning algorithms in WEKA (Table XXX). These 16 algorithms were selected from a list of 49 potential algorithms, whereby many of these were eliminated either due to them generating invalid models given the training dataset (such as the model generation process threw an error on creation), or performed poorly during manual pilot testing.

Since this may still result in a significantly large number of models, we include the option to further prune the number of combinations to ones that included up to a maximum number of features. This is configurable, depending on the total number of features to balance coverage and runtime duration. In addition to this, a WEKA framework has been developed using the WEKA Java API that parallelises execution of model building and evaluation to multiple CPU cores. We have built a virtualised ESXi (Chaubal, 2008) system, with a dual processor (six cores) setup for model building, which was able to speed up evaluation by approximately 10 times. The process was optimised for throughput by ensuring hardware resources were utilised correctly (Green et al., 2013).

By the end of this process, we will have a selection of computational models using various features, along with different machine learning algorithms. Each model will be accompanied by a measure of

| Evaluator Name | Evaluator Parameters | Search Name | Search Parameters |
|---|---|---|---|
| CfsSubsetEval | -P 1 -E 1 | BestFirst | -D 1 -N 5 |
| InfoGainAttributeEval | | Ranker | -T -1.7976931348623157E308 -N -1 |
| CorrelationAttributeEval | | Ranker | -T -1.7976931348623157E308 -N -1 |
| GainRatioAttributeEval | | Ranker | -T -1.7976931348623157E308 -N -1 |

TABLE XXIX: Feature Selection Methods

| Machine Learning Algorithm | Parameters |
|---|---|
| NaiveBayes | |
| NaiveBayesMultinomial | |
| NaiveBayesUpdateable | |
| Logistic | -R 1.0E-8 -M -1 -num-decimal-places 4 |
| MultilayerPerceptron | -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a |
| SGD | -F 0 -L 0.01 -R 1.0E-4 -E 500 -C 0.001 -S 1 |
| SimpleLogistic | -I 0 -M 500 -H 50 -W 0.0 |
| SMO | -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K ”weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007” -calibrator ”weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 |
| VotedPerceptron | -I 1 -E 1.0 -S 1 -M 10000 |
| AdaBoostM1 | -P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump |
| ClassificationViaRegression | -W weka.classifiers.trees.M5P – -M 4.0 |
| InputMappedClassifier | -I -trim -W weka.classifiers.rules.ZeroR |
| DecisionTable | -X 1 -S ”weka.attributeSelection.BestFirst -D 1 -N 5” |
| OneR | -B 6 |
| J48 | -C 0.25 -M 2 |
| RandomForest | -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1 |

TABLE XXX: Machine Learning Algorithms

accuracy, which will then be able to inform us of how good that model is in describing the data that we have. Also, these models will then be usable in the pipeline to help achieve its goals.

## 7.3 Initial Student Knowledge

The one constant that has appeared in this research thus far has been the importance of initial student knowledge. Pre-test scores have been important in generating linear regression models, and we also have evidence that adapting WOEs to student knowledge may be advantageous. In addition to this, prior work has touted WOEs being more successful with beginner rather than advanced students. Therefore, it may be useful to be able to gauge a student's initial ability level automatically rather than administering a pre-test before the tutorial can begin. There are several benefits for integrating an evaluation method directly into a lesson in this manner. The removal of a pre-test will allow the student to start problem solving faster, hence making the process more efficient. Also, our data utilises pre-test scores from tests that are not easily graded - students were required to draw images, describe issues in natural language, and write blocks of code. If we continue to use such a pre-test, a sophisticated auto-grading system would be needed. Alternatively the pre-test would have to be simplified to allow simple auto-grading.

Having split the students into beginner and advanced groups (dependent variable), we performed our machine learning classification process. Features used were ones only available at the end of the first problem solving exercise. The rationale is that at the end of problem 1 we will be able to predict if a student is a beginner or advanced student. Getting an indicator at such an early stage would then allow the system to quickly adapt its behaviour to the particular student profile.

After removing features from problems other than from problem 1, we were left with 127 features. Since this is still a sizeable amount, we performed our feature selection process, which gave a short-list of 26 different features (excluding the dependent variable). Even though there were over 20 features, we did not eliminate single count features due to there only being 11 features that were selected via multiple algorithms. However, to speed up the brute force process to a manageable time frame, we only evaluated models that included at most 7 features; the number of possible feature sets is then expressed by the binomial coefficient $\binom{26}{7}$. The selected features can be seen in Table XXXI.

| Feature Name | WOE Step Number | Occurrences in Feature Selection Algorithms |
|---|---|---|
| Solution Correct | N/A | 4 |
| User Acknowledge | N/A | 4 |
| WOE Exit Step Count | 7 | 4 |
| First Mean Explanation Step Dur | N/A | 3 |
| Good Submissions | N/A | 3 |
| Longest WOE Dur | 11 | 3 |
| Positive Feedback Given | N/A | 3 |
| Longest WOE Dur | 2 | 2 |
| Positive Feedback | N/A | 2 |
| Success Attempt | N/A | 2 |
| Problem Dur | N/A | 2 |
| Avg. Step Duration | N/A | 1 |
| First Step Dur | 1 | 1 |
| First Step Dur | 2 | 1 |
| First Step Dur | 6 | 1 |
| First Step Dur | 8 | 1 |
| First Step Dur | 9 | 1 |
| First Step Dur | 11 | 1 |
| First Mean Intro Step Dur | N/A | 1 |
| First Mean Pause Step Dur | N/A | 1 |
| Longest Dur | 9 | 1 |
| Mean Step Dur | 9 | 1 |
| St Dev WOE Duration | N/A | 1 |
| St Dev WOE Steps | N/A | 1 |
| Example Duration | N/A | 1 |
| Total WOE Duration | N/A | 1 |

TABLE XXXI: Selected Features for Predicting Initial Knowledge

WOE based features do make an appearance with 18 out of 26 features. One type of feature that has not been explained so far is *First WOE Step Duration*, which gives the duration of a step in the first problem when it had been stepped for the first time. We also see stepping durations for some of the step annotations, such as for the introduction step and any paused step.

From here, we employed our WEKA evaluation framework to build and test permutations of these features, which generated 35.95 million models. The top 10 models are shown in Table XXXII. A baseline of Naive Bayes is also given: Naive Bayes was selected as a baseline due to it being more meaningful than chance, yet being a simple classifier. The accuracy of the top model was deemed significantly better than compared to both majority class or the baseline for either all features or those gathered via feature selection ($\chi^2$, $p < 0.005$). The most promising algorithm here is the Multilayer Perceptron classifier, where the top model is also significantly better than such a classifier that has been trained on all available features or the set created from feature selection. This gives evidence that initial student knowledge can be evaluated, with accuracy of 77%, from just using the system on problem 1 rather than having to employ a pre-test on a student. This classifier could therefore be used on the first stage of the pipeline to evaluate student knowledge.

As already mentioned, the features identified as important through feature selection include many WOE related features. To get a better look at promising features, we looked at the frequency of features used in the top 1,000 models (Table XXXIII). The reason for this is that many of the top models shared similar, or identical, accuracies. Getting a frequency for each feature will tell us how often this feature contributes to a top model. Many of the features here do not exist in the majority of models - only two features occur in over half of the models. This can most likely be explained by the large number of available features and restriction on number of features that were contributed to any one model. However, the top feature which occurs in 99.3% of models is the count of examples that are exited on step 7 - that is half way through the example. This is interesting and reinforces our hypothesis that exiting WOEs may be a useful feature in our pipeline. The step duration also appears to be important with the 70.2% of models including the duration of step 11 (for the first time on that step). Although all

| Feature | Step | MLP | MLP | MLP | MLP | MLP | L○ | L2 | L3 | SGD | MLP | NB S | NB A | MLP S | MLP A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| User Ack. | | ○ | | | | ○ | | | | | ○ | ○ | ○ | ○ | ○ |
| Sol. Correct | | ○ | | ○ | ○ | ○ | ○ | ○ | ○ | | | ○ | ○ | ○ | ○ |
| Pos. Feedback Given | | ○ | ○ | ○ | | | | | | | ○ | ○ | ○ | ○ | ○ |
| Problem Dur | | | | | ○ | | | | ○ | | | ○ | ○ | ○ | ○ |
| Success Attempt | | | ○ | | | | | | | ○ | | ○ | ○ | ○ | ○ |
| Good Sub. | | ○ | | | ○ | ○ | | | | ○ | | ○ | ○ | ○ | ○ |
| Pos. Feedback | | | | | ○ | | ○ | ○ | | ○ | ○ | ○ | ○ | ○ | ○ |
| Total WOE Dur | | | | ○ | | | | | | | | ○ | ○ | ○ | ○ |
| StDev WOE Dur | | | | | | | | | | | | ○ | ○ | ○ | ○ |
| StDev WOE Steps | | | | | | | | | | | | ○ | ○ | ○ | ○ |
| Avg Step Dur | | | ○ | | | | | | | | | ○ | ○ | ○ | ○ |
| First WOE Step Dur | 1 | | | | | | | | ○ | | | ○ | ○ | ○ | ○ |
| First WOE Step Dur | 2 | | | | | | | ○ | ○ | | ○ | ○ | ○ | ○ | ○ |
| First WOE Step Dur | 6 | | | | | | | | | | | ○ | ○ | ○ | ○ |
| First WOE Step Dur | 8 | | ○ | | | | ○ | | | | | ○ | ○ | ○ | ○ |
| First WOE Step Dur | 9 | | | ○ | | | | | | | | ○ | ○ | ○ | ○ |
| First WOE Step Dur | 11 | ○ | ○ | ○ | | ○ | ○ | ○ | ○ | | ○ | ○ | ○ | ○ | ○ |
| Mean WOE Step Dur | 9 | | | | | | | | | | ○ | ○ | ○ | ○ | ○ |
| WOE Exit Step Count | 7 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Longest WOE Dur | 2 | ○ | ○ | ○ | | ○ | ○ | ○ | | | | ○ | ○ | ○ | ○ |
| Longest WOE Dur | 9 | ○ | | | | | | | | | | ○ | ○ | ○ | ○ |
| Longest WOE Dur | 11 | | | | ○ | | ○ | | | ○ | | ○ | ○ | ○ | ○ |
| FMean Intro Step Dur | | | ○ | | | | ○ | ○ | | ○ | ○ | ○ | ○ | ○ | ○ |
| FMean Exp. Step Dur | | ○ | | ○ | ○ | | | | | ○ | ○ | ○ | ○ | ○ | ○ |
| FMean Pause Step Dur | | | | | | | | | ○ | | | ○ | ○ | ○ | ○ |
| Accuracy | | .771 | .771 | .771 | .771 | .771 | .771 | .771 | .764 | .764 | .764 | .662 | .648 | .64 | .568 |
| Precision | | .799 | .782 | .785 | .782 | .799 | .785 | .789 | .779 | .784 | .809 | .669 | .654 | .645 | .569 |
| Recall | | .771 | .771 | .771 | .771 | .771 | .771 | .771 | .764 | .764 | .764 | .662 | .647 | .64 | .568 |
| F-Score | | .765 | .768 | .768 | .768 | .765 | .768 | .767 | .760 | .759 | .754 | .655 | .640 | .634 | .568 |

TABLE XXXII: Top Predictive Models for Predicting Initial Knowledge (Beginner/Advanced)

non-WOE specific features did occur in the top half of the table, it does show that many WOE features are just as important as non-WOE features in determining initial student knowledge.

| Feature | Step Number | Presence Frequency |
|---|---|---|
| WOE Exit Step Count | 7 | 993 |
| First WOE Step Dur | 11 | 705 |
| Solution Correct | 1 | 487 |
| Good Submissions | 1 | 475 |
| Positive Feedback Given | 1 | 457 |
| Positive Feedback | 1 | 442 |
| Longest WOE Dur | 11 | 407 |
| First Mean Explanation Step Dur | N/A | 339 |
| First Mean Intro Step Dur | N/A | 309 |
| First WOE Step Dur | 2 | 286 |
| Success Attempt | 1 | 271 |
| User Acknowledge | 1 | 269 |
| Longest WOE Dur | 2 | 259 |
| Total WOE Duration | 1 | 237 |
| Longest WOE Dur | 9 | 200 |
| First WOE Step Dur | 9 | 191 |
| First Mean Pause Step Dur | N/A | 183 |
| StDev WOE Duration | 1 | 181 |
| Mean WOE Step Dur | 9 | 180 |
| First WOE Step Dur | 8 | 179 |
| Avg Step Duration | 1 | 161 |
| Problem Dur | 1 | 136 |
| First WOE Step Dur | 1 | 118 |
| First WOE Step Dur | 6 | 77 |
| StDev WOE Steps | 1 | 72 |

TABLE XXXIII: Frequency of Features in top 1,000 Initial Knowledge Classification Models

Using this knowledge, we also gain some insight into the stepping behaviour of the student from features such as the stepping duration. This gives some evidence that behavioural stepping patterns may be related to initial student knowledge.

## 7.4 Serving Example Content

The second stage in the pipeline is to serve appropriate WOE content. As was seen in section 6.6, there was a significant difference between student learning when advanced students used short and standard WOEs. This evidence can be used within this stage to deliver appropriate WOE content to the student.

Since by the end of problem 1 we do know the student's initial state from the first stage in our pipeline, we can serve them appropriate content for problem 2 - advanced students will be served short examples, while beginners will receive standard examples.

## 7.5 Predicting Learning

The final part of the pipeline is to evaluate the student. The purpose of this step is two fold: (1) provide additional feedback to the second stage of the pipeline in order for it to better select WOE content, and (2) be able to automatically report student progress to an external entity (such as a teacher or grading repository). While this could be done using various metrics in the ITS, such as number of problems completed, an alternative is to predict the post-test score, which done with respect to the binary classification of high or low gainer.

The aim here is to create a model that can be built using a subset of features that can predict if a student is a high or low gainer at the end of the tutorial. This therefore allows us to use all generated features and not restricted to a certain subset of problem features. Finding a good model is two fold: (1)

we can use this model directly to predict a student's gain, (2) use the features that were used in such a model to gain additional insight into what may make a student a high or low gaining student.

Even though we can use any subset of features that we wish, there is an added layer of complexity with regards to the type of WOE that has been used. Thus far we have looked solely at standard WOEs while building models. This will not translate over to students who use short WOEs, which they could use if they were to have been deemed an advanced student at an earlier stage of the pipeline. The issue is that a feature for step one of a standard WOE is a different feature from that of step one in a short WOE. Available options are to either have separate features for each WOE type, so standard WOE step one is a separate feature from the corresponding short WOE, or we can choose to build separate models for each WOE type. Here, we have chosen to use the latter and build a separate model for each WOE type. Having this extra dimensionality in a model may introduce extra complexity that may reduce the effectiveness of a classifier.

### 7.5.1 Using Standard Worked-out Examples

The full feature set contains 559 features, which were then reduced to 18 after going through our feature selection process. Here, features that were present in only one feature selection algorithm were removed. Details of the selected features, and their counts, are shown in Table XXXIV. In addition to these features we also included the target feature (binary feature of high/low gainer) to learn from, and the student's initial state (binary feature of beginner/advanced).

