The University of Nottingham

School of Computer Science and Information Technology

Formative Computer Based Assessment in Diagram Based Domains

by Brett Bligh, BSc(Hons).

Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy, September 2006

To my parents…

Abstract

This research argues that the formative assessment of student coursework in free-form, diagram-based domains can be automated using CBA techniques in a way which is both feasible and useful. Formative assessment is that form of assessment in which the objective is to assist the process of learning undertaken by the student. The primary deliverable associated with formative assessment is feedback. CBA courseware provides facilities to implement the full lifecycle of an exercise through an integrated, online system. This research demonstrates that CBA offers unique opportunities for student learning through formative assessment, including allowing students to correct their solutions over a larger number of submissions than it would be feasible to allow within the context of traditional assessment forms.

The approach to research involves two main phases. The first phase involves designing and implementing an assessment course using the CourseMarker / DATsys CBA system. This system, in common with may other examples of CBA courseware, was intended primarily to conduct summative assessment. The benefits and limitations of the system are identified. The second phase identifies three extensions to the architecture which encapsulate the difference in requirements between summative assessment and formative assessment, presents a design for the extensions, documents their implementation as extensions to the CourseMarker / DATsys architecture and evaluates their contribution.

The three extensions are novel extensions for free-form CBA which allow the assessment of the aesthetic layout of student diagrams, the marking of student solutions where multiple model solutions are acceptable and the prioritisation and truncation of feedback prior to its presentation to the student.

Evaluation results indicate that the student learning process can be assisted through formative assessment which is automated using CBA courseware. The students learn through an iterative process in which feedback upon a submitted student coursework solution is used by the student to improve their solution, after which they may re-submit and receive further feedback.

Acknowledgements

To attempt to acknowledge every person who has contributed to my intellectual development, research and life during the four long years of my PhD is a fool's errand. I will not even make the attempt. Instead, I will single out a few notable people and apologise to the rest. You know who you all are.

First of all, I would like to thank Colin Higgins, the leader of the Learning Technology Research group and my supervisor throughout the PhD process. His encouragement and advice were essential to the completion of the research.

Athanasios Tsintsifas provided the starting point for this research through his design of the DATsys framework. He also provided a kick-start to the research in the form of an imposing breeze-block of essential reference materials.

Pavlos Symeonidis provided crucial practical advice and encouragement in his inimitable way and did his best to keep me on the straight and narrow.

The other members of the LTR group provided food for thought and a welcoming environment, especially Marjahan Begum, Swe Myo Htwe (Shannon), Geoff Gray and Nasirah Omar, who were a constant presence in the office and source of banter throughout.

I have relaxed and socialised with a large number of people in Nottingham over the past four years. However, Said Macharkah deserves a special mention for boosting my morale at particularly important moments. My long-term friends from back home have had to put up with much neglect during my stay in Nottingham and deserve my gratitude. The gang from "The Welly" have provided many light-hearted moments during a potentially dull writing up process.

Very special thanks are due to my fiancée, Marioara Urda. With the PhD complete, my priorities lie with her.

Last but absolutely not least, I wish to thank those members of my family — parents, grandparents and all — who have supported me in so many ways throughout my long years of studentship. It would not have been possible without you.

Thank you all, Brett Bligh

Table of Contents

List of Figures

List of Tables

List of Equations

Chapter 1

Introduction

Introduction

Higher education institutions are confronted with the challenge of providing academic courses to a higher number of students without the benefit of a proportionate increase in teaching staff. Student-to-staff ratios (SSRs) are less favourable to academic staff than in the past [Dfes05, Ml97] and SSR increases of 150% since the 1970s have been reported [AUT05]. The delivery of course materials, assessment of student work, detection of plagiarism and administration of course data are but a few of the academic tasks affected by the situation.

Formative assessment is that form of assessment in which the primary aim is to assist the process of learning [Kp01]. Formative assessment should occur throughout the learning process and have the primary aim of providing useful feedback to students [JMM+04]; it stands opposed to summative assessment, in which the primary aim is to provide an indicator of progress at the end of a particular learning process. Formative assessment has considerable pedagogic advantages over summative assessment: it encourages active student learning, can assess a wider range of learning outcomes, can help in the avoidance of mark aggregation and discourages plagiarism [Kp01]. However, formative assessment is more resource-intensive than summative assessment due to its frequency and the detail of the feedback to be provided to the student. Furthermore, summative assessment may be prioritised institutionally due to the need to indicate student achievement externally at the end of an academic course. Therefore, as SSRs become less favourable, the amount of formative assessment from which students can benefit has tended to be reduced.

Computer Based Assessment refers to the delivery of materials for teaching and assessment, the input of solutions by the students, an automated assessment process and the delivery of feedback, all achieved through an integrated, coherent online system. The process of coursework delivery, development of solutions by students, an automated process of assessment and the delivery of student feedback all occur online at the computer terminal [CE98a, SM97]. A prime motivator in the development of CBA technology was to reduce marking time in response to changing SSRs and this is a key benefit of the technology. The reasons for the development of CBA systems are therefore analogous to the reasons for the decline in formative assessment usage.

The CourseMarker CBA system [FHH+01, FHS+01] provides functionality for the authoring, running, marking, and administering of CBA exercises. It is the successor to the widely-used Ceilidh system [FHT+99] but has better performance, scalability, extensibility and maintainability. CourseMarker can accommodate diagram-based CBA through an integrated system known as DATsys [Ta02].

CourseMarker and DATsys represent a powerful mechanism for conducting diagram-based CBA in a summative context. However, the system is not suitable for purely formative assessment. Both the mechanism for the marking of diagrams and the feedback facilities are, while powerful, insufficiently flexible to provide formative assessment courses. Furthermore, there is no mechanism for the marking of diagram layout, which is essential to formative, diagram-based CBA.

The work in this dissertation presents research, design, implementation and evaluation of techniques that facilitate the construction of formative, diagram-based CBA exercises which are unique in the literature.

This chapter presents the motivation for the work and the scope of the thesis and highlights the key contributions novel to this work. The chapter ends with a chapter-by-chapter synopsis of the thesis.

1.1 Background

1.1.1 Motivation

A key objective of this research was to "close the gap" between the pedagogical practices of formative assessment and the field of CBA. Strategies in the literature which attempt to bolster the position of formative assessment within contemporary higher education range from a managed reduction to mechanisation; however, this mechanisation rarely extends beyond ideas such as paper tick-sheets and pre-written feedback statement banks [Rc01]. CBA systems, on the other hand, are often derived from earlier ad-hoc marking scripts informally developed to aid individual lecturers (usually in Computer Science departments); such a description fits Ceilidh precisely. Even when a system is formally designed, as with CourseMarker, the effectiveness of

material delivery and feedback is often measured in terms of student questionnaires rather than by reference to formal teaching principles.

The motivation for this research was therefore to prove that the delivery of a CBA course whose primary aim was for formative assessment could be feasibly achieved, could adhere to principles of good formative assessment and would be useful in practice. Diagram-based domains were used as a vehicle for the research due to the prevalence of diagram-based coursework in many academic disciplines and the free-response nature of the student submissions.

No earlier work describes the formative CBA of diagram-based domains. This is because only recently, with the completion of DATsys, has an extensible, scalable and maintainable framework for diagram-based CBA exercises been available.

1.1.2 Scope

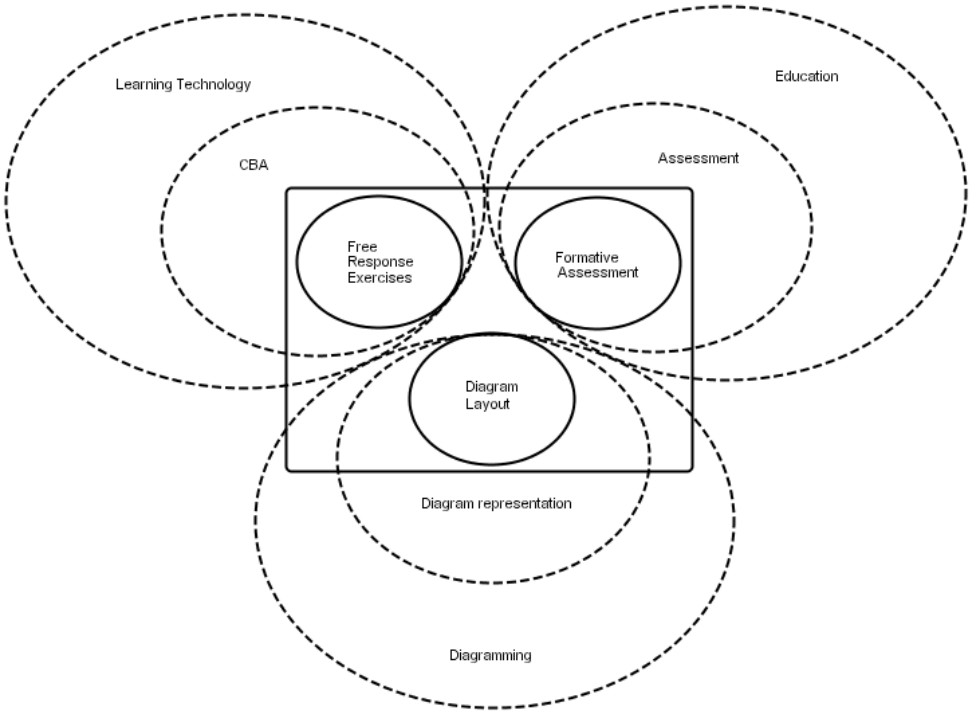

Figure 1.1: The thesis scope at a high level

The formative CBA of exercises in diagram-based domains requires theory and techniques from the disciplines of learning technology, education and diagramming. Figure 1.1 illustrates the relationships between these disciplines in the context of this work.