In total, approximately 3.93 million models were created and evaluated, with the top 10 models shown in Table XXXV. The last two sets of results in the table show the baselines of using Naive Bayes and Multilayer Perceptron on the two supersets - one being all feature selected attributes and the other

| Feature Name | Problem Number | Count |
|---|---|---|
| All Woes Incomplete | 6 | 4 |
| Avg. Step Duration | 2 | 4 |
| Initial State Beginner/Advanced | N/A | 4 |
| Problem List Requested | 3 | 4 |
| Question Given | 3 | 4 |
| St. Dev. Step Duration | 2 | 4 |
| All WOEs Incomplete | 3 | 3 |
| Avg. WOE Duration | 2 | 3 |
| Node Clicked | 3 | 3 |
| Node Clicked | 6 | 3 |
| Positive Question Answer Response | 3 | 3 |
| Positive Answer Response | 3 | 3 |
| Problem Duration | 2 | 3 |
| Attempts After WOE | 6 | 2 |
| Positive Feedback Given | 2 | 2 |
| Positive Feedback | 2 | 2 |
| Scratch Pad Next | 3 | 2 |
| Scratch Pad Previous | 3 | 2 |
| Attempts | 3 | 1 |
| Examples Duration | 2 | 1 |
| Good Submissions | 2 | 1 |
| New Problem Selected | 3 | 1 |
| Operation Execution Submitted | 3 | 1 |
| Positive Feedback Given | N/A | 1 |
| Positive Feedback Given | 3 | 1 |
| Positive Feedback | 3 | 1 |
| Solution Correct | 2 | 1 |
| Total WOE Duration | 2 | 1 |
| User Acknowledge | 3 | 1 |

TABLE XXXIV: Selected Features for Predicting Learning Gain for Standard WOE Students

including all features before feature selection. A $\chi^2$ of the results does show a significant difference ($p < 0.005$) in model accuracy between the top model and the baseline. This therefore suggests that the selected features do yield improved models over all features, and furthermore, subsets of the selected features are even more promising. Such a model can therefore be used in the final stage in our pipeline, where it will be able to report a predicted binary learning gain once the student has finished using a section of the system with accuracy of 81.3%. This will still be valid whether or not the student completed the tutorial. The purpose of this output is for external reporting, such as to a teacher, or for feedback into the WOE selection stage of the pipeline.

| Algorithm | Prob | MLP | MLP | MLP | J48 | J48 | J48 | J48 | ABM | ABM | MLP | NB S | NB A | MLP S | MLP A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Positive Feedback Given | 2 | ○ |  |  |  |  |  |  | ○ |  | ○ | ○ | ○ | ○ | ○ |
| Problem Dur | 2 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Positive Feedback | 2 |  |  |  |  |  |  | ○ |  |  | ○ | ○ | ○ | ○ | ○ |
| Avg WOE Duration | 2 |  | ○ | ○ |  |  | ○ |  | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Avg Step Duration | 2 |  |  | ○ | ○ | ○ |  |  | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| StDev StepDuration | 2 |  |  | ○ |  |  |  |  | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Problem List Requested | 3 |  |  | ○ |  |  |  | ○ |  | ○ |  | ○ | ○ | ○ | ○ |
| Node Clicked | 3 | ○ | ○ | ○ | ○ | ○ | ○ |  |  |  |  | ○ | ○ | ○ | ○ |
| Question Given | 3 |  |  |  |  |  |  |  | ○ |  | ○ | ○ | ○ | ○ | ○ |
| Scratch Pad Next | 3 |  | ○ |  |  | ○ | ○ |  | ○ |  | ○ | ○ | ○ | ○ | ○ |
| Scratch Pad Prev | 3 |  | ○ | ○ |  | ○ | ○ | ○ | ○ |  | ○ | ○ | ○ | ○ | ○ |
| Positive Question Answer Response | 3 | ○ |  | ○ | ○ | ○ |  |  | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Positive Answer Response | 3 | ○ | ○ |  | ○ |  | ○ | ○ | ○ |  | ○ | ○ | ○ | ○ | ○ |
| All WOEs Incomplete | 3 | ○ | ○ |  | ○ |  |  | ○ | ○ |  | ○ | ○ | ○ | ○ | ○ |
| Node Clicked | 6 |  | ○ | ○ |  |  |  |  |  |  | ○ | ○ | ○ | ○ | ○ |
| Attempts After WOE | 6 | ○ | ○ | ○ |  |  |  |  |  |  |  | ○ | ○ | ○ | ○ |
| All WOEs Incomplete | 6 |  | ○ |  | ○ | ○ | ○ |  | ○ | ○ |  | ○ | ○ | ○ | ○ |
| Initial State Beginner/Advanced |  | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| Accuracy |  | 0.813 | 0.806 | 0.806 | 0.799 | 0.799 | 0.799 | 0.799 | 0.799 | 0.799 | 0.799 | 0.669 | 0.547 | 0.683 | 0.683 |
| Precision |  | 0.821 | 0.816 | 0.813 | 0.807 | 0.810 | 0.802 | 0.807 | 0.807 | 0.807 | 0.804 | 0.688 | 0.546 | 0.685 | 0.683 |
| Recall |  | 0.813 | 0.806 | 0.806 | 0.799 | 0.799 | 0.799 | 0.799 | 0.799 | 0.799 | 0.799 | 0.669 | 0.547 | 0.683 | 0.683 |
| F-Score |  | 0.811 | 0.803 | 0.804 | 0.796 | 0.796 | 0.797 | 0.796 | 0.796 | 0.796 | 0.797 | 0.657 | 0.546 | 0.682 | 0.683 |

TABLE XXXV: Top Predictive Models for Predicting High/Low Gainer

Also of interest are the features that are used in building these models. The features selected from our feature selection bootstrapping method gave 18 different features. Out of these features, 6 of them are WOE based features, these being:

- All Woes Incomplete 3 - 1 if no WOEs were completed while working on problem 3.
- Avg. Step Duration 2 - The average time spent on a step in problem 2.
- St. Dev. Step Duration 2 - The standard deviation of time spent on a step in problem 2.
- All WOEs Incomplete 6 - 1 if no WOEs were completed while working on problem 6.
- Avg. WOE Duration 2 - The average duration of using a WOE in problem 2.
- Attempts After WOE 6 - Number of problem attempts after a WOE has been used in problem 6.

The incompleteness of examples are featured, which echoes our prior linear regression analysis. Also, out of these six features, three of them involve the amount of time spent on WOEs, whether that would be at a step or example level. It is also interesting to note that WOE features appear to be more prominent on problems 2, 3, and 6. This suggests that WOE stepping behaviour at various points in the tutorial could be an indicator of performance.

Going back to Table XXXV, the best models do not include all features, but rather a subset of them. The best model achieved was by using WEKAs Multilayer Perceptron algorithm (Arora, 2012). The features used in this model were:

- Positive Feedback Given 2 - Number of positive feedbacks given in problem 2.
- Problem Duration 2 - Total duration of problem 2.

- Node Clicked 3 - Number of times nodes had been clicked on the interface during problem 3.
- Positive Question Answer Response 3 - Number of times a question posed to the student was answered correctly on problem 3.
- Positive Answer Response 3 - Number of times a question posed to the student was answered correctly on problem 3.
- All WOEs Incomplete 3 - 1 if no WOEs were completed while working on problem 3.
- Attempts After WOE 6 - Number of problem attempts after a WOE has been used in problem 6.
- Initial Student Knowledge State (Beginner/Advanced) - 0 if the student was deemed to be a beginner, 1 if advanced, at the start of the exercise.

Out of these eight features, only two of them included WOE features, those being ones at a macro level rather than micro. This feature set only takes into account the most accurate model. In addition to this, Table XXXVI gives the frequencies of each feature that exists in the top 1,000 models, regardless of machine learning algorithm used. It is clear that the most important feature is knowing the student's initial state, which mirrors the importance of pre-test score in our previous linear regression models. This can easily be attained from the first step of the pipeline. Aside from the initial knowledge state, four out of the remaining nine features that appear in over half of the models are WOE features. Unlike the features in the top model, the top features do include macro level features (all WOEs incomplete, average WOE duration, and attempts after WOE completion) and micro level features (average step duration). This suggests that even though step level features are not included in the top model, they often contribute to high quality models, and thus may be a contributor to learning. These micro level features are linked to WOE stepping patterns. Therefore this does provide some evidence, albeit not as strong as when modelling initial student knowledge, that WOE usage patterns may also be an indicator of student learning.

| Feature | Problem Number | Presence Frequency |
|---|---|---|
| Initial Knowledge State (Beginner/Advanced) | N/A | 1000 |
| Problem Duration Problem | 2 | 851 |
| All WOEs Incomplete Problem | 6 | 649 |
| Positive Question Answer Response Problem | 3 | 647 |
| Positive Answer Response Problem | 3 | 635 |
| Scratch Pad Previous Count Problem | 3 | 614 |
| Attempts After WOE Problem | 6 | 577 |
| Scratch Pad Next Problem | 3 | 556 |
| Avg. Step Duration Problem | 2 | 546 |
| Avg. WOE Duration Problem | 2 | 525 |
| All WOEs Incomplete Problem | 3 | 470 |
| St. Dev. Step Duration Problem | 2 | 408 |
| Problem List Requested Problem | 3 | 364 |
| Node Clicked Count Problem | 6 | 342 |
| Node Clicked Count Problem | 3 | 294 |
| Positive Feedback Count Problem | 2 | 268 |
| Positive Feedback Given Count Problem | 2 | 258 |
| Question Given Count Problem | 3 | 211 |

TABLE XXXVI: Frequency of Features in top 1,000 Standard WOE Learning Gain Classification Models

In addition, all features that are related to problems only include problem 2, 3, and 6. It is unknown why this may be the case, however, a commonality of all three problems is that they are some of the initial problems of a given type that the student encounters. While barring problem 1, due to it having seemingly different behaviour to other problems, problems 2 and 3 can be seen as introductory problems, while 4 and 5 are similar to the prior ones. Problem 6 can be explained by it being of a different type of problem, a block completion problem, rather than a step-by-step problem. Problem 7 is similar to problem 6, thus it does not appear to be of importance. Also, few students reach this problem. It would be interesting to see what other similarities these problems have in common other than the obvious, as such features may be good at predicting knowledge growth.

### 7.5.2 Using Short Worked-out Examples

Although we have received promising results from the standard WOE classification, we do have several issues with building a corresponding classifier for short WOEs. Although the process would be the same, and therefore may achieve similar results, we do not currently have much data to build a binary classifier. The previous classifier was able to use 139 samples from our dataset, however, we only have 45 samples that can be used for this binary classifier. Another issue that we have from our data is that our samples are for certain conditions, whereby a student uses only one type of WOE for the duration of the exercise. This does not match up with our pipeline as all students start off with standard WOEs during the first problem, then they may be moved onto short WOEs for the rest of the exercise. A more suitable dataset would be one using the initial ability classifier, and then use the data points for students who were given access to short WOEs from problem 2.

Even though there are such deficiencies with our dataset, we did attempt to perform feature selection on all features available to students in the short WOE condition. The original 512 features in the dataset had been reduced to 20 features using the same process as with the prior classifier modelling. However, the features selected did not appear to be of the same quality as with prior analyses. Table XXXVII shows a list of all selected features, and the striking difference from other feature selection processes conducted so far is the low count of features that were selected in more than one algorithm. Of the four features that were selected in multiple algorithms, it is surprising to see that the number of times a student logged out of the system was featured. Also, WOE features only appear in two of the features selected.

| Feature Name | Problem Number | Count |
|---|---|---|
| Problem Duration | 6 | 4 |
| Solution Incorrect | 1 | 4 |
| User Logged Out Count | N/A | 4 |
| Bad Submissions | 1 | 3 |
| Attempts | 6 | 1 |
| Average WOE Duration | 2 | 1 |
| Good Submissions | 5 | 1 |
| Good Submissions | 6 | 1 |
| Negative Question Answer Response | N/A | 1 |
| Neutral Feedback Given | 5 | 1 |
| Neutral Feedback Given | 6 | 1 |
| New Problem Selected | 6 | 1 |
| Node Clicked Count | 1 | 1 |
| Operation Execution Submitted | 6 | 1 |
| Problem List Requested | 5 | 1 |
| Problem List Requested | 6 | 1 |
| Solution Correct | 5 | 1 |
| Success Attempt | 5 | 1 |
| Success Attempt | 6 | 1 |
| WOE Exit Step 1 Count | 1 | 1 |

TABLE XXXVII: Selected Features for Predicting Learning Gain for Short WOE Students

The unusual results here may be due to the lack of samples in our dataset. Even so, we did build models using our WEKA bootstrapping process using the selected features. Over 15 million models were built during the process, with 1,926 models achieving the highest level of accuracy at 86.7%. A frequency count of features in these models shows that the two WOE features selected earlier appeared in the top three features (Table XXXVIII), with the number of WOE exits on the first step of the first problem being shown in every model.

| Feature | Problem Number | Presence Frequency |
|---|---|---|
| WOE Exit Step 1 Count | 1 | 1926 |
| User Logged Out Count | N/A | 1886 |
| Average WOE Duration | 2 | 1792 |
| Problem List Requested | 6 | 1342 |
| Operation Execution Submitted | 6 | 1274 |
| Attempts | 6 | 1274 |
| Neutral Feedback Given | 6 | 1270 |
| New Problem Selected | 6 | 1229 |
| Bad Submissions | 1 | 1089 |
| Neutral Feedback Given | 5 | 1012 |
| Solution Incorrect | 1 | 1009 |
| Good Submissions | 5 | 966 |
| Solution Correct | 5 | 963 |
| Problem List Requested | 5 | 954 |
| Success Attempt | 5 | 932 |
| Success Attempt | 6 | 930 |
| Good Submissions | 6 | 860 |
| Problem Duration | 6 | 169 |
| Negative Question Answer Response | N/A | 165 |
| Node Clicked Count | 1 | 14 |

TABLE XXXVIII: Frequency of Features in top 1,926 Short WOE Learning Gain Classification Models

Although we have been able to create a set of models with high accuracy, the original two deficiencies still remain, those being the lack of data and the use of features of problem 1 that differ from our proposed pipeline. Therefore, these results should only be used to indicate that building models using short WOE data may be of use in building a learning gain classifier, and for further work to collect additional data that would be more relevant to our pipeline.

### 7.6 Review of Adaptive Pipeline

In this chapter, we introduced a pipeline that can be used in a tutoring system such as ChiQat-Tutor, that can adapt the system based on student profile data. The pipeline is able to predict initial student knowledge, then serve appropriate WOE content to the student, and it will finally classify the student's progress in the system as either a high or low gainer. This classification is then used for both external reporting, and potentially for feedback to the WOE selection component of the pipeline for future potential enhancements.

# CHAPTER 8

# CONCLUSIONS AND FUTURE WORK

In this research, we have investigated the use of worked-out examples in a computer science intelligent tutoring system. The overall research question posed is to ask if worked-out examples can be an effective strategy in such an ITS. Along the way we set out four tasks to answer this question.