From a Learning Technology perspective, this research investigates the area of free-response CBA exercises. CBA involves the delivery of course materials, the input of student solutions, the marking and the returning of feedback to the student automatically within an integrated online system. Free-response CBA allows students to construct a solution within an online environment rather than simply selecting one or more options from among distracters.

From the field of education research, this work focuses on formative assessment. Formative assessment involves the provision of feedback to the student in order to enhance learning. Good feedback should motivate learning and opportunities for the student to redeem a poor solution should be provided.

From the field of diagramming, this work investigates diagram layout. The function of a diagram is to convey meaning to the observer. For this to occur successfully a diagram must have an aesthetically acceptable layout as well as the correct diagram elements, connected in the correct manner.

1.2 Brief Overview

1.2.1 General Objectives

This research aims to investigate, propose, design, implement and evaluate techniques which allow formative assessment of diagram-based coursework exercises to be conducted through a CBA system with a practical amount of effort required by those responsible for setting and administering the course. These techniques are illustrated through deliverables which demonstrate the practical benefit of the work.

Two central questions formed the inspiration for this work:

- To what extent can CBA techniques be used to reduce the resources involved in setting a formatively assessed coursework in a diagram-based domain, marking student submissions and returning feedback, while still adhering to good formative assessment principles?

- To what extent would current, successful CBA practices need to be changed to conform to formal formative assessment guidelines?

DATsys provides a simple interactive interface for the authoring of new diagram domains. Marking, however, is achieved through the creation of marking tools. A more detailed set of questions arises from the need for generality across diagram domains:

- To what extent will it be possible for the educator to provide formative feedback in many diagram-based domains by configuring the system and writing feedback comments rather than through impractically complex programming?

- To what extent can standardisation of CBA processes occur without the assessment failing to meet the standards of formative assessment guidelines?

The next set of questions arises from the specific need to mark aesthetic diagram layout:

- To what extent can an automated system for the marking of diagram aesthetics generate useful results within a multitude of diagram domains?

- To what extent will domain-specific layout rules be required and can these be specified by the educator with a practical level of effort?

- What trade-offs are required in a system to mark diagram aesthetics to ensure generality across domains whilst at the same time allowing specialisation when necessary?

Finally, it is necessary to ask questions regarding the performance of a CBA system in conducting formative assessment within diagram-based domains:

- Can formative assessment be rendered less resource-intensive through the use of CBA technology? Conversely, can CBA technology be used to deliver good formative assessment?

- Can a formative assessment process automated using CBA technology enhance student learning?

1.2.2 Problems and Specific Objectives

The general objectives resulted in four major problem areas being examined.

The initial problem area concerns the concrete identification of the problems arising from applying CBA techniques to a formative assessment course. An initial period of research, documented in chapter 4, involved the running of a live formative assessment course for entity-relationship diagrams within a Database Systems module. This ensured that solutions proposed by the work were relevant to formative assessment practice and suggested the three subsequent problem areas for further research.

Further research concentrates on three main problem areas arising out of the practical experience gained. The first subsequent problem area is concerned with the aesthetic appearance of diagrams. Work in the field of graph layout is extensive [BET+94, Sk02] and work on aesthetics has been conducted in the context of user interface design [SP04, NTB00]. However, diagram-based CBA research to date [Ta02, HL98] has not attempted to assess diagram appearance. The objective within this problem area is to design and implement a flexible framework for the assessment of diagram appearance which takes into account general aesthetic principles and the layout rules of specific diagram domains. The approach must be general since multiple diagram domains need to be assessed.

The second subsequent problem area is concerned with the marking process. Diagram-based coursework may have one or more mutually exclusive solutions. CBA software, including CourseMarker, sometimes penalises solutions which deviate from the model solution [Fj01] and has no mechanism for accommodating multiple model solutions to a problem. Good formative assessment should be able to assist students attempting to solve the coursework in different ways. The objective within this problem area is to design and implement techniques to allow variation within model solutions to a problem in order that any one of several mutually exclusive solution elements may be considered a correct solution.

The third subsequent problem area is concerned with the provision of good formative feedback to the student. Feedback is the primary deliverable associated with formative assessment [Kp01] and detailed advice is available on the principles

of good formative assessment feedback [JMM+04]. CBA software can return detailed feedback to the student [HST02], however this feedback is often indiscriminate since it was developed within the context of summative assessment where the emphasis lies in providing a detailed breakdown of the mark obtained. In formative assessment it is more important that feedback is motivational, prioritised and limited in scope to focus student attention on the most serious weaknesses. The objective within this problem area is to design and implement techniques to provide targeted, motivational feedback within a formative context where the student can incrementally improve their solution using multiple submissions.

1.2.3 Approach

Initially, a live experiment was implemented using CourseMarker / DATsys. Coursework involving the construction of entity-relationship diagrams was assessed as part of an undergraduate module in Database Systems. A new marking tool for assessing entity-relationship diagrams within CourseMarker was developed. Problems arising from this live experiment were used to determine which aspects of CBA practice would need to be augmented to conduct formative assessment successfully, a general objective of the work as discussed in section 1.2.1. The results failed to achieve good formative assessment practice and the system did not meet CBA criteria in several key aspects, but the experiment was crucial in identifying the shortcomings of current practices. The subsequent design stage of the thesis could then confidently be targeted on real problems arising from CBA of formative assessment. This work is documented further in Chapter 4.

Subsequent work in the marking of diagram layout, handling of mutually exclusive solution cases and the delivery of truncated, prioritised feedback aimed for a high level of generality. The number of diagram-based domains used in coursework across multiple academic disciplines is large. The approach taken is to determine those factors common across domains and to encapsulate them systemically. The differences between domains are then specified on a per-domain basis through parameterisation and extensions.

To facilitate the marking of diagram layout, a distinction is drawn between aesthetic measures, which denote the commonality across domains, and structural measures, which represent the differences between domains. Layout marking of a specific

domain is achieved through configuration of the aesthetic measures to indicate the relative importance of the factors, together with specification of the structural measures on a per-domain basis, if required. In the handling of mutually exclusive solution cases, a high level of generality is achieved through authoring an expressive notation for the specification of solution cases and their relationships. To generate prioritised feedback, a system of prioritisation of marking factors is developed based upon the relative weight of the factor and the deficiency of the student solution within the factor. Categorisation of factors helps the feedback to be balanced.

1.2.4 Contributions

The primary contribution of this research is in the area of CBA. The combination of the handling of mutually exclusive solution cases and provision of truncated, prioritised feedback is new to CBA and aids the construction of formatively assessed courses in free-response domains using CBA software. CBA in diagram-based domains is also enhanced through the marking of the layout of student solutions. Coursework has been constructed in the domains of entity-relationship diagrams and object oriented design diagrams. The initial work also contributes to the understanding of the problems associated with the use of CBA software in unfamiliar (and unanticipated) contexts.

A second contribution is in the area of diagramming. A flexible and powerful platform for the generic assessment of diagram layout has been provided.

1.3 Synopsis of the Thesis

Chapter 1 outlines the background of the research by showing its motivation and its scope. It gives a brief overview in terms of the general objectives and approach of the work and explains the contributions made by the work. Theory and techniques from the fields of Learning Technology, Education and Diagramming are combined. The research aims to present techniques to achieve the formative CBA of student coursework in diagram-based domains. To achieve this, initial research is presented to demonstrate the problem areas: the marking of aesthetic diagram layout, the accommodation of mutually exclusive solution cases and the delivery of truncated,

prioritised feedback to the student. The work presents novel solutions within each of these specific problem areas.

Chapter 2 introduces the key concepts from the areas of CBA, formative assessment and diagramming. The focus of the work in CBA is in free-response exercises within a formative assessment context. The focus of the work in formative assessment is in providing feedback to the student to assist the process of further learning. The focus within a diagramming context centres on the perception of diagrams from the fields of graph layout and aesthetics. This chapter presents the background and main problems within each area, in order that chapter 3 can concentrate on the most relevant approaches found in the literature.

Chapter 3 presents a critical analysis of the work upon which this research is based, together with other relevant work from the literature. Existing work in free response CBA and diagram editing is documented and other approaches to cope with the resource-intensiveness of formative assessment are examined.

Chapter 4 presents the initial practical research conducted. Coursework involving the construction of entity-relationship diagrams was assessed using CourseMarker / DATsys as part of an undergraduate module in Database Systems. A new marking tool for assessing entity-relationship diagrams within CourseMarker was developed. The experiment is described, key results are presented and conclusions are drawn which feed into the subsequent design and implementation chapters.

Chapter 5 examines the provision of formative CBA within diagram-based domains and outlines the problems which must be overcome in light of the conclusions drawn by the preliminary work in chapter 4. The three identified problem areas are concerned with the assessment of aesthetic diagram layout, the handling of mutually exclusive sections of solutions and the provision of concise, motivational feedback to the student in line with formative assessment principles. The problem of balance between simplicity of configuration, so that the creation of formative assessment coursework by the educator is rendered practical, with expressiveness, so that many diagrammatic domains can be assessed, is examined.

Chapter 6 documents the design decisions made in creating subsystems to augment CourseMarker and shows how these decisions satisfy the objectives identified in

chapter 5. A generic framework for marking diagram layout is designed that can be customised to mark individual diagrammatic domains. An expressive notation for the specification of solution cases and their relationships is developed to facilitate the marking of coursework where the solution has one or more mutually exclusive elements. To generate prioritised feedback, a system of prioritisation of marking factors is developed based upon the relative weight of the factor and the deficiency of the student solution within the factor. Categorisation of factors helps the feedback to be balanced.

Chapter 7 reports on a prototype system which was developed as an extension to CourseMarker. It documents the three main subsystems developed in response to the three identified problems and shows how they interact with the existing CourseMarker CBA system and the integrated DATsys environment.

Chapter 8 evaluates the prototype system and documents the success of the approach taken by the research. Formative assessment can be conducted in multiple diagram-based domains using CBA techniques. The evaluation considers the success of the design from the point of view of educators and documents results with students in example domains.

Chapter 9 reviews the thesis' key points and shows how the evaluation of the system in chapter 8 relates to the general objectives for research stated in chapter 1. The contributions of this research to the fields of CBA, formative assessment and diagramming are discussed. Areas for future work are considered. The thesis shows that formative assessment within diagram-based domains can be feasibly conducted using CBA techniques and is useful in the practical context of higher education.

Chapter 2

CBA, Formative Assessment and Diagramming

Introduction

This chapter provides the research background in the fields of Computer Based Assessment, formative assessment and diagramming. A section is presented for each field.

Section 2.1 introduces Computer Based Assessment (CBA). CBA is defined specifically in terms of its relationships with other areas of learning technology. A brief historical overview of automatic assessment is provided and the motivation for the development of the technology is explained. The advantages and limitations of CBA techniques are considered. Methods to minimise the limitations associated with CBA usage are documented.

Section 2.2 provides an overview of formative assessment. Formative assessment is defined and its differences to other forms of assessment emphasised. The merits of formative assessment are considered and the decline in formative assessment usage within education institutions explained. Strategies to overcome those drawbacks of formative assessment which are responsible for this decline are considered. The provision of feedback as the primary aim of formative assessment is explained and criteria for good formative feedback are presented.

Section 2.3 examines the concept of diagrams. The role of diagrams in education is examined and the presence of diagrams in a wide number of academic disciplines is demonstrated. An overview of the academic study of diagrams is provided and the concept of aesthetics in diagramming is introduced.

2.1 Computer Based Assessment

2.1.1 Definition

As institutions seek to maintain teaching and assessment standards with decreasing unit-resource, attempts are being made to automate some or all of those processes necessary for conducting teaching, learning and assessment — such as authoring of course and assessment material and mark schemes, distribution of material and questions to the learner, development and submission of student solutions, course

administration and marking [CE98a]. The collection of processes necessary for conducting a piece of assessment is known as the *lifecycle* of the assessment. Computer Based Assessment (CBA) constitutes a section of learning technologies distinguished from others by the number and types of processes that are automated within the lifecycle. The relationships between CBA and Computer Assisted Assessment (CAA), Computer Assisted Learning (CAL) and Computer Based Learning (CBL) can be represented as shown in Figure 2.1 [HB06].

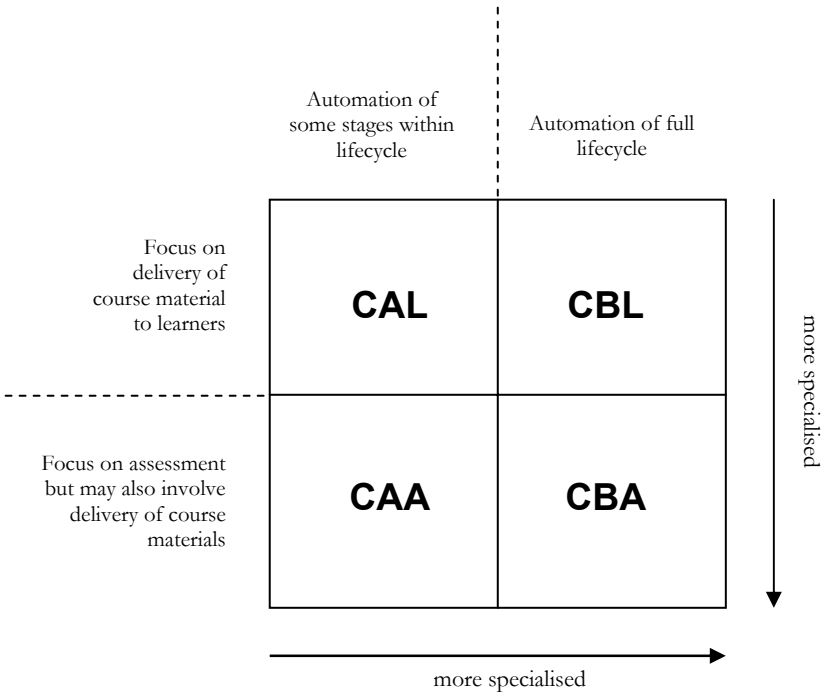

Figure 2.1: Relationships between CAL, CBL, CAA and CBA

Computer Assisted Learning (CAL) is a generalised term which refers to the use of technology to ease the learning process in virtually any way. Only tasks of teaching and learning may be automated and even these in a superficial way with little co-ordination between the automation of separate tasks. Thus, the delivery of lecture materials using software packages such as *Microsoft PowerPoint* or allowing students to print out lecture notes from a centrally available resource would constitute a basic form of CAL.

Computer Based Learning (CBL) is defined as that subset of CAL in which the learning materials must be presented to the student online via a computer terminal in a coherent system; the implication is that the student is primarily responsible for

navigating through the course materials available online and structuring their learning at their own pace. Only tasks of teaching and learning need to be automated, but the intention is to create a coherent system which can be utilised by the student with little need for input from a teacher.

An example of CBL is *MacCycle* [Bj93], used at St. Andrews University to teach second-year medical undergraduates about the menstrual cycle as a replacement for lectures on the topic. The students are able to work through the material, which includes text, images, video and interactive sections showing dynamic changes in hormone levels, at their own pace and are then asked to write and electronically submit an essay based upon what they have learned from the system. However, the assessment, an essay, is then later printed out by the tutor and hand-marked. CBL can therefore be seen as a specialisation of CAL in which an entire process of learning is conducted online through a computer terminal.

Computer Assisted Assessment (CAA) refers to the use of technology to deliver coursework to the students, mark student responses and conduct analysis of submitted coursework. In CAA the automation of certain tasks of assessment, teaching and learning are likely to be present, but some stage of the process (often the development of solutions by candidates) is still accomplished using mechanisms outside the system.

A common practice which constitutes CAA involves using Optical Mark Reader (OMR) technology to read student responses to an assessment from paper into an assessment system which compares the responses against a set of model answers. The teaching and assessment material would likely have been created using computerised means as well. In this way, CAA can be seen as be seen as that specialisation of CAL which involves automated marking and analysis of student submissions as well as the delivery of materials.

Computer Based Assessment (CBA) therefore refers to the delivery of materials for teaching and assessment, the input of solutions by the students, an automated assessment process and the delivery of feedback, all achieved through an integrated, coherent online system. It can therefore be seen as that specialisation of CAA in which the entire process (including the development of solutions by candidates) occurs online at a computer terminal and also as that specialisation of CBL in which

assessment must occur as part of the system, as well as the delivery of teaching and learning materials. CBA is the most specialised form of learning technology to be considered because it provides for the highest level of automation within a coherent system; this, in turn, means that CBA has more potential in terms of time saving than the other forms of learning technology. This model and definition are consistent with those prevailing in the literature [CE98a, Ta02, SM97].

2.1.2 Development of Automated Assessment

Early automated assessment systems began to appear towards the beginning of the 1960s. The use of computers to automate simple, repetitive processes was already appreciated and educators within fields such as computer science, physics and mathematics were eager to take advantage of the time-saving potential offered by automating the assessment process. The first automated assessment systems were characterised by the use of simple marking mechanisms to assess simple question types. Hollingsworth [Hj59, Hj60], describes a system used as early as 1959 to assess a student machine language course which, despite problems of unreliability and lack of security, was seen to be clearly justified on "economic grounds".

The system described by Forsythe and Wirth [FW65] is similar to that of Hollingsworth in that a simple matching mechanism was used to assess carefully simplified exercises, in this case in the programming language Balgol, a variation on Algol-58. Students submitted their work on punched cards and the marking was done as a batch after the deadline for the exercise had passed. Subsequent systems tended to improve the level of automation achieved and the number of exercise domains which could be assessed.

Hext and Winings [HW69] describe a system which could assess three domains (two variants of Algol and an assembly language) and whose batch processing was entirely automated. Later papers by Taylor and Deever [TD76], Rottman and Hudson [RH83] and Myers [Mr86] expanded automated assessment usage into domains outside Computer Science, in physics, mathematics and chemistry respectively.