We started off by asking if worked-out examples are of use in human one-on-one tutoring in this domain. We concluded that they can be beneficial to students, which led us onto a second task on investigating if they can be effective in an ITS. Our conclusion was mixed with some students benefiting from the strategy while others did not. The interesting takeaway from this task was that there appeared to be some behavioural differences between students that would gain more-so from WOEs over other students.

This then posed a third task of finding out what properties of WOEs may be useful for students. Given the results from our second task, we hypothesised that regulation of WOE delivery and the length of examples may be important properties to include in an ITS. Our results were inconclusive regarding regulation, but showed promise for the property of WOE length. It appeared that advanced students gain more from our authored short WOE rather than our standard version. This was not the case for novice students.

Finally, we concluded with a final task where we mapped out a potential improvement to ChiQat-Tutor. We chose to use a pipeline of steps that would determine a students initial ability level, serve appropriate content, and then predict if the student was a high or low gaining student. The models used in this pipeline appear to be fairly robust with accuracies over 75%. Also, these models do use WOE based features, suggesting that student WOE behaviour could be an indicator for student evaluation.

We therefore conclude that worked-out examples can be an effective strategy to use in a computer science intelligent tutoring system, although much work has to be done to utilise them correctly.

## 8.1 Contributions

In this research, we have provided seven contributions:

- Establish the usefulness of worked-out examples in a computer science intelligent tutoring system.
- A flexible computer science ITS, ChiQat-Tutor, that allows evaluating various teaching strategies.
- Worked-out examples may be of benefit to some students, some of the time, and cannot be considered a silver bullet.
- Students worked-out example behavioural patterns can offer insight into potential learning gains.
- Short, concise worked-out example may be more beneficial to advanced students than longer ones. Beginners do not benefit in this way.
- A pipeline that uses system and WOE features to potentially improve student learning.
- Initial student knowledge level can be classified in an ITS, which can be improved via the use of WOE based features.

In addition to these main contributions we have also provided the ground work for further investigation involving regulation of worked-out examples.

## 8.2 Next Steps

We have covered much of what we set out the accomplish with regards to our research questions and goals. Regarding the research question of whether or not worked-out example features can contribute to a Computer Science Intelligent Tutoring System, we can state that we do have some evidence that such features may be able to in several ways. Even so, there are some immediate next steps that should be taken.

First and foremost would be to gain a greater understanding of how the features in our computational models are effective. We had only gone as far as to identify what features contribute to either learning or assessing initial knowledge. The next step would be to see how each of the features affect learning, what features are effective in conjunction with others, and then understand how such features could be used in a cognitive sense.

The top models tended to use a form of neural network, which would be non-trivial to decipher. However, some models can be analysed fairly easily, such as Naive Bayes or J48 (decision tree). Knowing the impact of these features may give us hints on how to alter teaching strategies in ChiQat-Tutor.

An interesting line of research is how WOEs should be authored for students. One of the merits of this research is that we were able to use a classifier to understand a student's initial level of knowledge, which can then be used to serve an appropriate on-demand example. Furthermore, we identified that Short WOEs were significantly desirable for advanced students, while the opposite was true (non-significantly though) for beginners. Next we would have to look at student profiles and what attributes of WOEs are optimised for those profiles.

We took a coarse grained approach to WOE adaptivity by only looking at three types of WOE: standard, analogy, and short. We need to understand what makes WOEs effective for various profiles. This would require a deeper analysis into WOE construction. There are many facets of WOE, such as the amount of technical detail contained in a WOE, the language used, graphical features, etc. Some facets were exercised by using analogies, with the use of different graphics and relatable content. Short WOEs attempted to alter the language in a WOE by making it more concise, such as removing superfluous text and steps. We found that being more concise was advantageous for advanced users. However, these three WOE types are just starting blocks to further analysis.

WOEs should be more carefully constructed to test individual hypothesises from a more pragmatic viewpoint. For example, analogies included several changes over standard WOEs, such as graphics and a story. It is unknown if any of these elements, and if so which ones, contribute towards the effectiveness of the strategy. Combinations of such elements may also be of importance, for example graphics may only be advantageous if short language is used.

Even though we should look at other WOE content, the significance of Short vs Standard WOEs for advanced students cannot be ignored. An analysis of what differences exist between the two WOE types may uncover features that may be needed for one profile over another. There are three major differences between these WOEs and the standard WOES: more direct language; no reflection; and fewer steps.

The language of WOEs could be analysed in conjunction with the amount of time spent on individual steps, which has been shown to be a valuable feature. An analysis of natural language could be undertaken to see if there are any language specific features, such as parts of speech, word sentiment, or word complexity, that may be appropriate for different student profiles.

The type of steps being used could be investigated in greater depth. Standard WOEs included reflection in Problem 1 where a mistake was made and the tutor prompted the student to think about the mistake. This appeared to be better for beginners than advanced students. We did touch on these step types in our analysis by annotating the examples. A more formal analysis of these could be conducted to see if ones such as pausing are detrimental to the progress of advanced students or not.

Finally, more data should be collected to truly evaluate regulation. It is unfortunate that the sample sizes for these experiments was fairly small, even though a lot of the ground work was put in place, and some interesting results were achieved. Getting more data here may allow us to make solid claim over the observations.

## 8.3 Future Work

There are many ways forward in this research, aside from the next steps that have already been laid out. First and foremost would be to see how this work may translate to other activities and domains. We have focused purely on the domain of computer science, and within that, linked lists. Literature has shown that WOEs do work well is certain domains, usually ones rooted in algorithmic concepts. Even though WOEs may not be as beneficial to other domains, it may be interesting to see if there are some populations of these domains that may benefit from them. This could then lead onto the question whether there are any behavioural features from the user that may provide insight into their potential goals.

We only looked at a single type of example here, those being static examples that the student needs to click through. There are different types of examples in current literature, such as faded examples (i.e. examples that interleave example steps and problem solving steps with the proportion of each being dictated by student performance). It would be interesting to see how these other types of examples affect learning, and if there are any different behavioural traits that would yield even stronger features that could be of use in an adaptive system. The modality of examples, such as audible and video examples may also lead to new feature types.

Our system utilises a fairly coarse grain approach to adaptivity. The proposed pipeline only adapts the system after the first problem (that gauges initial student knowledge), and then again at the end of the seventh problem (which is currently the end of the tutorial). It would be highly advantageous for adaptivity to be more immediate. Other work, such as that with faded examples, will change WOE content based on the student's performance on the prior activity. Such a fine-grained approach would allow the system to respond quicker than the currently proposed pipeline could do.

In this investigation, we have been trying to gather features from the student's usage of the system. However, there may be some features outside of the system that may be useful that do relate to the use of examples. For example, we could look at the student's affective state while using examples. We hypothesised, and provided evidence for, that student behaviour during WOEs can be used as useful features for the ITS, whether that is to gauge performance or current knowledge. Some states, such as pausing between steps, may be an indicator of an affective state, such as confusion.

# APPENDICES

# Appendix A

# WORKED-OUT EXAMPLE CODING MANUAL

## A.1 Definition

A worked out example is a step-by-step demonstration of how to perform a task or how to solve a problem (Clark et al., 2011). Worked out examples consist of a problem formulation, solution steps, and the final solution (Hilbert et al., 2004).

## A.2 Coding Category

Worked out examples are to be primarily marked with a single coding group, ***worked-out***. There will also be two subgroups, ***level1-worked-out*** and ***level2-worked-out***, which will signify nested worked out examples.

The ***worked-out*** code group contains 3 codes:

- begin-worked-out: The beginning of a worked out example episode
- end-worked-out: The ending of a worked out example episode
- single-worked-out: If a worked out example starts and finishes in one turn, this code marks that turn as a worked out example.

Similarly, the ***level1-worked-out*** code group contains 3 codes:

- lvl1-begin-worked-out
- lvl1-end-worked-out
- lvl1-single-worked-out

The ***level2-worked-out*** code group contains 3 codes:

- lvl2-begin-worked-out
- lvl2-end-worked-out
- lvl2-single-worked-out

The codes for ***level1-worked-out*** and ***level2-worked-out*** serve the same goals with the codes for ***worked-out***.

The three coding groups are to be used in a hierarchical structure. A ***worked-out*** example episode can contain several ***level1-worked-out*** example episodes, a ***level1-worked-out*** example episode can contain several ***level2-worked-out*** example episodes. A child episode must start and end within its parent episode. A ***level1-worked-out*** and ***level2-worked-out*** example episode cannot exist without a **direct** parent episode. A single turn episode (marked with single-worked-out and lvl1-single-worked-out) cannot have child episodes.

## Appendix A (Continued)

### A.3 Marking Worked Out Examples

### A.3.1 Outline

A worked out example must possess the following:

- Specific problem formulation
- Steps to solve the problem
- Problem solution/conclusion

A worked out example is not:

- A problem to be solved by the student (however, can lead to one if the student does not solve the problem and the tutor solves it for them)
- A procedure that has no initial problem formulation

### A.3.2 Examples

A worked out example starts when the tutor introduced a specific problem. Each of the following examples are the start of a worked out example.

(A.1) JAC: now if we were to delete five +...

(A.2) JAC: um there's one where you do a search on a tree say we search for +// let's come here and say we searched for nine.

(A.3) LOW: suppose we were trying to delete e@l.

Before a tutor gives a specific example, they may give a general description or goal for the specific example. Those descriptions do not count the beginning of a worked out example. In the following episode, the work out examples start from the second turn.[1]

(A.4) LOW: so we got to invent a search algorithm.
LOW: so the first one is search for +// let me cover it up for you +// search for eight. [START]

(A.5) LOW: case zero, delete a node with zero kids, which is called by the way a leaf.
LOW: so for example, suppose you're going to take six back out. [START]

(A.6) LOW: alright, the case which at first seems extremely difficult is case two, two kids.
LOW: so example they say here is to delete the five. [START]

[1] The general description was discussed in the project meeting on 09/22/2012. Right now, we do not mark anything on the general description, but we may mark it as goal in the next pass of annotations.

## Appendix A (Continued)

A worked out example ends when the tutor draws the final solution for the first time. A possible example of this can be seen in Example Equation A.7, when the solution is given for a YES/NO style question.

(A.7) JAC: so this is a binary search tree.

For worked out examples about inserting/deleting/search node in a data structure, the worked out example ended when the tutor achieved the original goal.

The tutors may give explanations about why they took the approach they demonstrated (see Example Equation A.8, the episode ended at the first line), or generalize the specific example to a class of examples (see Example Equation A.9), or elaborate the problem (see Example Equation A.10). The rule about whether to include those turns into the worked out example episode is whether those turns happened before the tutor draw the final solution for the first time.

(A.8) JAC: it's not a typical binary search tree. [END]
JAC: the standard format is # right child is always greater than the parent # left child is always less than the parent.
101: okay.
JAC: okay.
JAC: so here we have +// we have exactly the opposite.
JAC: we have right child is less than the parent and left child is greater than the parent.
JAC: so the right is +// so right is greater # left is less than.
JAC: so here we don't have a binary search tree.

(A.9) LOW: make five point to eight, exactly. [END]
LOW: and to talk about that in general, if we're in this case with one kid, doesn't really matter if it's a left kid or a right kid.
...
LOW: so that case is simple and as you see, it modified almost nothing in the tree.

(A.10) JAC: so what we would do in case of finding null would be to return here # false or we return +// we might return null. [END]
101: +< uh huh.
101: okay.
JAC: depends on your +// depends on your function and how you're implementing this um we might return null because you want to know that it's not there and you might say okay since four is not there I'm going to add four or you might return false only because you just want to find out if it's a new xxx or not.

During an example, there is the possibility of the episode becoming interactive. There are several scenarios where this may happen. Firstly, the tutor could be interrupted by the student when they are presenting the example, giving the solution to the example (see Example Equation A.11).

## Appendix A (Continued)

(A.11) LOW: *uh, as a simple example *uh, use fractions.
LOW: ok, seven +// where would seven and a half go?
111: right, left, left, the right.
LOW: so seven and a half would hang off here.

Another scenario is when the student asks the tutor a question. Questions could be ask for clarification of what has just been presented to them. Thirdly, it is also possible for the tutor to start quizzing the student during an example. This would allow the tutor to assess if the material is being absorbed by the student.

For the examples in which the student participated, if student gave the answers directly (see example Equation A.11 and example Equation A.12, and the tutor did not further explain/correct/demonstrate the solution steps, then the episode is NOT a worked-out example.

(A.12) LOW: how (a)bout this guy? #
103: it looks good.
LOW: looks good.

For a worked out example, the solution may be given by the student, in that case, the example did not end when the solution was given. The example ended after the tutor gave solution steps (see example Equation A.13.

(A.13) LOW: is this a binary search tree?
LOW: you see the violation of that property anywhere or does it hold anywhere?
103: right here.
LOW: right.
LOW: so it is not correct?
103: yeah.
LOW: so left is less l@l l@l.
LOW: left is less.
LOW: so this is wrong.
LOW: it shouldn't be to the right.

# Appendix B

# ANALOGY CODING MANUAL

*Primary authors: Rachel Harsley (rharsl2@uic.edu) and Mehrdad Alizadeh (maliza2@uic.edu)*

## B.1 Definition

### B.1.1 Analogy

Gentner defines analogies as (Graham and Bechtel, 1998): partial similarities between different situations that support further inferences. Specifically, analogy is a kind of similarity in which the same system of relations holds across different objects. Analogies thus capture parallels across different situations. The analogical process is described as follows (Gentner and Colhoun, 2010):

- retrieval: given some current situation in working memory, a prior similar or analogous example may be retrieved from long term memory
- mapping: given two cases in working memory, mapping consists of aligning their representational structures to derive the commonalities and projecting inferences from one analog to the other.
- evaluation of the analogy and its inferences
- abstraction of the structure common to both analogies.
- rerepresentation: adaptation or of one or both representations to improve the match.

### B.1.2 Analogous Terms

Analogous terms are the words used to describe an analogy that are not the terminology common to the primary domain. Examples:

- person is an analogous term - We are going to tell person@F ok you are going to point to person@E
- point *is not* an analogous term - We are going to tell @F ok you are going to point to @E
- in front of is an analogous term - If we think of each of these as a person in line, person c@1 knows whos in front of him

If a term can be used in both the analogy and the primary domain, the term is not considered to be an analogous term unless the instructor explicitly describes a mapping. Examples include:

- point *is not* an analogous term - We are going to tell @F ok you are going to point to @E
- element is an analogous term - like a grocery list or whatever. uh so it has got a first element, second element, third element, fourth element and so on.

# Appendix B (Continued)

## B.2 Coding Category

Analogies are to be primarily marked with a single coding group, analogy. The analogy code group contains 3 codes:

- begin-analogy: The beginning of an analogy episode.
  - The beginning of the analogy is marked when either of the following occur
    - the prior similar or analogous example is first mentioned
    - the prior similar or analogous example is revisited after a prior analogy ending (new mapping).
- end-analogy: The end of an analogy episode
  - The end of the analogy is marked when either of the following occurs (in order of precedence)
    - when the current sessions problem is revisited without the analogous terms introduced in the analogy for more than ten (n>10) utterances [1]
    - a new problem is introduced without the analogous terms introduced in the analogy for more than ten (n>10) utterances*
    - the prior similar or analogous example is last mentioned
- single-analogy:
  - If an analogy starts and finishes in one turn, this code marks that turn as an analogy
  - If the prior similar or analogous example is revisited in a single line

## B.3 Marking Analogies

### B.3.1 Outline

An analogy must possess the following:

- Citing of a prior similar or analogous example to a current topic
- Explanation of at least one parallel characteristic between the topic and example (mapping)

In addition to the required attributes, an analogy may additionally possess the following:

- Description of where the analogy holds and does not hold in comparison to the current topic
- Modification of the analogy to fit the evaluated shortcomings in comparison

[1] end-analogy should be marked at the location where the count of utterances first began

## Appendix B (Continued)

### B.3.2 Examples

Binary search tree is explained using family tree analogy (Example Equation B.1).

(B.1) 440 [00:22:14.14]LOW: *uh a binary tree is kind of like mother and father and xxx
441 [00:22:19.26]109:a family tree.
442 [00:22:20.26]LOW: no that's not bad *uh that's bad.
443 [00:22:22.19]LOW: it's +// because families can have more than two kids.
444 [00:22:26.00]LOW: so here what it means is that binary is that each node can have two trees, two children.
445 [00:22:29.18]109:+< two kids.

Linked Lists are explained using a grocery list analogy (Example Equation B.2).

(B.2) 010 [00:00:17.24]*LOW: like a grocery list or whatever.
011 [00:00:19.18]*LOW: uh so it has got a first element, second element, third element, fourth element and so on.

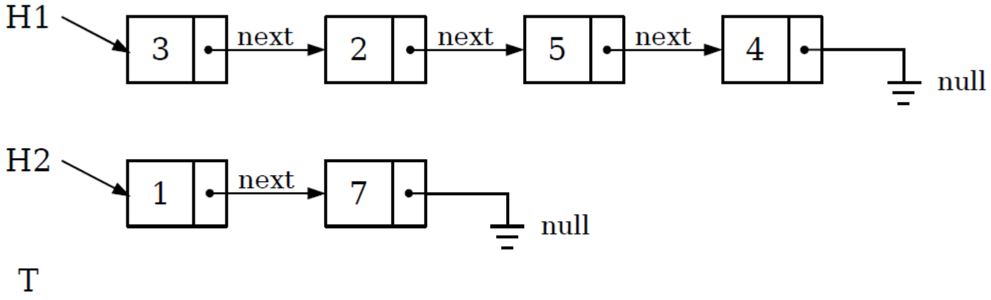
Stacks are explained using a stack of legos analogy (Example Equation B.3).

(B.3) 219 [00:22:07.07]JAC: think of the stack as a bunch of legos, okay?
220 [00:22:14.09]JAC: and each time you put out a lego +...
221 [00:22:20.01]JAC: okay <we'll call this>[//] we'll just go a@l b@l c@l d@l e@l, and so forth.
222 [00:22:29.12]JAC: okay?
223 [00:22:30.12]JAC: so we're stacking our legos up.
224 [00:22:33.03]JAC: if we want to take a lego off we can only take the lego off that we just inserted.
225 [00:22:42.18]JAC: right?
226 [00:22:43.06]JAC: (be)cause we're building from the bottom up.
227 [00:22:47.04]JAC: okay?
228 [00:22:48.10]JAC: so we can only <take in>[//] take off whatever we put in last.

# Appendix C

## CHIQAT-TUTOR PRE/POST TEST

You have the following two linked lists, starting from the head pointers H1 and H2. You also have a temporary pointer T.



1. Look at the following procedure. The procedure is written in pseudo C/C++/Java, but dont worry about programming details such as declarations etc. What is the status of the data structures after its execution? Draw a picture representing them.

```
T = H2;
while (T.next != null) {
   T = T.next;
}
T.next = H1;
```

2. Consider the following variation of the same procedure. Why doesn't it work?

```
T = H2;
while (T != null) {
   T = T.next;
}
T.next = H1;
```

3. Write a sequence of operations, in pseudo-code or in a programming language of your choice, that moves the first node of the original list H1 to the end of the list.

Originally designed and used in (Fossati, 2009a)

# Appendix D

## CHIQAT-TUTOR PRE/POST TEST GRADING SCHEME

| Question | Score | Criteria |
|---|---|---|
| 1 | 5 | The solution is correct and complete. |
| | 4 | The solution is correct, but something minor is missing (e.g. the state of temporary pointers, or the null pointer at the end of the lists) |
| | 3 | There are some minor mistakes. The overall solution reveals some understanding of the topic. |
| | 2 | There are some serious mistakes, revealing flaws in the understanding of the topic. |
| | 1 | The solution is totally wrong or garbled. |
| | 0 | The answer is blank. |
| 2 | 5 | The explanation is correct, complete, and well formulated. |
| | 4 | The explanation is correct but not so well formulated. |
| | 3 | The explanation makes some sense, but it is somewhat fuzzy. |
| | 2 | The explanation does not make much sense. |
| | 1 | The explanation is wrong or garbled. |
| | 0 | The answer is blank. |
| 3 | 5 | The solution is completely correct. |
| | 4 | There is one mistake. |
| | 3 | There are two mistakes. |
| | 2 | There are some mistakes, but the solution makes some sense. |
| | 1 | The solution is totally wrong or garbled. |
| | 0 | The answer is blank. |

Originally designed and used in (Fossati, 2009a)

# Appendix E

## CHIQAT-TUTOR LOG MESSAGES AND FEATURES

A list of all messages, either logged during experiments or derived from log data, is given in Table XXXIX. Messages including [p] indicate that the message is related to a problem, and [s] is related to a step number within a WOE for the first problem.

| Log Message | Description |
|---|---|
| All WOEs Complete [p] | 1 if all executed WOEs were completed in a problem |
| Application Window Maximized | Total count of times application window was maximised |
| Application Window Maximized [p] | Count of times application window was maximised in a problem |
| Application Window Minimized | Total count of times application window was minimised |
| Application Window Minimized [p] | Count of times application window was minimised in a problem |
| Attempts [p] | Attempts made at a problem |
| Attempts After WOE [p] | Attempts of a problem made after a WOE was used |
| Avg Step Duration [p] | Average WOE step duration for a given problem |
| Avg WOE Duration [p] | Average duration of WOEs for a given problem |
| Avg WOE Steps [p] | Average number of steps used in WOEs for a problem |
| Bad Submissions [p] | Number of bad problem submissions made for a problem |
| Collection Duration | Total session duration in milliseconds |
| Command Editor Paste | Number of times text had been pasted into the block text editor |
| Command Editor Paste [p] | Number of times text had been pasted into the block text editor on a problem |
| Command Line Paste | Number of times text had been pasted into the block text editor |
| Command Line Paste [p] | Number of times text had been pasted into the block text editor on a problem |
| Completed WOEs [p] | Count of WOEs completed for for a problem |
| Date | Date of session |
| Example Completed | Number of completed examples |
| Example Completed [p] | Number of completed examples for a problem |
| Example Quit | Number of times examples were terminated |
| Example Quit [p] | Number of times examples were terminated on a problem |
| Example Requested | Number of examples requested by the user |
| Example Requested [p] | Number of examples requested by the user on a problem |
| Example Started | Number of examples started |

## Appendix E (Continued)

| | |
|---|---|
| Example Started [p] | Number of examples started per problem |
| Example Timed Out (Step) | Count of example step time outs |
| Example Timed Out (Step) [p] | Count of example step time outs for a problem |
| Example Timed Out (Whole) | Count of example time outs |
| Example Timed Out (Whole) [p] | Count of example time outs for a problem |
| Examples Duration [p] | Total duration of examples for a problem |
| Examples Used [p] | Number of examples used for a problem |
| Executing Example Step | Count of WOE steps made |
| Executing Example Step [p] | Count of WOE steps made for a problem |
| First Log Time | The time stamp of the first log message |
| First WOE Complete [p] | 1 if the WOE executed WOE on a problem is completed, otherwise 0 |
| First WOE Step Dur [s] | First step duration for a given step in a WOE on problem 1 |
| First Mean Conclusion Step Dur | Mean duration of conclusion steps in problem 1 |
| First Mean Definition Step Dur | Mean duration of definition steps in problem 1 |
| First Mean Explanation Step Dur | Mean duration of explanation steps in problem 1 |
| First Mean Intro Step Dur | Mean duration of introduction steps in problem 1 |
| First Mean Operation Step Dur | Mean duration of operation steps in problem 1 |
| First Mean Pause Step Dur | Mean duration of pause steps in problem 1 |
| Good Submissions [p] | Number of good submissions made for a problem |
| Incomplete WOEs [p] | Number of incomplete WOEs for a problem |
| Last Log Time | The last log message time stamp |
| Last Problem Attempted | The last problem attempted by the user |
| Lesson Started | Number of times a ChiQat lesson had been started |
| Lesson Tutorial Requested | Number of times the lesson tutorial was requested |
| Longest WOE Step Dur [s] | Longest duration of a given step taken in problem 1 |
| Mean WOE Step Dur [s] | Mean duration of a step in problem 1 |
| Negative Feedback Given | Number of times negative feedback had been given |
| Negative Feedback Given [p] | Number of times negative feedback had been given on a problem |
| Negative Question Answer Response | Number of times a negative response was given to a question answer |
| Negative Question Answer Response [p] | Number of times a negative response was given to a question answer for a problem |
| Negative Answer Response [p] | Count of negative answers |
| Negative Feedback [p] | Count of negative answers for a problem |
| Neutral Feedback Given | Number of times neutral feedback had been given |
| Neutral Feedback Given [p] | Number of times neutral feedback had been given for a problem |
| Neutral Question Answer Response | Count of neutral responses to questions |

## Appendix E (Continued)

| | |
|---|---|
| Neutral Question Answer Response [p] | Count of neutral responses to questions for a problem |
| New Problem Selected | Number of times new problems were selected |
| New Problem Selected [p] | Number of times new problems were selected during a problem |
| Node Clicked | Times a graphical linked list node was clicked |
| Node Clicked [p] | Times a graphical linked list node was clicked for a problem |
| Operation Execution Submitted | Number of times an operation was submitted |
| Operation Execution Submitted [p] | Number of times an operation was submitted for a problem |
| Operation Redone | Count of redo operations made |
| Operation Redone [p] | Count of redo operations made for a problem |
| Operation Undone | Count of undo operations made |
| Operation Undone [p] | Count of redo operations made for a problem |
| Positive Feedback Given | Count of positive feedbacks given |
| Positive Feedback Given [p] | Count of positive feedbacks given for a problem |
| Positive Question Answer Response | Count of positive answer responses |
| Positive Question Answer Response [p] | Count of positive answer responses for a problem |
| Positive Feedback [p] | Count of positive feedback |
| Problem List Requested | Number of times the problem list has been requested |
| Problem List Requested [p] | Number of times the problem list has been requested during a problem |
| Problem Restarted | Number of times a problem has been restarted |
| Problem Restarted [p] | Number of times a problem has been restarted during a problem |
| Problem Duration [p] | Problem duration in milliseconds |
| Proportion Completed WOEs [p] | Proportion of WOEs completed during a problem (measured as $\frac{CompletedExamples}{TotalExamples}$) |
| Question Given | Count of questions given to the user |
| Question Given [p] | Count of questions given to the user for a problem |
| Redo Operation Requested | Redo operations requested by the user |
| Redo Operation Requested [p] | Redo operations requested by the user for a problem |
| Scratch Pad Next | Number of times the scratch pad has been moved to the next page |
| Scratch Pad Next [p] | Number of times the scratch pad has been moved to the next page for a problem |
| Scratch Pad Prev | Number of times the scratch pad has been moved to the previous page |
| Scratch Pad Prev [p] | Number of times the scratch pad has been moved to the next page for a problem |
| Solution Correct | Number of correct solutions made |

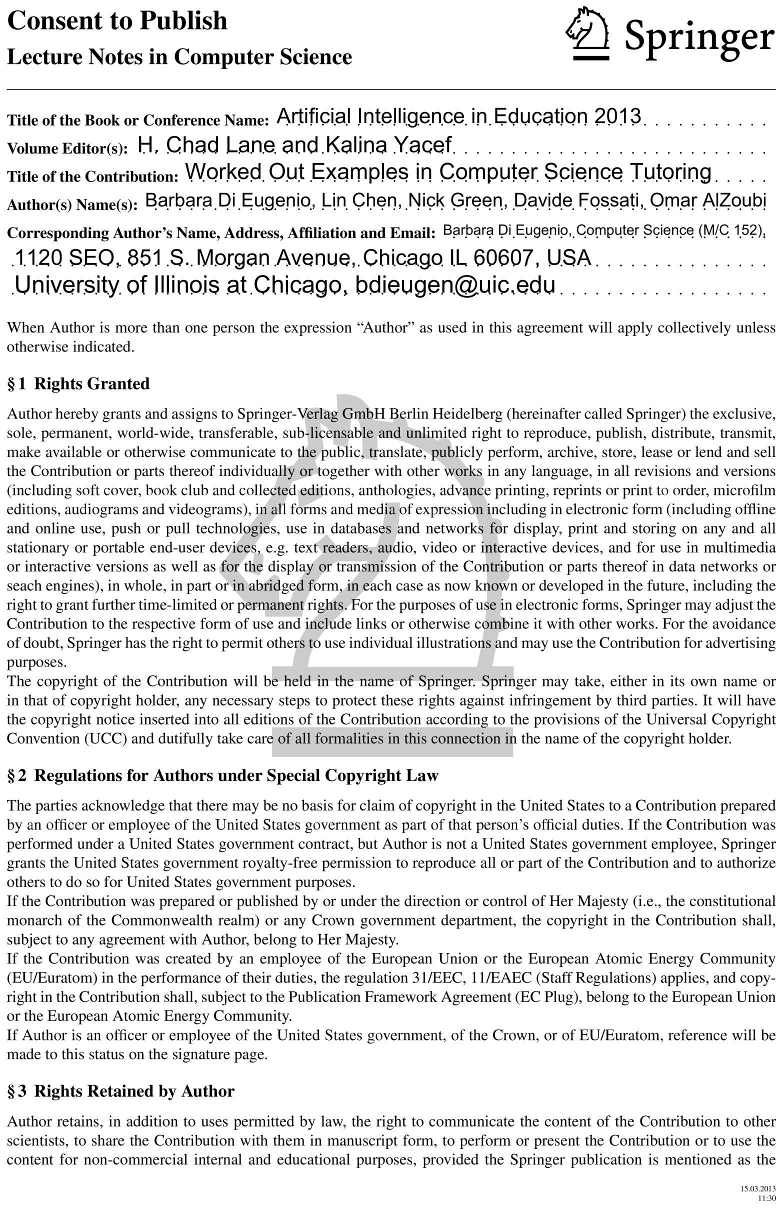
## Appendix E (Continued)

| | |
|---|---|
| Solution Correct [p] | Number of correct solutions made for a problem |
| Solution Incorrect | Number of incorrect solutions submitted |
| Solution Incorrect [p] | Number of incorrect solutions submitted for a problem |
| Solved Problems | Number of problems solved |
| Starting Lesson Tutorial | Number of times a lesson tutorial has been started |
| StDev Step Duration [p] | Standard deviation of step durations for a problem |
| StDev WOE Duration [p] | Standard deviation of WOE durations for a problem |
| StDev WOE Steps [p] | Standard deviation of number of step made for a problem |
| Success Attempt [p] | Number of successful attempts for a problem |
| Template Help Item Selected | Number of times the template help menu is used |
| Template Help Item Selected [p] | Number of times the template help menu is used for a problem |
| Template Help Requested | Number of times the template help menu is requested |
| Template Help Requested [p] | Number of times the template help menu is requested for a problem |
| Total WOE Duration [p] | Total duration of WOEs for a problem |
| Tutorial Duration | Total duration of all WOEs |
| Tutorial Views | Number of system tutorial views |
| Undo Operation Requested | Time an undo operation has been requested by the user |
| Undo Operation Requested [p] | Time an undo operation has been requested by the user for a problem |
| User Acknowledge | Total number of user acknowledgements |
| User Acknowledge [p] | Total number of user acknowledgements for a problem |
| User logged in | Count of user log ins |
| User Logged Out | Count of user log outs |
| User Logged Out [p] | Count of user log outs during a problem |
| WOE Exit Step Count [s] | Number of times a WOE has been terminated on a given step in problem 1 |