Despite a historical progression charting an increased level of automation in the assessment process and a gradual widening of the domains covered by automated assessment, all of the systems described above involve simple assessment

mechanisms being used to assess exercises which were carefully constructed by the educator to be "assessable". Rather than to improve the pedagogic quality of assessment, automated assessment was seen as a mechanism with the potential to increase the speed with which assessment could be carried out and thus to allow the assessment of increasing numbers of students to be rendered feasible. As student numbers increase further, this factor is likely to continue to be a motivator in the development of automated assessment systems.

2.1.3 Motivation in Automated Assessment

Higher education institutions are confronted with the challenge of providing academic courses to a higher number of students without the benefit of a proportionate increase in teaching staff. Student-to-staff ratios (SSRs) are less favourable to academic staff than in the past [Dfes05, Ml97], and SSR increases of 150% since the 1970s have been reported [AUT05]. The delivery of course materials, assessment of student work, detection of plagiarism and administration of course data are but few of the academic tasks affected by the situation. Section 2.1.2 showed that, historically, automated assessment research has been motivated by a desire to assess more students with increased speed; this motivation would appear set to increase given current conditions.

In the systems described in section 2.1.2 the automation often led to a change in the presentation of the assessment process itself, which was constrained by the limitations of the technology available. Students were obliged to submit their solutions in a simplified form which could be assessed by the system; this introduced notations for representing solutions which were far removed from the coursework problems themselves. Only in more recent times have the pedagogic implications of automated assessment come to be acknowledged. An evaluation of prominent contemporary automated assessment systems is provided in Chapter 3. However, it is clear that pedagogic principles cannot be ignored when the development of the automated assessment field is considered.

Automated assessment is an inter-disciplinary topic and contributors to the field often maintain the perspective of their original discipline when making contributions. Section 2.1.1 defined CBA technology in terms of the number and types of processes automated. Often, the extent to which the assessment process is

automated is determined by educator preconception rather than the practical limits of the technology available. Canup and Shackelford [CS98], for example, argue that final marking must always be performed by human graders using automation as a simple aid. Mason and Woit [MW98, MW99] propose approaches which involve an online system for the presentation of examination materials and collection of student submissions but only a limited role for automatic marking. Joy and Griffiths [JG04] describe a system which facilitates online student submission and allows both students and graders to run automatic tests on submissions, but which provides neither learning materials nor an integrated, fully automated marking process by design. None of these systems, therefore, constitute CBA technology as defined in section 2.1.1. This work will utilise a CBA approach based upon the full automation of the lifecycle of an exercise in order to maximise the resource-saving potential offered by the research. Furthermore, an integrated CBA system which allows student solutions to be developed in a naturalistic, intuitive way within an interactive environment will be used to minimise abstract representations which are removed from the student learning process.

2.1.4 Benefits of Computer Based Assessment

The benefits of CBA technology fit into two broad categories: the *practical* and the *pedagogical*.

The practical reasons were the motivation for the development of CBA in the first place and were the focus of section 2.1.2. Charman and Elmes [CE98a] acknowledge that CBA develops out of the desire to automate an increasingly large workload of assessment, within the context of providing higher education to a larger proportion of the population without proportionately higher resources. They consider that such a scenario often leads to the following assessment strategies being adopted:

- Reducing the assessment loading for students;
- Evaluation of the function of each piece of assessment;
- Diversification of the assessment portfolio.

Adoption of CBA techniques is often as a result of a decision to diversify the assessment portfolio since CBA can be used to save time in the assessment process,

rather than abandoning sections of assessment altogether. Rust [Rc01] suggests six methods for confronting the same issues: front ending, doing the assessment in class, use of self and peer assessment, group assessment techniques, assessment mechanisation, and strategic reduction strategies. CBA would clearly constitute a mechanisation strategy, although it is worth noting that Rust's own suggested mechanisation strategies are confined to the use of paper tick-sheets and statement banks to aid in the traditional feedback process.

Charman and Elmes [CE98a] emphasise, however, that the resource-saving potential of CBA technology is often manifest in the long term. CBA systems are non-trivial to develop and maintain and commercial systems may be costly to purchase. A cost-benefit analysis of CBA technology used in a limited context over a short timescale can easily be negative; six strategies are, therefore, suggested to maximise the practical benefits of CBA technology.

Firstly, the advantages of CBA technology will be maximised over a large timescale. CBA technology is often difficult to develop or learn to use. Writing questions and feedback can often be more time-consuming, early in the process, than the traditional assessment methods being replaced would have been in their entirety. However, once the technology is in place the resource-savings can be utilised repeatedly since the process is automated and the total time costs compared with tutor based assessment will often show considerable benefit.

Secondly, additional resources for development may be available. Many institutions have funds available for technological development. Linking CBA deployment with research may also allow further resources to be allocated.

Thirdly, introducing CBA technology may allow module delivery to be restructured, allowing teaching assistants to assist in the administration of the CBA system.

Fourthly, wider departmental, or even institutional, benefits should be considered. If a strategy for using the CBA technology across several modules or even departments can be developed then the benefits of the technology are increased and the resource burden shared.

Fifthly, existing assessments in the subject area should be considered. Many projects in educational technology have been developed in recent years. Making direct use of the experiences and even technological infrastructure developed by those who have already experimented with automated assessment in the subject area can save considerable resources. To this end, Charman and Elmes [CE98b] provide a useful handbook which documents several existing academic CBA systems.

Sixthly, existing question banks should be utilised if they exist. Some institutions have existing question banks whose questions could be automatically assessed, and the establishment of national and international question banks is now underway [SP03]. Re-usable digital resources which can be used to support learning are available in Learning Object Repositories (LORs), which are often web-based. A prominent example is MERLOT [Mf04]. Neven and Duval [ND02] provide an overview of pertinent issues associated with LOR use.

The *pedagogical* benefits of CBA technology are often overlooked. Experience [CE98a, BBF+93] has shown that CBA software:

- Increases assistance to weaker students because problems in learning can be immediately traced and teaching strategies adapted accordingly;
- Provides immediate feedback which ensures that the student can internalise the submission and feedback as one entity while both are fresh in the mind;
- Increases student consciousness about the assessment process since students are more willing to contest automatically generated results and, therefore, become interested in determining what the assessor is looking for in a model solution;
- Increases student confidence by allowing easy early exercises and by demonstrating to students that they are performing well;
- Encourages students to effectively manage their own workload since students can increase their mark through multiple submissions if they begin to submit before the deadline;

Criterion	*Meaning*	*Application to CBA*
Valid	The assessment should measure what you want to measure and not depend on other qualities	Will measure specified coursework aspects assuming good initial assessment design
Reliable	The assessment should be consistent between assessors and for the same assessor on different occasions	The same assessment process will run for each submission; consistency is absolute
Fair	Assessment should provide equal opportunity to succeed; students should *perceive* the assessment as fair	Design-dependent; CBA has no inherent advantages
Equitable	Assessment should not discriminate between students other than by ability. Particular talents (e.g. exam technique) should not be disproportionately favoured.	The same assessment process will run for each submission; discrimination is non-existent
Formative	See section 2.2 for a full definition	CBA provides a good opportunity to run assessment frequently throughout the learning process, and to provide multiple submissions with full feedback each time
Timely	Assessment should occur throughout the learning programme	CBA provides a good opportunity to run assessment frequently throughout the learning process
Incremental	Assessment should be a gradual process allowing achievement to be 'built up'	Design-dependent; CBA has no inherent advantages
Redeemable	Initial failure should not be absolute and students should have a second chance	CBA is suited to allowing multiple submissions should the designer wish this
Demanding	Assessments should be pitched at the right level of achievement and not be easy	Design-dependent; CBA has no inherent advantages
Efficient	Assessment should make efficient use of available resources; over-assessment should be avoided	Considerable time and other resource savings to be made; originally a motivator for CBA's inception

Table 2.1: CBA provides concrete pedagogical benefits

- Provides an opportunity to put the information learned in a course immediately into effect in the next piece of work.

Brown et al [BRS96] argue that good assessment should be valid, reliable, fair, equitable, formative, timely, incremental, redeemable, demanding and efficient.

Table 2.1 provides an explanation of each of these terms and then considers whether CBA meets the criterion in question. The definitions are consistent with those in [CE98a]. It can be seen that in 7 of the 10 criteria CBA is likely to present a distinct pedagogic advantage over traditional assessment, while in the remaining 3 criteria CBA has no negative effect. Hence CBA can be said to have concrete pedagogic benefits.

2.1.5 Limitations of Computer Based Assessment

Like the benefits of CBA which were the focus of section 2.1.4, the limitations of CBA can be separated into two large categories: *practical* and *pedagogical*. Awareness of these issues, together with careful design and prior planning, can help to minimise the problems encountered during the assessment process.

A survey of teaching staff with technical backgrounds by Inoue [Iy01] concluded that, given institutional support for the use of technology in education, the following six practical factors have the greatest influence on the success or failure of educational technology in general:

- **Teachers' knowledge and skills in technology.** Training programs for educators are useful to ensure their awareness of available technology, their ability to choose appropriate technology and their ability to use the technology correctly.

- **Availability of hardware and software.** Research into educational software must be encouraged and appropriate hardware developed.

- **Commitment by involved parties.** Educators must have the determination to persevere in solving problems associated with implementing educational technology, rather than returning to traditional assessment forms.

- **Availability of time.** Educators must be aware that implementing educational technology can be an initially time-consuming process and allow time for this in their schedule.