TABLE XXXIX: Messages Logged for Analysis (including derived)

# Appendix F

## COPYRIGHT - AIED 2013



### Consent to Publish

**Lecture Notes in Computer Science**

Springer

**Title of the Book or Conference Name:** Artificial Intelligence in Education 2013

**Volume Editor(s):** H. Chad Lane and Kalina Yacef

**Title of the Contribution:** Worked Out Examples in Computer Science Tutoring

**Author(s) Name(s):** Barbara Di Eugenio, Lin Chen, Nick Green, Davide Fossati, Omar AlZoubi

**Corresponding Author's Name, Address, Affiliation and Email:** Barbara Di Eugenio, Computer Science (M/C 152),
1120 SEO, 851 S. Morgan Avenue, Chicago IL 60607, USA
University of Illinois at Chicago, bdieugen@uic.edu

When Author is more than one person the expression "Author" as used in this agreement will apply collectively unless otherwise indicated.

**§ 1 Rights Granted**

Author hereby grants and assigns to Springer-Verlag GmbH Berlin Heidelberg (hereinafter called Springer) the exclusive, sole, permanent, world-wide, transferable, sub-licensable and unlimited right to reproduce, publish, distribute, transmit, make available or otherwise communicate to the public, translate, publicly perform, archive, store, lease or lend and sell the Contribution or parts thereof individually or together with other works in any language, in all revisions and versions (including soft cover, book club and collected editions, anthologies, advance printing, reprints or print to order, microfilm editions, audiograms and videograms), in all forms and media of expression including in electronic form (including offline and online use, push or pull technologies, use in databases and networks for display, print and storing on any and all stationary or portable end-user devices, e.g. text readers, audio, video or interactive devices, and for use in multimedia or interactive versions as well as for the display or transmission of the Contribution or parts thereof in data networks or seach engines), in whole, in part or in abridged form, in each case as now known or developed in the future, including the right to grant further time-limited or permanent rights. For the purposes of use in electronic forms, Springer may adjust the Contribution to the respective form of use and include links or otherwise combine it with other works. For the avoidance of doubt, Springer has the right to permit others to use individual illustrations and may use the Contribution for advertising purposes.

The copyright of the Contribution will be held in the name of Springer. Springer may take, either in its own name or in that of copyright holder, any necessary steps to protect these rights against infringement by third parties. It will have the copyright notice inserted into all editions of the Contribution according to the provisions of the Universal Copyright Convention (UCC) and dutifully take care of all formalities in this connection in the name of the copyright holder.

**§ 2 Regulations for Authors under Special Copyright Law**

The parties acknowledge that there may be no basis for claim of copyright in the United States to a Contribution prepared by an officer or employee of the United States government as part of that person's official duties. If the Contribution was performed under a United States government contract, but Author is not a United States government employee, Springer grants the United States government royalty-free permission to reproduce all or part of the Contribution and to authorize others to do so for United States government purposes.

If the Contribution was prepared or published by or under the direction or control of Her Majesty (i.e., the constitutional monarch of the Commonwealth realm) or any Crown government department, the copyright in the Contribution shall, subject to any agreement with Author, belong to Her Majesty.

If the Contribution was created by an employee of the European Union or the European Atomic Energy Community (EU/Euratom) in the performance of their duties, the regulation 31/EEC, 11/EAEC (Staff Regulations) applies, and copyright in the Contribution shall, subject to the Publication Framework Agreement (EC Plug), belong to the European Union or the European Atomic Energy Community.

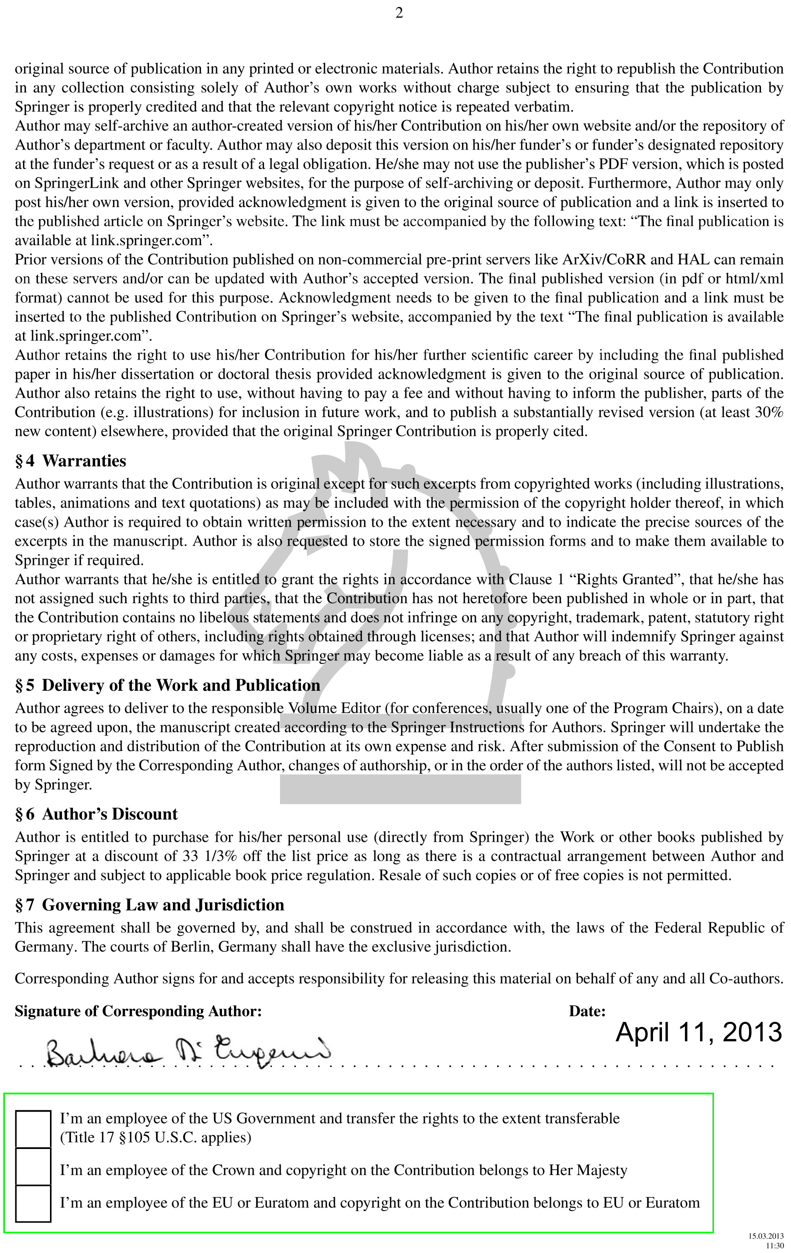
If Author is an officer or employee of the United States government, of the Crown, or of EU/Euratom, reference will be made to this status on the signature page.

**§ 3 Rights Retained by Author**

Author retains, in addition to uses permitted by law, the right to communicate the content of the Contribution to other scientists, to share the Contribution with them in manuscript form, to perform or present the Contribution or to use the content for non-commercial internal and educational purposes, provided the Springer publication is mentioned as the

15.03.2013
11:30

# Appendix F (Continued)



original source of publication in any printed or electronic materials. Author retains the right to republish the Contribution in any collection consisting solely of Author's own works without charge subject to ensuring that the publication by Springer is properly credited and that the relevant copyright notice is repeated verbatim.

Author may self-archive an author-created version of his/her Contribution on his/her own website and/or the repository of Author's department or faculty. Author may also deposit this version on his/her funder's or funder's designated repository at the funder's request or as a result of a legal obligation. He/she may not use the publisher's PDF version, which is posted on SpringerLink and other Springer websites, for the purpose of self-archiving or deposit. Furthermore, Author may only post his/her own version, provided acknowledgment is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be accompanied by the following text: "The final publication is available at link.springer.com".

Prior versions of the Contribution published on non-commercial pre-print servers like ArXiv/CoRR and HAL can remain on these servers and/or can be updated with Author's accepted version. The final published version (in pdf or html/xml format) cannot be used for this purpose. Acknowledgment needs to be given to the final publication and a link must be inserted to the published Contribution on Springer's website, accompanied by the text "The final publication is available at link.springer.com".

Author retains the right to use his/her Contribution for his/her further scientific career by including the final published paper in his/her dissertation or doctoral thesis provided acknowledgment is given to the original source of publication. Author also retains the right to use, without having to pay a fee and without having to inform the publisher, parts of the Contribution (e.g. illustrations) for inclusion in future work, and to publish a substantially revised version (at least 30% new content) elsewhere, provided that the original Springer Contribution is properly cited.

### § 4 Warranties

Author warrants that the Contribution is original except for such excerpts from copyrighted works (including illustrations, tables, animations and text quotations) as may be included with the permission of the copyright holder thereof, in which case(s) Author is required to obtain written permission to the extent necessary and to indicate the precise sources of the excerpts in the manuscript. Author is also requested to store the signed permission forms and to make them available to Springer if required.

Author warrants that he/she is entitled to grant the rights in accordance with Clause 1 "Rights Granted", that he/she has not assigned such rights to third parties, that the Contribution has not heretofore been published in whole or in part, that the Contribution contains no libelous statements and does not infringe on any copyright, trademark, patent, statutory right or proprietary right of others, including rights obtained through licenses; and that Author will indemnify Springer against any costs, expenses or damages for which Springer may become liable as a result of any breach of this warranty.

### § 5 Delivery of the Work and Publication

Author agrees to deliver to the responsible Volume Editor (for conferences, usually one of the Program Chairs), on a date to be agreed upon, the manuscript created according to the Springer Instructions for Authors. Springer will undertake the reproduction and distribution of the Contribution at its own expense and risk. After submission of the Consent to Publish form Signed by the Corresponding Author, changes of authorship, or in the order of the authors listed, will not be accepted by Springer.

### § 6 Author's Discount

Author is entitled to purchase for his/her personal use (directly from Springer) the Work or other books published by Springer at a discount of 33 1/3% off the list price as long as there is a contractual arrangement between Author and Springer and subject to applicable book price regulation. Resale of such copies or of free copies is not permitted.

### § 7 Governing Law and Jurisdiction

This agreement shall be governed by, and shall be construed in accordance with, the laws of the Federal Republic of Germany. The courts of Berlin, Germany shall have the exclusive jurisdiction.

Corresponding Author signs for and accepts responsibility for releasing this material on behalf of any and all Co-authors.

**Signature of Corresponding Author:** **Date:**

Barbara Di Eugenio ........................................ April 11, 2013

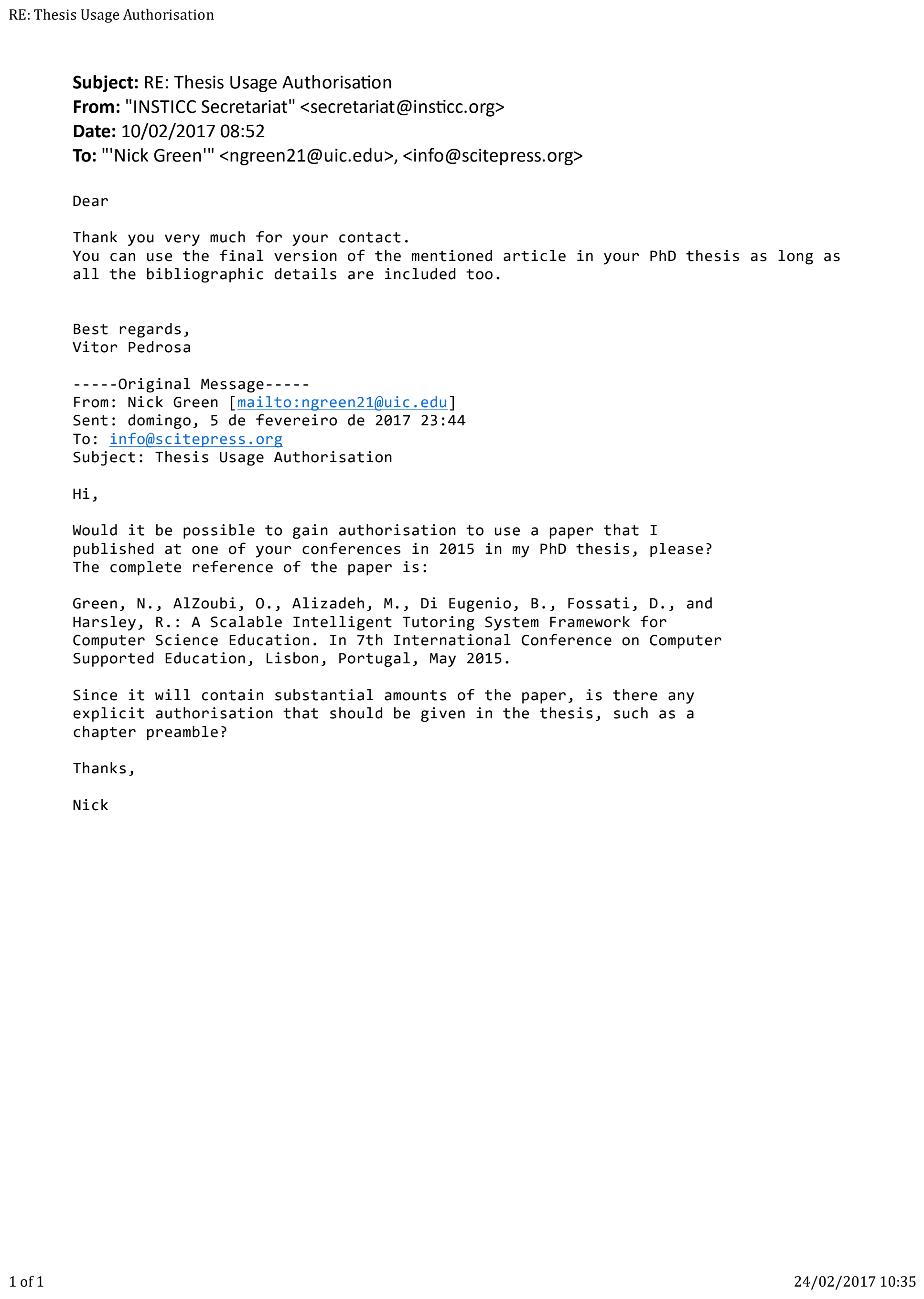
□ I'm an employee of the US Government and transfer the rights to the extent transferable (Title 17 §105 U.S.C. applies)

□ I'm an employee of the Crown and copyright on the Contribution belongs to Her Majesty

□ I'm an employee of the EU or Euratom and copyright on the Contribution belongs to EU or Euratom

# Appendix G

## COPYRIGHT AUTHORISATION - CSEDU 2015



RE: Thesis Usage Authorisation

**Subject:** RE: Thesis Usage Authorisation
**From:** "INSTICC Secretariat" <secretariat@insticc.org>
**Date:** 10/02/2017 08:52
**To:** "'Nick Green'" <ngreen21@uic.edu>, <info@scitepress.org>

Dear

Thank you very much for your contact.
You can use the final version of the mentioned article in your PhD thesis as long as all the bibliographic details are included too.