- **Availability of technical support.** Educational technology can be difficult to implement. Developers must provide appropriate technical support to educators to overcome technical obstacles.

- **Cost of hardware.** The hardware on which the education software runs must not be prohibitively expensive for an institution to purchase and install.

Inoue emphasises that his findings are consistent with those of other studies, and furthermore identifies the latter three limitations as oft-stated inhibitors for computer use generally. Charman and Elmes [CE98a] focus on three central problems in this area: the availability of equipment for writing CBA material, the availability of equipment for the delivery of CBA material to students and the existence of an infrastructure to implement CBA delivery on a suitably large scale.

In order to define the *pedagogic* limitations of CBA techniques it is necessary to define precisely the *type* of learning which is to be assessed. Bloom's Taxonomy of learning objectives [BEF+56] is the learning model most cited by the automated assessment community and classifies learning into three categories: cognitive, affective and psychomotor. Most assessment is an attempt to evaluate cognitive learning. Bloom's Taxonomy further divides the cognitive learning domain into six levels of increasing cognitive complexity as illustrated in table 2.2. Each cognitive level is assumed to encompass those below it; for example, Comprehension cannot occur without Knowledge. Bloom's Taxonomy is both simple and easy to apply.

As a result of research conducted during the 1990s a revised version of Bloom's Taxonomy has been proposed by a group of researchers led by Anderson and Krathwohl [AK01] in which the six cognitive levels are renamed Remembering, Understanding, Applying, Analysing, Evaluating and Creating and form the Cognitive Process Dimension of a two-dimensional taxonomy. The second dimension is the Knowledge Dimension, comprising Factual Knowledge, Conceptual Knowledge, Procedural Knowledge and Meta-Cognitive Knowledge. In the revised Bloom's taxonomy, curricular standards are aligned with both the Cognitive Process

Dimension and the Knowledge Dimension according to their position in a two-dimensional table.

Cognitive Level	*Meaning*
1. Knowledge	To recall information approximately as it was learned
2. Comprehension	To interpret information based upon prior learning
3. Application	To select data and principles to solve a problem with minimum outside assistance
4. Analysis	To distinguish and relate the assumptions, structure or hypotheses of a statement
5. Synthesis	To originate and integrate ideas into a proposal that is new to the student
6. Evaluation	To critique on the basis of explicit standards

Table 2.2: Bloom's levels of cognitive learning

While many educationalists agree with Bloom's general approach, some cognitive psychologists, who doubt either the ordering or the distinction between the cognitive levels, debate Bloom's explicitly tiered architecture. For example, the alternative RECAP taxonomy [Ib84, Ib95] advocates the view that the three highest cognitive levels in Bloom's taxonomy, Analysis, Synthesis and Evaluation, cannot be robustly distinguished and hence presents a general category called "problem-solving" which represents the three combined. Assessors who simply wish to ensure that surface-learning is avoided have successfully used both taxonomies [DK01, BEH+05] and it is clear that the implications for assessment design are less than for research into cognitive psychology.

The radically different SOLO taxonomy [BC82] is notable because it deviates from Bloom's approach. SOLO is based upon the evaluation of student responses to assessment rather than the design of the assessment itself. Stephens and Percik [SP03], however, argue that SOLO sacrifices validity for increased reliability. Since CBA has already been shown to be reliable in section 2.1.4, automated assessment is overwhelmingly devised using either Bloom's or Bloom-like taxonomies.

The pedagogic limitations of CBA techniques are related to the type of student response which can be assessed by the system. *Fixed-response assessment* is a term used to refer to assessment modes in which students must choose their answer from a pre-designated selection of alternatives including distracters. Fixed-response assessment modes, such as multiple-choice questions, are often criticised for simply assessing the Knowledge of a student (the lowest level in Bloom's taxonomy) and therefore encouraging surface-learning strategies [JA00]. Although this problem occurs when traditional assessment methods are used, the problem is exacerbated when CBA assessment is considered because of the *prevalence* of fixed-response assessment in CBA systems. Johnstone and Ambusaidi [JA00] note that such methods of assessment have the following disadvantages:

- Students can guess the correct solution from the alternatives offered;
- Awarding negative marks to incorrect answers to discourage guessing discourages even "educated" guesses and, furthermore, can be shown to be statistically futile [BM00];
- Students can usually eliminate many distracters from common sense, leading to a situation sometimes called "*multiple true-false*";
- It is often unclear to the educator *why* a student chose an answer — students can get the correct answer for the wrong reason;
- *Negative discrimination* can occur in which more knowledgeable students are disproportionately tempted by incorrect distracters;
- Students can be disproportionately affected by precise wording: the success rate of a question can realistically be changed by 20% by the simple use of an unfamiliar word.

In their overview of the automated assessment field, Bull and Danson [BD04] note that "*CAA is most commonly associated with multiple-choice questions*" while Culwin [Cf98] notes that that the majority of CBA systems to date consider only fixed responses. Fixed-response assessment has the practical advantage to the CBA designer that the assessment algorithm can be kept simple, since it is necessary only to check whether the correct response has been provided. It is for this same reason

that other mechanised learning systems, such as Optical Mark Recognition, are based upon fixed-response questions. However, this has led to CBA suffering from many of the perceived disadvantages of exclusively fixed-response assessment in the minds of educators, since the two are seen to be synonymous; in a web survey conducted by Carter et al [CDE+03], only 36% of academics agreed or strongly agreed with the statement "*It is possible to test high-order learning using CAA*".

It is, however, possible to take action to minimise the pedagogic limitations of CBA. Charman and Elmes [CE98a] suggest the careful construction of assertion-reason multiple-choice questions as a technique for assessing deeper student understanding of material, while Duke-Williams and King [DK01] present research into question design techniques for use with multiple-choice and graphical hotspot questions to ensure assessment of higher learning outcomes. Furthermore, it is possible to conduct *free-response CBA* where the pedagogic limitations are reduced, though this is less common in the literature due to the complexity of the marking algorithms required. Section 2.1.6 presents a taxonomy for CBA in which the types of fixed-response and free-response questions are considered. Their advantages and limitations are then examined.

Bull and Danson [BD04] argue that the prevalence of CBA which tests only basic knowledge is a result of misconception, lack of pedagogic understanding and poor question design rather than inherent limitations of the technology. They counter the problem in "*cultural acceptance*" by drawing attention to the existence of automated assessment systems which "*draw on an extensive range of sophisticated question types by using computers to create questions which would not be possible using the medium of paper.*" They continue that: "*CAA offers the opportunity to creatively extend the range and type of assessment methods used* [to] *support and enhance student learning in ways which are not possible with paper-based assessments.*"

2.1.6 A Taxonomy for Computer Based Assessment

Section 2.1.5 introduced the concepts of *fixed-response* and *free-response* automated assessment. The key difference between the two is the process by which the learner constructs their solution to the problem.

In fixed-response systems the learner chooses a solution from a fixed number of clearly defined alternatives. One or more of the alternatives is the correct solution; the other alternatives are incorrect and serve as distracters. The automated assessment system is required to record whether the solution submitted was correct or a distracter; no unanticipated solutions are permitted. Fixed-response assessment may also be referred to as *objective assessment*.

In free-response systems the student is presented with an environment within which a solution can be constructed in a freeform way. Free-response assessment requires a more complex marking algorithm since the student solution cannot be precisely anticipated. Free-response automated assessment constitutes only a small minority of the platforms in existence because the complexity of the development process acts as a deterrent. Culwin [Cf98] notes that the development of free-response automated assessment is, in comparison with its fixed-response counterpart, "*much harder or even impossible.*"

2.1.6.1 Fixed-response Automated Assessment

Fixed-response automated assessment includes the assessment of multiple-choice questions (MCQs), short response exercises or graphical hotspot exercises.

Multiple Choice Questions (MCQs) present the user with a statement or stem, followed by a series of choices from which a selection must be made [JA00]. Traditionally, one choice was the correct answer, often referred to as "the key". The remaining choices, "the distracters", were incorrect. A variation in which the student must identify more than one key is often referred to as a "multiple response question". Frosini et al [FLM98] provide a list of MCQ variants: simple true/false, item order, multiple choice, multiple response, combination, gap filling and best answer. Modern automated assessment systems which provide a platform for MCQs do not limit either the stem or the choices to be text; images, sounds or video footage are all examples of credible MCQ components.

Many commercial systems offer platforms for conducting large scale automated assessment of MCQs. *QuestionMark* [BSP+03], which claims to be the world-leader, allows for the creation of multiple-choice questions within an interactive authoring environment and includes facilities for the presentation of questions involving a

wide variety of colours, fonts and pictures, as well as the option to automatically open other applications (for example, a spreadsheet package or calculator) during the assessment. The associated program Perception [BFK+04] allows for assessment to be distributed over the Web. Commercial competitors with comparable facilities include *EQL Interactive Assessor* [Mp95], *LXR*TEST* [GW01] and *QuizIt* [TBF97]. A comparison of the features of these tools is provided by Baklavas et al [BER+99].

Short response exercises require the student to provide an answer in the form of a word, short phrase or number. Students typically present their answer in response to a stem question, as for MCQs, or may be asked to complete a sentence or phrase. The assessment system compares the answer provided against one or more correct answers, or "keys". The system may expect an exact response, or a degree of flexibility may be introduced through the use of regular expression-like notations such as Oracles [ZF92] or allowance of rounding errors for numerical answers. Academic systems which support short response exercises include TRIADS [Md99] and Ceilidh [BBF+93].

Graphical Hotspot Exercises require the student to select an area on a graphic, to connect two or more graphics together using a connection line or to arrange graphics on a canvas containing pre-defined positions. The correct response is defined in advance, usually as a sequence of "target areas" within which an answer is deemed to be correct. Later versions of the commercial *QuestionMark* software [BSP+03] allow this type of exercise while, previously, many domain-specific systems were created using multimedia authoring packages such as Asymetrix Toolbook [Asy94] and Macromedia Authorware [Mac95].

2.1.6.2 Free-response Automated Assessment

Free-response automated assessment includes the assessment of programming assignments, essay exercises and diagrammatic exercises.

Programming assignments occur frequently within Computer Science education. Efforts to automate them result from the historically increasing popularity of technology courses, in terms of student numbers, and the background of many educational technology pioneers within the Computer Science field itself.

The *Ceilidh* system [BBF+93] was an important pioneer in demonstrating the feasibility and usefulness of automatically assessing programming assignments. Ceilidh was also one of the first systems to cater for the full lifecycle of a CBA exercise. Ceilidh checked for the presence of designated tokens in a student's program text and simulated output using an extended regular expression notation called "oracles" [ZF92]. Ceilidh was rendered able to mark assignments in many domains, including several programming languages, through its multi-layered architecture and its introduction of several key CBA concepts such as marking tools, multiple user views and a logical course structure. Ceilidh was an important influence on subsequent CBA development and is considered in more detail in section 3.3.1.

The direct successor to the Ceilidh system is *CourseMarker* [FHH+01, FHS+01]. CourseMarker takes its inspiration from the most successful aspects of Ceilidh but benefits from an improved, object-oriented design and a platform-independent implementation in Java. The result was a system which built upon the success of Ceilidh, but which had increased usability, maintainability and extensibility. The CourseMarker system was used as a platform for the work described in this thesis and is considered in more detail in section 3.3.2.

Another system influenced by the example of Ceilidh is ASSYST [Jd00, JU97]. Like Ceilidh, ASSYST caters for the full lifecycle of a CBA exercise and is aimed to be a complete "grading support system" rather than a simple assessment tool. ASSYST is used to analyse programming assignments in the C language according to correctness, style, complexity and run-time efficiency; like Ceilidh, it is possible to allocate proportional weightings to the tests. Unlike Ceilidh, ASSYST takes a "hybrid" approach between CBA and manual marking. Jackson and User claim that this enables the system to benefit from the marking consistency and speed associated with CBA while still maintaining fine control over student results. This approach does, however, negate some of the benefits of CBA, such as the ability to allow great numbers of submissions and the immediate return of full feedback, since teacher intervention in the assessment process is required for each submission.

Another hybrid approach is taken by the BOSS system [JG04, JL98], which facilitates online student submission and allows both students and teachers to run automatic

tests on submissions, but which provides neither learning materials nor an integrated, fully automated marking process by design. The student can run automated tests as an aid to constructing and evaluating their solution and is then able to submit their solution online. The educators can then run automated tests on the solution, invoke plagiarism detection mechanisms, mark the solution and return feedback online. BOSS does not operate as a fully automated, coherent CBA system as a design decision. Joy and Griffiths [JG04] acknowledge that an integrated CBA approach can act as a formative process; however, they state that the aim of BOSS is to concentrate "*on the process, and measuring the correctness of students' code*" and argue that CourseMarker and Ceilidh "prescribe" a style of programming through their frequent, automated checking.

The Kassandra system described by von Matt [Mu94] conducts automatic testing of programs written in the Matlab, Maple and Oberon languages. These languages are mathematically based and correctness is determined by matching output data with defined test data. Little infrastructure is provided for automatic testing and test software is developed by the exercise developer. Kassandra does, however, support more than one type of user: the "student" and the "assistant".

The RoboProf system described by Daly [Dc99] also uses output checking. RoboProf concentrates on assessing the syntax and structure of programming languages rather than program correctness. RoboProf is influenced by the architecture of Ceilidh and is used for formative assessment purposes. RoboProf is considered in more detail in section 3.1.2.1 on CBA systems in a formative assessment context.

The TRAKLA system described by Korhonen and Malmi [KM00] is primarily a Computer Based Learning system which presents a visual environment to teach students the concepts of algorithms and data structures through the use of diagrams and animations. However, a formative assessment component used to test student understanding has been introduced based upon the Ceilidh model. The latest version, TRAKLA2, is considered in more detail in section 3.2.1.

The ASAP system described by Douce et al [DLO+05] utilises "test classes" which can be seen as analogous to Ceilidh's marking tools. Each test class must provide objective criteria for evaluation, feedback for the test and a single mark to evaluate the submission. Standardisation of test classes is accomplished through a template

superclass which all test classes must extend, as in CourseMarker. ASAP was developed to be integrated into institution-wide e-learning frameworks such as Blackboard or WebCT using VLE standards. This allows ASAP to take advantage of institutional infrastructure and represents an advance on the typical CBA approach of developing standalone systems.

The JEWL system described by English [Ej02] assesses student programs which involve a Graphical User Interface (GUI). English believes that this increases student motivation since programs with GUIs are seen by students as "real programs" rather than toys. The JEWL system is an object-oriented toolkit; the student solution is replaced by a "test harness" which interprets those instructions which the student program executes. Further research into the assessment of student GUIs is also being carried out using CourseMarker as a platform [GH06].

Essay exercises are popular as an assessment tool since they are seen as a proven way to test higher-order cognitive skills such as synthesis and analysis.

Landauer [LD97, LHL98] described an approach based upon Latent Semantic Analysis (LSA). LSA emphasises essay content by analysing word co-occurrences while ignoring the linguistic and structural features of an essay. LSA scores typically correlate as well with human graders as different graders do with each other [CO97]. A variant of LSA was used as the basis of a prototype system for assessing essays using Ceilidh as a platform [FL94].

Page [Pe94] describes the Project Essay Grade (PEG) system, which uses a model with the essay's surface features, including document length, word length and punctuation features, as independent variables and the essay score as the dependent variable. PEG scores have been found to correlate better with human graders than the graders correlate with each other [PPK97].

Rudner and Liang [RL02] describe a statistical approach based upon Bayesian networks which is simple to implement and can be used on short essays. A student response is classified into one of three grades (complete, partially complete or incomplete) according to probabilities which have been associated with features of the essay as likely to be appropriate, partial or inappropriate. Rudner and Liang argue that their approach can incorporate the best features of earlier methodologies.

Diagrammatic exercises are the least common of the free-response CBA exercise types examined here. Hirmanpour [Hi88] describes an automated diagramming tool which allows students to develop Data Flow Diagrams, Entity-Relationship diagrams and Structure charts without using licensed software. The assessment however, is traditional; automated assessment of diagrammatic exercises is regarded as difficult.

Power [Pc99] presents a development environment called Designer which allows the student to interactively design structure diagrams. Designer can analyse the diagrams and present programs and control structures. Designer allows the student to interactively "walk-through" the program represented by the structure diagram they have designed. The commercial object-oriented design package IBM Rational Rose [Qt99] allows code templates to be generated from object-oriented class diagrams in the C++ and Java programming languages, although this functionality is primarily aimed at software developers.

Hoggarth and Lockyer [HL98] developed a system in response to problems in teaching systems analysis and design. Computer Aided Software Engineering (CASE) tools are often used in teaching this subject to allow students to apply the basic theory and concepts, but most CASE tools are intended for commercial use and do not cater for students who require assistance in underlying concepts. Hoggarth and Lockyer describe an interactive CBA learning system which embeds a Computer Assisted Learning (CAL) system within an existing CASE system to assist student understanding. A verification mechanism for student diagrams relies on the student manually matching the meanings of 'tokens' (the names of diagram components) in their solution with the corresponding tokens in the model solution. The verification mechanism then compares the diagrams as two directional "flows" of modes and connections and notes the differences in ordering between the two. Specific feedback is then provided to the student, which can be used to improve the solution in an iterative, formative process. Hoggarth and Lockyer's diagram comparison system and the feedback provided are reviewed in more detail in section 3.2.3.

DATsys [Ta02] is a framework for conducting diagram-based CBA. DATsys caters for the full lifecycle of automated assessment exercises through integration with the CourseMarker CBA system. Diagrammatic domains can be defined by exercise developers without programming, using the diagram editor component *Daidalos*.

Marking tools for diagrams are defined on a per-domain basis as CourseMarker marking tools. Tsintsifas reports on the use of DATsys to assess student coursework in logic design, flowchart and object-oriented design as part of a first-year undergraduate course in Software Tools. Logic design exercises involved the student drawing an analogue circuit diagram, which was assessed by a marking tool which simulated a circuit based upon the student diagram, provided test data and checked output properties. Flowcharts were translated into programs and marked as such using tools developed in CourseMarker for the assessment of programming exercises. Object-oriented design diagrams were marked using a tool which tested the features of the tool, such as the presence of nodes and the connections between them. DATsys was used as a platform for this work and its architecture and infrastructure is detailed further in section 3.3.3.