Best regards,
Vitor Pedrosa

-----Original Message-----
From: Nick Green [mailto:ngreen21@uic.edu]
Sent: domingo, 5 de fevereiro de 2017 23:44
To: info@scitepress.org
Subject: Thesis Usage Authorisation

Hi,

Would it be possible to gain authorisation to use a paper that I published at one of your conferences in 2015 in my PhD thesis, please? The complete reference of the paper is:

Green, N., AlZoubi, O., Alizadeh, M., Di Eugenio, B., Fossati, D., and Harsley, R.: A Scalable Intelligent Tutoring System Framework for Computer Science Education. In 7th International Conference on Computer Supported Education, Lisbon, Portugal, May 2015.

Since it will contain substantial amounts of the paper, is there any explicit authorisation that should be given in the thesis, such as a chapter preamble?

Thanks,

Nick

1 of 1 24/02/2017 10:35

# Appendix H

# COPYRIGHT - ICEDUTECH 2015

### Copyright Transfer Agreement / Applied Computing (AC 2015) Proceedings

**Title of Work:** ____________________________________________________

____________________________________________________

**Author(s):** ____________________________________________________

____________________________________________________

____________________________________________________

____________________________________________________

____________________________________________________

____________________________________________________

Copyright to the above work (including, without limitation, the right to publish the work in whole or in part in any and all forms and media, now or hereafter known) is hereby transferred to the International Association for Development of the Information Society (hereinafter IADIS), effective as of the date of this agreement, on the understanding that the work has been accepted for presentation at a meeting sponsored by IADIS and for publication in the proceedings of that meeting.

However, each of the authors and the employers for whom the work was performed reserve all other rights, specifically including the following:

(1) All proprietary rights other than copyright and the publication rights transferred to IADIS;
(2) The right to publish in a journal or collection or to be used in future works of the author's own (such as articles or books) all or part of this work, provided that acknowledgement is given to IADIS and a full citation to its publication in the particular proceedings is included;
(3) The right to make oral presentation of the material in any forum;
(4) The right to make copies of the work for internal distribution within the author's organization and for external distribution as a preprint, reprint, technical report, or related class of document. In the case of a work prepared under a government contract, if the contract so requires, that government may reproduce all or portions of the article and may authorize others to do so, for official government purposes only.

**Date:** _______________

**Signature(s):** ____________________________________________________

**Name(s)** ____________________________________________________

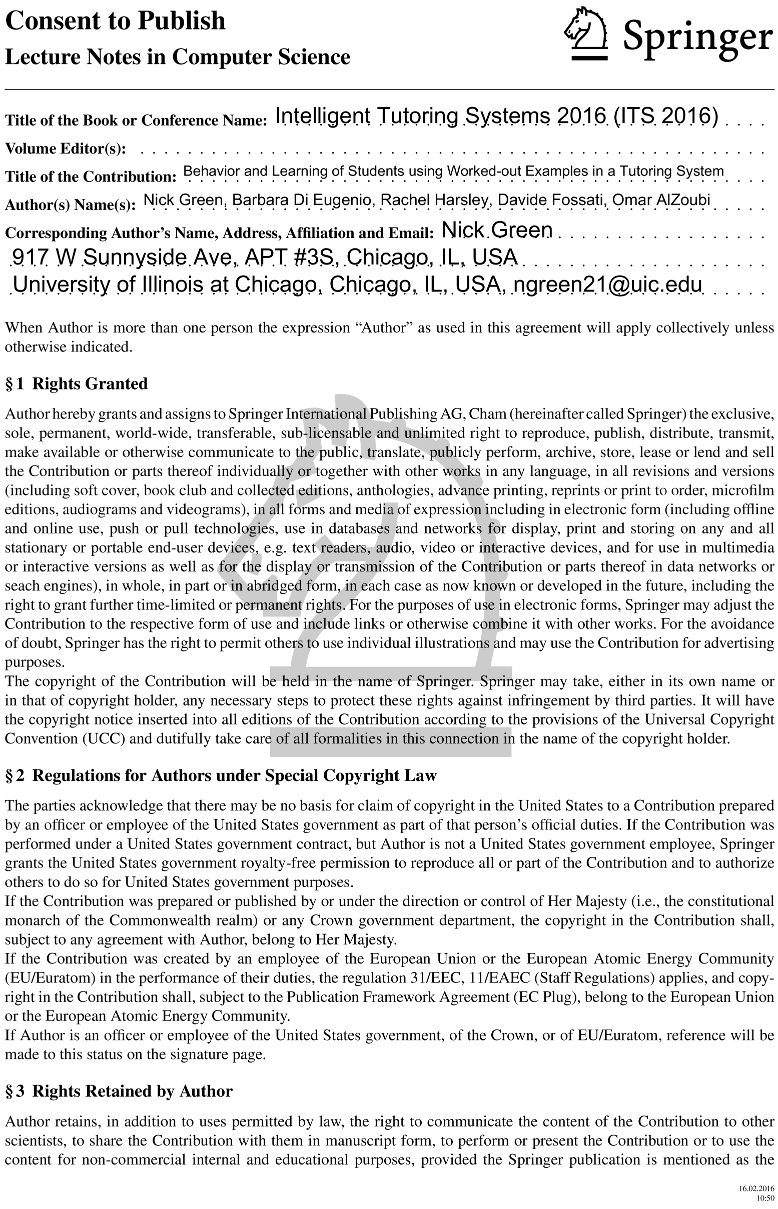
(please print) (official representative if work is done “for hire”)

Please send by email the signed agreement to the AC 2015 Conference Secretariat:

IADIS,
secretariat@computing-conf.org

# Appendix I

## COPYRIGHT - ITS 2016



### Consent to Publish

**Lecture Notes in Computer Science**

Springer

**Title of the Book or Conference Name:** Intelligent Tutoring Systems 2016 (ITS 2016)
**Volume Editor(s):**
**Title of the Contribution:** Behavior and Learning of Students using Worked-out Examples in a Tutoring System
**Author(s) Name(s):** Nick Green, Barbara Di Eugenio, Rachel Harsley, Davide Fossati, Omar AlZoubi
**Corresponding Author's Name, Address, Affiliation and Email:** Nick Green
917 W Sunnyside Ave, APT #3S, Chicago, IL, USA
University of Illinois at Chicago, Chicago, IL, USA, ngreen21@uic.edu

When Author is more than one person the expression "Author" as used in this agreement will apply collectively unless otherwise indicated.

#### § 1 Rights Granted

Author hereby grants and assigns to Springer International Publishing AG, Cham (hereinafter called Springer) the exclusive, sole, permanent, world-wide, transferable, sub-licensable and unlimited right to reproduce, publish, distribute, transmit, make available or otherwise communicate to the public, translate, publicly perform, archive, store, lease or lend and sell the Contribution or parts thereof individually or together with other works in any language, in all revisions and versions (including soft cover, book club and collected editions, anthologies, advance printing, reprints or print to order, microfilm editions, audiograms and videograms), in all forms and media of expression including in electronic form (including offline and online use, push or pull technologies, use in databases and networks for display, print and storing on any and all stationary or portable end-user devices, e.g. text readers, audio, video or interactive devices, and for use in multimedia or interactive versions as well as for the display or transmission of the Contribution or parts thereof in data networks or seach engines), in whole, in part or in abridged form, in each case as now known or developed in the future, including the right to grant further time-limited or permanent rights. For the purposes of use in electronic forms, Springer may adjust the Contribution to the respective form of use and include links or otherwise combine it with other works. For the avoidance of doubt, Springer has the right to permit others to use individual illustrations and may use the Contribution for advertising purposes.
The copyright of the Contribution will be held in the name of Springer. Springer may take, either in its own name or in that of copyright holder, any necessary steps to protect these rights against infringement by third parties. It will have the copyright notice inserted into all editions of the Contribution according to the provisions of the Universal Copyright Convention (UCC) and dutifully take care of all formalities in this connection in the name of the copyright holder.

#### § 2 Regulations for Authors under Special Copyright Law

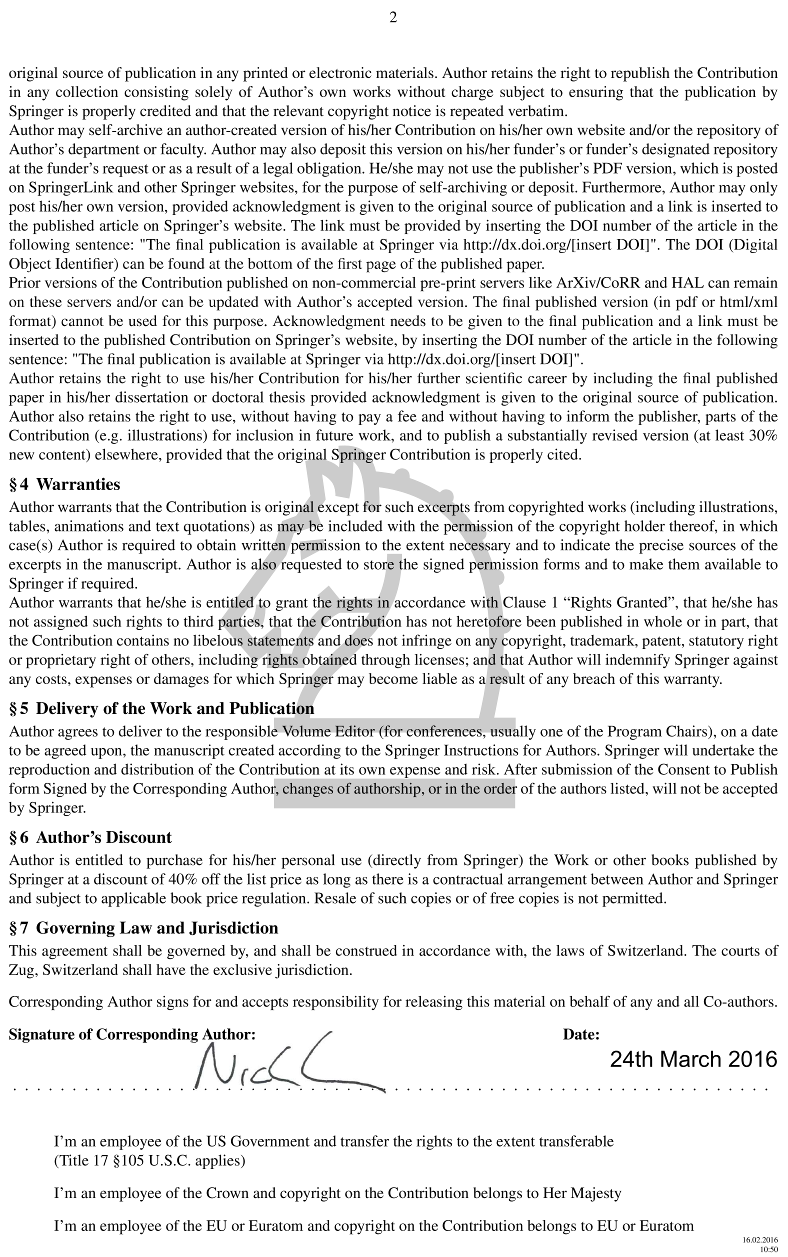
The parties acknowledge that there may be no basis for claim of copyright in the United States to a Contribution prepared by an officer or employee of the United States government as part of that person's official duties. If the Contribution was performed under a United States government contract, but Author is not a United States government employee, Springer grants the United States government royalty-free permission to reproduce all or part of the Contribution and to authorize others to do so for United States government purposes.
If the Contribution was prepared or published by or under the direction or control of Her Majesty (i.e., the constitutional monarch of the Commonwealth realm) or any Crown government department, the copyright in the Contribution shall, subject to any agreement with Author, belong to Her Majesty.
If the Contribution was created by an employee of the European Union or the European Atomic Energy Community (EU/Euratom) in the performance of their duties, the regulation 31/EEC, 11/EAEC (Staff Regulations) applies, and copyright in the Contribution shall, subject to the Publication Framework Agreement (EC Plug), belong to the European Union or the European Atomic Energy Community.
If Author is an officer or employee of the United States government, of the Crown, or of EU/Euratom, reference will be made to this status on the signature page.

#### § 3 Rights Retained by Author

Author retains, in addition to uses permitted by law, the right to communicate the content of the Contribution to other scientists, to share the Contribution with them in manuscript form, to perform or present the Contribution or to use the content for non-commercial internal and educational purposes, provided the Springer publication is mentioned as the

16.02.2016
10:50

# Appendix I (Continued)



original source of publication in any printed or electronic materials. Author retains the right to republish the Contribution in any collection consisting solely of Author's own works without charge subject to ensuring that the publication by Springer is properly credited and that the relevant copyright notice is repeated verbatim.

Author may self-archive an author-created version of his/her Contribution on his/her own website and/or the repository of Author's department or faculty. Author may also deposit this version on his/her funder's or funder's designated repository at the funder's request or as a result of a legal obligation. He/she may not use the publisher's PDF version, which is posted on SpringerLink and other Springer websites, for the purpose of self-archiving or deposit. Furthermore, Author may only post his/her own version, provided acknowledgment is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be provided by inserting the DOI number of the article in the following sentence: "The final publication is available at Springer via http://dx.doi.org/[insert DOI]". The DOI (Digital Object Identifier) can be found at the bottom of the first page of the published paper.

Prior versions of the Contribution published on non-commercial pre-print servers like ArXiv/CoRR and HAL can remain on these servers and/or can be updated with Author's accepted version. The final published version (in pdf or html/xml format) cannot be used for this purpose. Acknowledgment needs to be given to the final publication and a link must be inserted to the published Contribution on Springer's website, by inserting the DOI number of the article in the following sentence: "The final publication is available at Springer via http://dx.doi.org/[insert DOI]".