Thomas [Tp04] reports on the use of a simple drawing tool to allow students to draw diagrams as part of an online examination. The drawing tool used a diagram representation consisting of simple nodes and links and allowed students to arrange the diagram elements and to define the text labels which were associated with them. Most students were able to use the drawing tool even though they were under exam conditions and unfamiliar with the tool itself. Later research [TWS05] investigated the assessment of student entity-relationship diagrams using a system which compared the *features* of a student solution with those of a model in a similar way to DATsys' assessment of object-oriented design diagrams. This system, together with the feedback provided to students, is further reviewed in section 3.2.4.

2.1.7 Assessing Higher Cognitive Levels using CBA

Automated assessment has traditionally been associated with testing only the lower levels of Bloom's cognitive taxonomy due to the most common, fixed-response question types being dismissed as "mere" objective testing. It is still common for automated assessment to be regarded as having an "*inability*" to test higher skills due to its reliance on "*simple techniques such as pattern-matching of input*" [RJE02].

Research into assessing higher levels of Bloom's cognitive taxonomy fits into two broad categories. The first approach is to carefully design objective test questions according to criteria designed to force students to demonstrate higher order cognitive abilities such as analysis, synthesis and evaluation. The second approach is

to utilise more complex automated assessment mechanisms to enable CBA to assess question types which would traditionally be used to assess higher order cognitive abilities.

McKenna and Bull [MB99] present an overview of the techniques often used to design effective objective test questions. The techniques focus primarily upon the construction of multiple choice questions and a series of weak questions with improved counterparts are presented. Techniques discussed include: constructing the question stem as a definite statement, avoiding irrelevant material, constructing the stem to test a student understanding of the domain rather than reading comprehension, concentrating material in the stem and avoiding option duplication. Techniques for extending MCQs to the preferred variants, such as multiple true / false questions, assertion-reason items, multiple response questions, matching test items and text match response problems are then considered.

Duke-Williams and King [DK01] set out an explicit approach to question design using a revised version of Bloom's taxonomy. The authors note the limitations of traditional approaches, such as constructing questions using verbs known to be associated with higher-order learning outcomes, and demonstrate a system of question design which makes use of both MCQs and graphical hotspot questions. Stephens and Percik [SP03] document the procedure of creating questions based upon Bloom's taxonomy through a process of concept mapping.

This research forms part of the second strand: that of automating the assessment of more complex question types through the use of more complex assessment mechanisms. Essays [RL02], programming assignments and diagrammatic questions [Ta02] are suitable for assessing the higher cognitive levels of Bloom's taxonomy and can therefore be used to reduce the pedagogical drawbacks associated with CBA usage.

2.1.8 Summary

Section 2.1 defined Computer Based Assessment in relation to other areas of learning technology in terms of the number and types of processes that are automated. The motivations for the development of CBA technology were considered, and a brief history of CBA development was provided. CBA has practical advantages, such as

the saving of resources (time), and pedagogical advantages in terms of reliability and other assessment criteria. CBA's practical limitations are primarily infrastructural, since introducing CBA into an academic environment is a resource-intensive process. CBA's pedagogical limitations relate to its perceived inability to assess the higher cognitive levels as defined in taxonomies such as Bloom's. Attempts to minimise these pedagogical limitations may include the careful construction of objective questions or the automation of question types traditionally used to assess higher cognitive levels. An overview of the fixed-response and free-response CBA question types was provided. This research forms part of the free-response strand of CBA into the automated assessment of diagrammatic exercises. Section 2.2 will introduce the key concepts associated with formative assessment.

2.2 Formative Assessment

2.2.1 Definition

Formative assessment is typically defined thus: "*Formative assessment involves methods designed to establish what progress a student is making during learning and to enable giving feedback on it*" [Bj93].

Such a definition carries two implications: firstly, that formative assessment must occur during the process of learning and, secondly, that the most important deliverable associated with formative assessment is feedback. These two implications are complementary; the aim of the feedback is to improve the learning of the student whilst that learning is still ongoing. Thus, formative assessment stands opposed to summative assessment, whose central function is to provide an indicator of achievement (e.g. in the form of a grade) at the conclusion of a unit of learning, rather than feedback. Knight [Kp01] goes further in arguing that only formative assessment truly provides feedback and that the results of summative assessment merely constitute "*feedout*" since they may have little impact on the subsequent learning process.

Assessment may sometimes be drawn into the four categories of formative assessment, summative assessment, diagnostic assessment and self-assessment. In this model, summative assessment remains opposed to the other three types in both

form and function. Self and diagnostic assessments constitute further specialisations of formative assessment with specific forms and purposes [Mm02].

The working definition of formative assessment adopted here is as follows:

> "*Formative assessment is assessment conducted throughout the learning process, as an integral part of that process, where the central aim is to provide feedback to enable the enhancement of learning.*"

It is inferred, firstly, that feedback includes that given to both student and educator to enhance the process of learning and, secondly, that the provision of feedback takes priority over the generation of "feedout" such as summary grades or marks.

2.2.2 Benefits of Formative Assessment

There are many pedagogic advantages associated with formative assessment [Kp01]. Formative assessment:

- Encourages openness among students;
- Can be used to assess a great scope of learning outcomes;
- Can help in avoiding mark aggregation;
- Discourages plagiarism.

Rowntree [Rd87] considers the relationship between an education system's learning objectives and its assessment mechanisms thus: "*If we wish to discover the truth about an education system, we must look into its assessment procedures. What student qualities and achievements are actively valued and rewarded by the system. How are its purposes and intentions realised? To what extent are the ideals, aims and objectives professed by the system ever truly perceived, valued and striven for by those who make their way within it? The answers to such questions are to be found in what the system requires the students to do in order to survive and prosper. The spirit and style of student assessment defines the de facto curriculum.*"

Assessment, therefore, defines what students learn. Few students will expend effort in trying to acquire those skills and knowledge which are not rewarded by the system, irrespective of the stated aims and objectives of the course on paper. Brown

[Bg01] notes that, "*assessment shapes learning, so if you want to change learning change the assessment method*." Students increasingly learn how "to play the exam game", engaging in surface learning strategies at the expense of genuinely broad and deep learning.

By its nature, summative assessment invites deceit since the informed student is aware that it is their work that is being assessed, not themselves. The student has an interest in emphasising their knowledge and hiding their ignorance in any work which is summatively assessed in the hope of attaining the greatest possible final grade. Similarly, the student has an interest in focusing only on those sections of a syllabus which are likely to be directly assessed and ignoring all others.

Knight [Kp01] notes that good formative assessment encourages disclosure rather than deceit. The student is much more likely to admit an area of ignorance if the consequence is further assistance, rather than a lower mark. Given Rowntree's assertion above, this implies that formative assessment encourages a more general programme of learning, reduces the impetus for students to "play the exam game" and allows insight by the teacher into the effectiveness of the syllabus and the teaching methods employed.

Similarly, formative assessment discourages plagiarism. Stefani and Carroll [SC01] argue that plagiarism is difficult to define precisely and that much plagiarism may be unintentional and could be solved by better education of students on academic standards. However, plagiarism is still an increasingly high-priority concern within higher education. Since formative assessment encourages disclosure, and students feel it is in their best interests to admit to weakness, then plagiarists harm only themselves [Kp01]. In a formative assessment environment, therefore, plagiarism is less likely.

Section 2.1.4 introduced the concept of reliability in relation to assessment. Modern higher education learning outcomes are complex and demand assessment of what are often referred to as "soft skills". Knight presents extracts from a course handbook in order to argue that many learning outcomes cannot be assessed reliably as part of a summative assessment; Knight argues that formative assessment is the only authentic method of providing feedback on these outcomes. The feedback is as fuzzy as the learning outcomes, but the impact of this disadvantage can be minimised so

long as students are warned in advance and the assessment is formative. Thus, it is often impossible to assess many fuzzy outcomes reliably and affordably other than with formative assessment since it is less constrained by reliability concerns.

Furthermore, Knight argues that an increase in the proportion of formative assessment can be utilised (as part of a strategy which also incorporates a proportionally smaller summative assessment element) as a solution to the noted problem of mark aggregation within increasingly modular higher education courses.

Black and William [BW98] conducted a survey of 681 research publications on formative assessment. They concluded that formative assessment acts to improve the student learning process to an extent that, "*if best practices were achieved in mathematics on a nationwide scale that would raise 'average' countries such as England and the USA into the top five*".

Despite the pedagogical advantages associated with formative assessment, recent times have seen a marked decline in its use on higher education courses. The next section examines this phenomenon and explains why it has occurred.

2.2.3 Drawbacks associated with Formative Assessment

The drawbacks associated with formative assessment can be grouped into two broad categories: the pedagogic and the practical. Many of the pedagogic drawbacks can be ameliorated using known techniques. The practical drawbacks are traditionally seen as more implacable.

Yorke [Ym01] identifies four main academic problems associated with high formative assessment use:

- A student who wished to terminate a module which traversed multiple semesters before the end of the final semester would have difficulty obtaining credit if the assessment conducted up until that point was formative;
- Again, in a multi-semester module, it is difficult to ensure equity between semesters if formative and summative assessment is unevenly distributed across the semesters;

- Students may under-prioritise formative assessment when under pressure to complete summative assessments (and, possibly, to engage in paid employment);

- Problems occur in ensuring that formative assessment is maximally effective — the student must be provided with good feedback and make good use of the feedback to improve their future learning.