Author retains the right to use his/her Contribution for his/her further scientific career by including the final published paper in his/her dissertation or doctoral thesis provided acknowledgment is given to the original source of publication. Author also retains the right to use, without having to pay a fee and without having to inform the publisher, parts of the Contribution (e.g. illustrations) for inclusion in future work, and to publish a substantially revised version (at least 30% new content) elsewhere, provided that the original Springer Contribution is properly cited.

### § 4 Warranties

Author warrants that the Contribution is original except for such excerpts from copyrighted works (including illustrations, tables, animations and text quotations) as may be included with the permission of the copyright holder thereof, in which case(s) Author is required to obtain written permission to the extent necessary and to indicate the precise sources of the excerpts in the manuscript. Author is also requested to store the signed permission forms and to make them available to Springer if required.

Author warrants that he/she is entitled to grant the rights in accordance with Clause 1 "Rights Granted", that he/she has not assigned such rights to third parties, that the Contribution has not heretofore been published in whole or in part, that the Contribution contains no libelous statements and does not infringe on any copyright, trademark, patent, statutory right or proprietary right of others, including rights obtained through licenses; and that Author will indemnify Springer against any costs, expenses or damages for which Springer may become liable as a result of any breach of this warranty.

### § 5 Delivery of the Work and Publication

Author agrees to deliver to the responsible Volume Editor (for conferences, usually one of the Program Chairs), on a date to be agreed upon, the manuscript created according to the Springer Instructions for Authors. Springer will undertake the reproduction and distribution of the Contribution at its own expense and risk. After submission of the Consent to Publish form Signed by the Corresponding Author, changes of authorship, or in the order of the authors listed, will not be accepted by Springer.

### § 6 Author's Discount

Author is entitled to purchase for his/her personal use (directly from Springer) the Work or other books published by Springer at a discount of 40% off the list price as long as there is a contractual arrangement between Author and Springer and subject to applicable book price regulation. Resale of such copies or of free copies is not permitted.

### § 7 Governing Law and Jurisdiction

This agreement shall be governed by, and shall be construed in accordance with, the laws of Switzerland. The courts of Zug, Switzerland shall have the exclusive jurisdiction.

Corresponding Author signs for and accepts responsibility for releasing this material on behalf of any and all Co-authors.

**Signature of Corresponding Author:** Nick **Date:** 24th March 2016

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

I'm an employee of the US Government and transfer the rights to the extent transferable
(Title 17 §105 U.S.C. applies)

I'm an employee of the Crown and copyright on the Contribution belongs to Her Majesty

I'm an employee of the EU or Euratom and copyright on the Contribution belongs to EU or Euratom

16.02.2016
10:50

# CITED LITERATURE

Aleven, V., McLaren, B. M., Sewall, J., and Koedinger, K. R.: The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In Intelligent Tutoring Systems, pages 61–70. Springer, 2006.

Aleven, V., Mclaren, B. M., Sewall, J., and Koedinger, K. R.: A new paradigm for intelligent tutoring systems: Example-tracing tutors. International Journal of Artificial Intelligence in Education, 19(2):105–154, 2009.

Aleven, V., McLaren, B. M., Sewall, J., van Velsen, M., Popescu, O., Demi, S., Ringenberg, M., and Koedinger, K. R.: Example-tracing tutors: intelligent tutor development for non-programmers. International Journal of Artificial Intelligence in Education, 26(1):224–269, 2016.

Alizadeh, M., Di Eugenio, B., Harsley, R., Green, N., Fossati, D., and AlZoubi, O.: A Study of Analogy in Computer Science Tutorial Dialogues. In CSEDU 2015, 7th International Conference on Computer Supported Education, Lisbon, Portugal, May 2015.

AlZoubi, O., Fossati, D., Di Eugenio, B., and Green, N.: ChiQat-Tutor: An Integrated Environment for Learning Recursion. In ITS-AIEDCS 2014, 12th International Conference on Intelligent Tutoring Systems (ITS), 2nd Workshop on AI-supported Education for Computer Science (AIEDCS), Honolulu, HI, June 2014. Short paper.

AlZoubi, O., Fossati, D., Di Eugenio, B., Green, N., Alizadeh, M., and Harsley, R.: A Hybrid Model for Teaching Recursion. In SIGITE 2015, 16th Annual Conference on Information Technology Education, Chicago, IL, October 2015.

Arora, R.: Comparative analysis of classification algorithms on different datasets using WEKA. International Journal of Computer Applications, 54(13), 2012.

Atkinson, R. K., Derry, S. J., Renkl, A., and Wortham, D.: Learning from examples: Instructional principles from the worked examples research. Review of educational research, 70(2):181–214, 2000.

Atkinson, R. K. and Renkl, A.: Interactive example-based learning environments: Using interactive elements to encourage effective processing of worked examples. Educational Psychology Review, 19(3):375–386, 2007.

Badaracco, M. and Martnez, L.: An intelligent tutoring system architecture for competency-based learning. In Knowlege-based and intelligent information and engineering systems, pages 124–133. Springer, 2011.

Baker, R. S., Corbett, A. T., Koedinger, K. R., and Wagner, A. Z.: Off-task behavior in the cognitive tutor classroom: when students game the system. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 383–390. ACM, 2004.

Beaubouef, T. and Mason, J.: Why the high attrition rate for computer science students: some thoughts and observations. ACM SIGCSE Bulletin, 37(2):103–106, 2005.

Bloom, B. S.: The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. Educational researcher, pages 4–16, 1984.

Brusilovsky, P., Schwarz, E., and Weber, G.: ELM-ART: An intelligent tutoring system on World Wide Web. In Intelligent tutoring systems, pages 261–269. Springer, 1996.

Carletta, J.: Assessing agreement on classification tasks: the kappa statistic. Computational linguistics, 22(2):249–254, 1996.

Chandler, P. and Sweller, J.: Cognitive load theory and the format of instruction. Cognition and instruction, 8(4):293–332, 1991.

Chang, C.-C. and Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3):27, 2011.

Chaubal, C.: The architecture of vmware esxi. VMware White Paper, 1(7), 2008.

Chen, L., Di Eugenio, B., Fossati, D., Ohlsson, S., and Cosejo, D.: Exploring effective dialogue act sequences in one-on-one computer science tutoring dialogues. In Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications, pages 65–75. Association for Computational Linguistics, 2011.

Chi, M. T., Siler, S. A., Jeong, H., Yamauchi, T., and Hausmann, R. G.: Learning from human tutoring. Cognitive Science, 25(4):471–533, 2001.

Choudhury, R. R., Yin, H., and Fox, A.: Scale-Driven Automatic Hint Generation for Coding Style. In International Conference on Intelligent Tutoring Systems, pages 122–132. Springer, 2016.

Clark, R. C., Nguyen, F., and Sweller, J.: Efficiency in learning: Evidence-based guidelines to manage cognitive load. John Wiley & Sons, 2011.

Cohen, J.: A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1):37–46, 1960.

Cohen, P. A., Kulik, J. A., and Kulik, C.-L. C.: Educational outcomes of tutoring: A meta-analysis of findings. American educational research journal, 19(2):237–248, 1982.

Cohoon, J. M.: Toward improving female retention in the computer science major. Communications of the ACM, 44(5):108–114, 2001.

Corbett, A. T., Koedinger, K. R., and Anderson, J. R.: Intelligent tutoring systems. Handbook of humancomputer interaction, pages 849–874, 1997.

del Vado Vrseda, R., Fernndez, P., Muoz, S., and Murillo, A.: An Intelligent Tutoring System for Interactive Learning of Data Structures. In International Conference on Computational Science, pages 53–62. Springer, 2009.

Di Eugenio, B., Chen, L., Green, N., Fossati, D., and AlZoubi, O.: Worked Out Examples in Computer Science Tutoring. In 16th International Conference on Artificial Intelligence in Education, Memphis, TN, July 2013. Short paper.

Di Eugenio, B. and Glass, M.: The kappa statistic: A second look. Computational linguistics, 30(1):95–101, 2004.

Di Eugenio, B., Green, N., AlZoubi, O., Alizadeh, M., Harsley, R., and Fossati, D.: Worked-out Example in a Computer Science Intelligent Tutoring System. In 16th Annual Conference on Information Technology Education, Chicago, IL, October 2015.

Elkan, C.: Magical thinking in data mining: lessons from CoIL challenge 2000. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pages 426–431. ACM, 2001.

Ezen-Can, A. and Boyer, K. E.: Understanding student language: An unsupervised dialogue act classification approach. JEDM-Journal of Educational Data Mining, 7(1):51–78, 2015.

Fossati, D.: The role of positive feedback in Intelligent Tutoring Systems. In ACL 2008, The 46th Annual Meeting of the Association for Computational Linguistics, Student Research Workshop, Columbus, OH, June 2008.

Fossati, D.: Automatic modeling of procedural knowledge and feedback generation in a computer science tutoring system. Doctoral dissertation, University of Illinois at Chicago, 2009.

Fossati, D.: Automatic Modeling of Procedural Knowledge and Feedback Generation in a Computer Science Tutoring System. Doctoral dissertation, University of Illinois at Chicago, 2009.

Fossati, D., Di Eugenio, B., Brown, C., and Ohlsson, S.: Learning linked lists: Experiments with the iList system. In ITS 2008, The 9th International Conference on Intelligent Tutoring Systems, pages 80–89, Montreal, Canada, June 2008.

Fossati, D., Di Eugenio, B., Brown, C., Ohlsson, S., Cosejo, D., and Chen, L.: Supporting Computer Science curriculum: Exploring and learning linked lists with iList. IEEE Transactions on Learning Technologies, Special Issue on Real-World Applications of Intelligent Tutoring Systems, 2(2):107–120, June 2009.

Fossati, D., Di Eugenio, B., Ohllson, S., Brown, C., and Chen, L.: Generating proactive feedback to help students stay on track. In ITS 2010, The 10th International Conference on Intelligent Tutoring Systems, Pittsburgh, PA, June 2010. Short paper.

Fossati, D., Di Eugenio, B., Ohlsson, S., Brown, C., and Chen, L.: Data driven automatic feedback generation in the iList intelligent tutoring system. Technology, Instruction, Cognition, and Learning (TICL), Special Issue on Role of Data in Instructional Processes, 10(1):5–26, 2015.

Fossati, D., Di Eugenio, B., Ohlsson, S., Brown, C., Chen, L., and Cosejo, D.: I learn from you, you learn from me: How to make iList learn from students. In AIED 2009, The 14th International Conference on Artificial Intelligence in Education, Brighton, UK, July 2009.

Gadgil, S. and Nokes, T.: Analogical scaffolding in collaborative learning. In annual meeting of the Cognitive Science Society, Amsterdam, The Netherlands. Citeseer, 2009.

Gal-Ezer, J. and Harel, D.: What (else) should CS educators know? Communications of the ACM, 41(9):77–84, 1998.

Gentner, D.: Analogy. A companion to cognitive science, pages 107–113, 1998.

Gentner, D. and Colhoun, J.: Analogical processes in human thinking and learning. In Towards a theory of thinking, pages 35–48. Springer, 2010.

Gentner, D., Loewenstein, J., and Thompson, L.: Learning and transfer: A general role for analogical encoding. Journal of Educational Psychology, 95(2):393, 2003.

Gong, Y., Beck, J. E., Heffernan, N. T., and Forbes-Summers, E.: The fine-grained impact of gaming (?) on learning. In Intelligent Tutoring Systems, pages 194–203. Springer, 2010.

Graesser, A. C., Chipman, P., Haynes, B. C., and Olney, A.: AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. Education, IEEE Transactions on, 48(4):612–618, 2005.

Graesser, A. C., Conley, M. W., and Olney, A.: Intelligent tutoring systems. 2012.

Graham, G. and Bechtel, W.: A companion to cognitive science. 1998.

Green, N., AlZoubi, O., Alizadeh, M., Di Eugenio, B., Fossati, D., and Harsley, R.: A Scalable Intelligent Tutoring System Framework for Computer Science Education. In 7th International Conference on Computer Supported Education, Lisbon, Portugal, May 2015.

Green, N., Buy, U., and Bartholomew, R.: Assessing Performance of Software Defined Radios on Multicore Hardware. Munich, Germany, June 2013.

Green, N., Di Eugenio, B., Harsley, R., Fossati, D., and AlZoubi, O.: Behavior and Learning of Students Using Worked-Out Examples in a Tutoring System. In International Conference on Intelligent Tutoring Systems, pages 389–395. Springer, 2016.

Green, N., Di Eugenio, B., Harsley, R., Fossati, D., AlZoubi, O., and Alizadeh, M.: Student Behavior with Worked-out Examples in a Computer Science Intelligent Tutoring System. In International Conference on Educational Technologies, Florianopolis, Santa Catarina, Brazil, November 2015.

Guyon, I. and Elisseeff, A.: An introduction to variable and feature selection. Journal of machine learning research, 3(Mar):1157–1182, 2003.

Hake, R. R.: Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. American journal of Physics, 66(1):64–74, 1998.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H.: The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1):10–18, 2009.

Harley, J. M., Trevors, G. J., Azevedo, R., and others: Clustering and profiling students according to their interactions with an intelligent tutoring system fostering self-regulated learning. JEDM-Journal of Educational Data Mining, 5(1):104–146, 2013.

Hilbert, T. S., Schworm, S., and Renkl, A.: Learning from worked-out examples: The transition from instructional explanations to self-explanation prompts. *Instructional design for effective and enjoyable computer-supported learning*, pages 184–192, 2004.

Hofstadter, D. R.: Analogy as the core of cognition. *The analogical mind: Perspectives from cognitive science*, pages 499–538, 2001.

Hooshyar, D., Ahmad, R. B., Yousefi, M., Yusop, F. D., and Horng, S.-J.: A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers. *Journal of Computer Assisted Learning*, 31(4):345–361, 2015.

Hsin, W.-J.: Teaching recursion using recursion graphs. *Journal of Computing Sciences in Colleges*, 23(4):217–222, 2008.

Janning, R., Schatten, C., and Schmidt-Thieme, L.: Perceived task-difficulty recognition from log-file information for the use in adaptive intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, pages 1–22, 2016.

Jeffries, S. and Everatt, J.: Working memory: its role in dyslexia and other specific learning difficulties. *Dyslexia*, 10(3):196–214, 2004.

Kalyuga, S., Ayres, P., Chandler, P., and Sweller, J.: The expertise reversal effect. *Educational psychologist*, 38(1):23–31, 2003.

Kameenui, E. J. and Carnine, D. W.: *Effective teaching strategies that accommodate diverse learners.*. ERIC, 1998.

Khuwaja, R. A., Evens, M. W., Rovick, A. A., and Michael, J. A.: Knowledge representation for an intelligent tutoring system based on a multilevel causal model. In *Intelligent Tutoring Systems*, pages 217–224. Springer, 1992.

Lewis, C.: *Attrition in Introductory Computer Science at the University of California*. Berkeley, 2010.

Liu, Z., Mostafavi, B., and Barnes, T.: Combining Worked Examples and Problem Solving in a Data-Driven Logic Tutor. In *International Conference on Intelligent Tutoring Systems*, pages 347–353. Springer, 2016.

Marx, J. D. and Cummings, K.: Normalized change. *American Journal of Physics*, 75(1):87–91, 2007.

McLaren, B. M., Gog, T. v., Ganoe, C., Yaron, D., and Karabinos, M.: Worked Examples are More Efficient for Learning than High-Assistance Instructional Software. In Artificial Intelligence in Education, pages 710–713. June 2015.

McLaren, B. M., Lim, S.-J., and Koedinger, K. R.: When and how often should worked examples be given to students? New results and a summary of the current state of research. In Proceedings of the 30th annual conference of the cognitive science society, pages 2176–2181, 2008.

McLaren, B. M., Lim, S.-J., and Koedinger, K. R.: When is assistance helpful to learning? Results in combining worked examples and intelligent tutoring. In Intelligent Tutoring Systems, pages 677–680. Springer, 2008.

McLaren, B. M., van Gog, T., Ganoe, C., Karabinos, M., and Yaron, D.: The efficiency of worked examples compared to erroneous examples, tutored problem solving, and problem solving in computer-based learning environments. Computers in Human Behavior, 55:87–99, 2016.

Miller, G.: The Magical Number Seven, Plus or Minus Two. Psychological Review, 1956.

Miller, G. A., Galanter, E., and Pribram, K. H.: Plans and the structure of behavior. Holt, Rinehart and Winston. Inc., New York, 1960.

Mills, B., Evens, M., and Freedman, R.: Implementing directed lines of reasoning in an intelligent tutoring system using the atlas planning environment. In Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on, volume 1, pages 729–733. IEEE, 2004.

Mitrovic, A. and Suraweera, P.: Teaching Database Design with Constraint-Based Tutors. International Journal of Artificial Intelligence in Education, 26(1):448–456, 2016.

Morrison, B. B., Margulieux, L. E., and Guzdial, M.: Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. In Proceedings of the Eleventh Annual International Conference on International Computing Education Research, ICER ’15, pages 21–29, New York, NY, USA, 2015. ACM.

Mostow, J. and others: Evaluating tutors that listen: An overview of Project LISTEN. 2001.

Najar, A. S. and Mitrovic, A.: Do novices and advanced students benefit differently from worked examples and ITS? In Int. Conf. Computers in Education, pages 20–29, 2013.

Najar, A. S., Mitrovic, A., and McLaren, B. M.: Adaptive Support versus Alternating Worked Examples and Tutored Problems: Which Leads to Better Learning? In User Modeling, Adaptation, and Personalization, pages 171–182. Springer, 2014.

Najar, A. S., Mitrovic, A., and McLaren, B. M.: Examples and tutored problems: adaptive support using assistance scores. In Proceedings of the 24th International Conference on Artificial Intelligence, pages 4317–4323. AAAI Press, 2015.

Najar, A. S., Mitrovic, A., and McLaren, B. M.: Learning with intelligent tutors and worked examples: selecting learning activities adaptively leads to better learning outcomes than a fixed curriculum. User Modeling and User-Adapted Interaction, 26(5):459–491, 2016.

Nakabayashi, K., Maruyama, M., Koike, Y., Kato, Y., Touhei, H., and Fukuhara, Y.: Architecture of an Intelligent Tutoring System on the WWW. In Proc. of, pages 39–46, 1997.

Nwana, H. S.: Intelligent tutoring systems: an overview. Artificial Intelligence Review, 4(4):251–277, December 1990.

Nye, B. D.: Intelligent tutoring systems by and for the developing world: a review of trends and approaches for educational technology in a global context. International Journal of Artificial Intelligence in Education, 25(2):177–203, 2014.

Ohlsson, S., Eugenio, B. D., Chow, B., Fossati, D., Lu, X., and Kershaw, T. C.: Beyond the code-and-count analysis of tutoring dialogues. In AIED07, 13th International Conference on Artificial Intelligence in Education, Marina Del Rey, CA, July 2007.

Ososky, S.: Generalized Intelligent Framework for Tutoring (GIFT) Cloud/Virtual Open Campus quick start guide. Technical report, ARL-CR-0796). Orlando, FL: US Army Research Laboratory, 2016.

Paas, F., Renkl, A., and Sweller, J.: Cognitive load theory and instructional design: Recent developments. Educational psychologist, 38(1):1–4, 2003.

Porter, L., Guzdial, M., McDowell, C., and Simon, B.: Success in introductory programming: What works? Communications of the ACM, 56(8):34–36, 2013.

Renkl, A.: Worked-out examples: Instructional explanations support learning by self-explanations. Learning and instruction, 12(5):529–556, 2002.

Renkl, A.: The worked-out-example principle in multimedia learning. The Cambridge handbook of multimedia learning, pages 229–245, 2005.

Rish, I.: An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence, volume 3, pages 41–46. IBM New York, 2001.

Ritter, S., Anderson, J. R., Koedinger, K. R., and Corbett, A.: Cognitive Tutor: Applied research in mathematics education. Psychonomic bulletin & review, 14(2):249–255, 2007.

Rizvi, M. and Humphries, T.: A Scratch-based CS0 course for at-risk computer science majors. In Frontiers in Education Conference (FIE), 2012, pages 1–5. IEEE, 2012.

Romero, C. and Ventura, S.: Educational data mining: A survey from 1995 to 2005. Expert systems with applications, 33(1):135–146, 2007.

Salden, R. J., Aleven, V. A., Renkl, A., and Schwonke, R.: Worked examples and tutored problem solving: Redundant or synergistic forms of support? Topics in Cognitive Science, 1(1):203–213, 2009.

Schwonke, R., Wittwer, J., Aleven, V., Salden, R., Krieg, C., and Renkl, A.: Can tutored problem solving benefit from faded worked-out examples. In Proceedings of EuroCogSci, volume 7, pages 59–64, 2007.

Shute, V. J. and Psotka, J.: Intelligent Tutoring Systems: Past, Present, and Future. Technical report, DTIC Document, 1994.

Sleeman, D. and Brown, J. S.: Intelligent tutoring systems. 1982.

Smith, J. E.: The effect of the Carnegie Algebra Tutor on student achievement and attitude in introductory high school algebra. Doctoral dissertation, Citeseer, 2001.

Sweller, J.: Cognitive load during problem solving: Effects on learning. Cognitive science, 12(2):257–285, 1988.

Sweller, J.: Cognitive load theory, learning difficulty, and instructional design. Learning and instruction, 4(4):295–312, 1994.

Sweller, J.: The worked example effect and human cognition. Learning and Instruction, 16(2):165–169, 2006.

Sweller, J.: Cognitive load theory. The psychology of learning and motivation: Cognition in education, 55:37–76, 2011.

Sweller, J. and Cooper, G. A.: The use of worked examples as a substitute for problem solving in learning algebra. Cognition and Instruction, 2(1):59–89, 1985.

Sykes, E. R. and Franek, F.: An intelligent tutoring system prototype for learning to program java TM. In IEEE International Conference on Advanced Learning Technologies, 2003.

Topley, K.: JavaFX Developer's Guide. Pearson Education, 2010.

Van Gerven, P. W. M., Paas, F., Van Merrinboer, J. J. G., and Schmidt, H. G.: Cognitive load theory and aging: Effects of worked examples on training efficiency. Learning and Instruction, 12(1):87–105, 2002.

Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M.: The Andes physics tutoring system: Lessons learned. International Journal of Artificial Intelligence in Education, 15(3):147–204, 2005.

Wiggins, J. B., Boyer, K. E., Baikadi, A., Ezen-Can, A., Grafsgaard, J. F., Ha, E. Y., Lester, J. C., Mitchell, C. M., and Wiebe, E. N.: JavaTutor: an intelligent tutoring system that adapts to cognitive and affective states during computer programming. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education, pages 599–599. ACM, 2015.

Wiggins, J. B., Grafsgaard, J. F., Boyer, K. E., Wiebe, E. N., and Lester, J. C.: Do You Think You Can? The Influence of Student Self-Efficacy on the Effectiveness of Tutorial Dialogue for Computer Science. International Journal of Artificial Intelligence in Education, pages 1–24, 2016.

Wolfe, C. R., Reyna, V. F., Widmer, C. L., Cedillos-Whynott, E. M., Brust-Renck, P. G., Weil, A. M., and Hu, X.: Understanding genetic breast cancer risk: Processing loci of the BRCA Gist intelligent tutoring system. Learning and Individual Differences, 49:178–189, 2016.

Wood, D., Bruner, J. S., Ross, G., and others: The role of tutoring in problem solving. Journal of child psychology and psychiatry, 17(2):89–100, 1976.

Woolf, B. P.: Intelligent multimedia tutoring systems. Communications of the ACM, 39(4):30–31, 1996.

Woolf, B. P., Arroyo, I., Cooper, D., Burleson, W., and Muldner, K.: Affective tutors: Automatic detection of and response to student emotion. In Advances in Intelligent Tutoring Systems, pages 207–227. Springer, 2010.

Zimmerman, B. J.: Self-regulated learning and academic achievement: An overview. Educational psychologist, 25(1):3–17, 1990.

# VITA

## NICK GREEN

### Education

- 1st Class Honours - BSc Computer Science, Queen Mary, University of London, United Kingdom, 2001

### Work Experience

- VP of Technology, Elite Force Technologies LLC, 2016-Present
- Research Assistant, University of Illinois at Chicago, 2011-2016
- CTO, Sky-Farmer LLC, 2014-2016
- Team Lead, Sony Computer Entertainment Europe, 2005-2011
- Systems Engineer, PI Vision, 2001-2004

### Conference Publications

- N. Green, B. Di Eugenio, R. Harsley, D. Fossati, O. Alzoubi. Behavior and Learning of Students using Worked-out Examples in a Tutoring System. 13th International Conference on Intelligent Tutoring Systems. Zagreb, Croatia, June 2016
- R. Harsley, B. Di Eugenio, N. Green, D. Fossati, S. Acharya. Integrating Support for Collaboration in a Computer Science Intelligent Tutoring System. 13th International Conference on Intelligent Tutoring Systems. Zagreb, Croatia, June 2016
- R. Harsley, N. Green, B. Di Eugenio, S. Acharya, D. Fossati, O. AlZoubi. Collab-ChiQat: A Collaborative Remaking of a Computer Science Intelligent Tutoring System. CSCW '16: Computer Supported Cooperative Work and Social Computing Companion Proceedings. San Francisco, CA, USA, 2016
- R. Harsley, N. Green, M. Alizadeh, S. Acharya, D. Fossati, B. Di Eugenio, O. AlZoubi. Incorporating Analogies and Worked Out Examples as Pedagogical Strategies in a Computer Science Tutoring System. Proceedings of the 47th ACM Technical Symposium on Computer Science Education (SIGCSE). Memphis, TN, USA, 2016
- N. Green, D. Fossati, B. Di Eugenio, R. Harsley, O. AlZoubi, M. Alizadeh. Student Behavior with Worked-out Examples in a Computer Science Intelligent Tutoring System. 3rd International Conference on Educational Technologies. Florianpolis, Santa Catarina, Brazil, 2015

- B. Di Eugenio, N. Green, O. AlZoubi, M. Alizadeh, R. Harsley, D. Fossati. Worked-out Examples in a Computer Science Intelligent Tutoring System. The 16th Annual Conference on Information Technology Education. Chicago, IL, 2015
- O. AlZoubi, D. Fossati, B. Di Eugenio, N. Green, M. Alizadeh, R. Harsley. A Hybrid Model for Teaching Recursion. The 16th Annual Conference on Information Technology Education. Chicago, IL, 2015
- R. Harsley, N. Green, M. Alizadeh, A. Tandon, A. Prabhu. Sick Kitty Toward Promoting Deductive Reasoning through an Embodied Medical Diagnosis Game. In Proceedings of the Games Learning Society (GLS) Conference. Madison, WI: ETC Press, 2015
- N. Green, O. AlZoubi, M. Alizadeh, B. Di Eugenio, D. Fossati, R. Harsley, A Scalable Intelligent Tutoring System Framework for Computer Science Education, 7th International Conference on Computer Supported Education (CSEDU15), May 2015
- M. Alizadeh, B. Di Eugenio, R. Harsley, N. Green, D. Fossati, O. AlZoubi, A Study of Analogy in Computer Science Tutorial Dialogues, 7th International Conference on Computer Supported Education (CSEDU15), May 2015
- O. Alzoubi, D. Fossati, B. Di Eugenio, and N. Green. ChiQat-Tutor: An Integrated Environment for Learning Recursion. Proc. of the Second Workshop on AI-supported Education for Computer Science (AIEDCS) (at ITS 2014). Honolulu, HI, June 2014.
- Omar Alzoubi, Davide Fossati, Barbara Di Eugenio, Nick Green, Lin Chen. Predicting Students Performance and Problem Solving Behavior from iList Log Data. ICCE 2013, The 21st International Conference on Computers in Education, Bali Indonesia, November 2013
- Barbara Di Eugenio, Nick Green, Rajen Subba. Detecting Life Events in Feeds from Twitter. ICSC 2013, Seventh IEEE International Conference on Semantic Computing, Irvine CA, September 16-18, 2013
- Barbara Di Eugenio, Lin Chen, Nick Green, Davide Fossati and Omar Alzoubi. Worked Out Examples in Computer Science Tutoring. AIED 2013, the 16th International Conference on Artificial Intelligence in Education, Memphis TN, July 2013 (Short paper)
- Nick Green, Redge Bartholomew, Ugo Buy. Assessing Performance of Software Defined Radios on Multicore Hardware. SDR-WInnComm-Europe 2013, Munich Germany, June 2013

**Publication Reviews (Meta Reviewer)**

- Journal of Educational Data Mining, 2015
- Evolving and Adaptive Intelligent Systems, 2015
- Evolving and Adaptive Intelligent Systems, 2015
- Educational Data Mining, 2015

- Artificial Intelligence in Education, 2015
- Intelligent Tutoring Systems, 2015
- Artificial Intelligence in Education, 2013

### Teaching Experience

- CS440, Software Engineering I, University of Illinois at Chicago

### Research Experience

- An Analysis of Multicore Hardware in a Real-time Operating System. Ugo Buy. University of Illinois at Chicago. 2015.
- An Investigation into Automatic Parallelisation of C Code. Ugo Buy, Redge Bartholomew. University of Illinois at Chicago. Sponsored by Rockwell Collins. 2013-2015.
- ChiQat-Tutor: Research into a Computer Science Intelligent Tutoring System. Davide Fossati, Barbara Di Eugenio. Carnegie Mellon University in Qatar and University of Illinois at Chicago. Sponsored by the Qatar National Research Fund. 2012-2015.
- The Performance Effects of Multicore Hardware on a Software Defined Radio. Ugo Buy, Redge Bartholomew. University of Illinois at Chicago. Sponsored by Rockwell Collins. 2012-2013
- Life Event Detection in Social Media Feeds. Barbara Di Eugenio, Rajen Subba. University of Illinois at Chicago. Sponsored by Yahoo Research. 2011-2013.

### Honours and Awards

- 50 for the Future Award (Illinois Technology Foundation, 2015)
- UIC Student Travel Presenters Award (UIC, 2013)
- Long Service Award (SCEE R&D, 2011)
- The Rabin Ezra Memorial Award for outstanding service to game development community (SCEE R&D, 2009)