The first two issues cannot be decisively addressed within the scope of assessment design. They are systemic and can only be dealt with at the programme or even institutional level.

A common suggestion which attempts to address the third problem is the use of a *two-part assessment strategy* [Kp01]. Here, formative assessment is used initially, with a summative element introduced later as a motivator. So long as the summative component is in some way based upon the formative — for example, an assignment could be marked formatively, to be followed by a question based upon the assignment which is assessed summatively — then the assessment as a whole is viewed with higher priority by the student. Hence the pedagogic advantages of formative assessment can still be retained. Depending upon the timing of the summatively assessed components, this could, additionally, go some way towards addressing the first two problems. Unfortunately, adopting a two-part assessment strategy exacerbates the practical problems associated with conducting formative assessment still further.

The problem of designing useful, effective formative feedback is examined in section 2.2.5.

The practical problems associated with formative assessment are simpler yet more consequential than the pedagogic problems. Effectively, formative assessment is viewed as being costly to undertake, especially in terms of time to mark assessments. The creation of rich and meaningful feedback to the student is more involved than simply assigning a grade or mark. In the past this was tolerated, but as Chapter 1 pointed out, recent years have seen a marked decline in staff-to-student ratios and educators are expected to teach students with ever-decreasing unit-resource. Given this deterioration, many staff simply believe that they do not have the time or other

resources which would be necessary to undertake formative assessment to the levels used in the past.

Furthermore, the two-part assessment strategy proposed as a solution to some of the pedagogic problems associated with formative assessment would seem to indicate that one should implement formative assessment *and then do the summative assessment as well anyway*. Comprehensive formative assessment thus becomes viewed as a pipe-dream.

Several strategies have been proposed to overcome this practical difficulty. The next section will briefly outline these strategies and place this work in context among them.

2.2.4 Managing the Resource Intensiveness of Formative Assessment

Rust [Rc01] presents an overview of the assessment issues associated with teaching large groups. Rust argues that there are six main methods which can be used to maintain the quality of assessment within these difficult conditions:

- "Front-ending" the assessment;
- Conducting assessment in class;
- Conducting self and peer assessments;
- Conducting group assessments;
- "Mechanising" the assessment;
- Strategically reducing the amount of assessment conducted.

"Front-ending" the assessment refers to a strategy concentrating educator and student effort at the beginning of the course. The purpose is to "set up" the students for the work they are going to have to complete. An example is the creation and dissemination of very detailed instructions or checklists, including examples, of what is expected from the course's assessment. Rust argues that this reduces the marking time associated with misinterpretation of work and results in fewer student requests for guidance.

Conducting assessment in class refers to a strategy of performing assessment alongside the presentation of teaching materials, within allocated class time. Examples include setting assignments which can be undertaken and / or marked in class or alternatively giving general feedback in class rather than individual feedback to students after marking is complete.

Self and peer-assessment can be used as a technique for generating feedback for students which would have been too time-consuming for staff to write. Group assessment can also be used as a useful device for solving common student problems. Self, peer and group assessment constitute active areas of research in their own right; a general introduction is provided by Race [Rp01].

In its most general form, the strategy of mechanising the assessment refers to standardising the assessment process in order to save time. Rust, in common with many outside the CBA field, restricts ambition in this area to speeding up feedback using statement banks and feedback tick-sheets. Statement banks are pre-written archives of feedback statements from which the teacher chooses the most appropriate; this saves time because the teacher does not have to consider the wording of the feedback provided to the student. Feedback tick-sheets contain grids of tick-boxes aligned with both scores and feedback statements. Feedback is returned to the student by ticking the appropriate boxes and handing back the sheet itself as a statement of feedback. Rust also promotes the use of objective tests such as MCQs to ease the marking workload. Section 2.1 outlined the approach taken by Computer Based Assessment technology to further mechanise the assessment and feedback process.

A strategic reduction strategy can be split into two distinct approaches: reducing the amount of assessment conducted, or reducing the amount of time spent providing feedback. This is the least preferred option available, although it is often viewed as practical. Rust argues that if assessment reduction is carefully considered then the effect on students need not be hugely detrimental in all cases.

This work falls within boundaries of the fifth strategy for teaching large groups, that of mechanising the assessment. The benefits and drawbacks of mechanising assessment using CBA software have been examined in sections 2.1.4 and 2.1.5 respectively.

Front-ending has been criticised for resulting in student work of high conformity and little originality. Performing the assessment in class involves allocating time previously used for teaching to assessment, hence simply "moving" the problem and restricting teaching opportunities. A strategic reduction strategy in assessment would view formative assessment as more "expendable" than the summative components; formative assessment would, therefore, be disproportionately reduced.

Self, peer and group assessment are assessment forms which offer considerable opportunities for future research. However, the validity and reliability of these forms is as yet unproven. Furthermore, the use of these assessment forms as part of a two-part assessment strategy would require considerable problems to be overcome; these forms may, therefore, be susceptible to low student take-up or effort.

The next section focuses on criteria for designing effective feedback for formative assessment.

2.2.5 Effective Feedback for Formative Assessment

The central aim of formative assessment has been defined previously in section 2.1.1 as "to provide feedback to enable the enhancement of learning". In order to claim that good formative assessment has occurred, therefore, it is necessary to show that good formative feedback has been produced. Nicol and Macfarlane-Dick have proposed seven principles of formative feedback practice as a result of their conceptual model based upon student-centred learning methodologies [JMM+04]. These criteria will be used to judge the effectiveness of formative assessment conducted using CBA techniques and are briefly explained here.

A good feedback framework for formative assessment should:

1. Facilitate the development of self-assessment (reflection) in learning

An assessment programme which is overly educator-led will produce students dependent on others for instruction. A student should be expected to monitor the divergence between their internal perception of the task and the outcomes being produced. Critical self-assessment is a good technique for allowing students to evaluate the weaknesses of their own work and thus to incrementally improve their standards and develop their evaluative skills.

2. Encourage teacher and peer dialogue around learning

Educator feedback serves as a valuable objective yardstick against which students can evaluate their work, but there is considerable evidence that students find this difficult to internalise and respond to productively. Feedback from peers may be useful because another novice experiencing the same learning curve may have experienced similar problems and is likely to be able to communicate compatibly. Furthermore, the educator should be willing to respond to queries relating to feedback. In this way, feedback should be viewed as a continuous dialogue rather than as simple information transmission.

3. Clarify what constitutes good performance

A student is attempting to close the gap between their *own* internal perception of a task and their current outcomes. Thus, the degree of *overlap* between the student's internal perception of the task and the actual goals of the educator in setting the exercise is important and should be maximised. Poor performance by students may be related to a misinterpretation of what is required. Hence feedback should include mechanisms to clarify task requirements if students are to improve their performance in future.

4. Provide opportunities to improve performance

Feedback should change subsequent student behaviour. The impact of feedback is reduced if students receive feedback which points out the errors in the specific solution, only to move on to a different assignment. In such a situation, a student may regard the feedback as irrelevant. Students should therefore make a response to feedback soon after it is delivered; ideally an opportunity should be provided to repeat the task-performance-feedback cycle. A good method for achieving this is by allowing resubmission. Furthermore, feedback should support students in producing a piece of work by offering constructive advice rather than mechanically listing errors.

5. Deliver information focused on student learning

Feedback should focus on the objectives of the task being attempted rather than providing a list of unrelated strengths and weaknesses on a per-solution basis.

Feedback should be delivered in good time and not be overwhelming in quantity, utilising a manageable number of prioritised comments in order to maximize the likelihood of corrective action. Feedback sheets with lengthy criteria lists and marks discourage a view of the exercise as a holistic entirety. Hence the number of criteria about which feedback is given should be controlled. Feedback should be available for the student to consult in the future.

6. Encourage positive motivational beliefs and self-esteem

Student motivation is related to the type of external feedback the student receives. Frequent high stakes assessment involving marks or grades lowers motivation and leads to students concentrating on passing the test rather than learning. A mixture of grades and feedback comments leads to students concentrating on the former at the expense of the latter. Therefore frequent assessment in which only feedback comments are provided is recommended. Feedback should also concentrate on achieving future learning goals rather than current failure in order to lead to a motivational 'incremental view' of learning.

7. Provide information to educators that can be used to help shape the teaching

If feedback to the students is to be of a high standard then the assessment process should include a mechanism for feeding back good information to teachers.

2.2.6 Summary

Section 2.2 introduced the concepts of formative assessment. Formative assessment is assessment conducted throughout the learning process, as an integral part of that process, where the central aim is to provide feedback to enable the enhancement of learning. Formative assessment has concrete pedagogic benefits and has been shown to improve student learning. The pedagogic drawbacks associated with formative assessment are not the main reason for its decline in usage in higher education courses and can be minimised through institutional planning and the adoption of a two-part assessment strategy. Formative assessment has declined in use because it is seen as a resource-intensive assessment mode. Strategies for reducing resource-intensive assessment include front ending, doing the assessment in class, use of self and peer assessment, group assessment techniques, use of a mechanisation strategy, and strategic reduction strategies; this work constitutes an example of the use of a

mechanisation strategy. Criteria were presented by which the effectiveness of formative assessment feedback can be analysed. Section 2.3 introduces the concepts associated with diagrams, examines their role in learning and assessment and provides an overview of criteria used to assess good diagram layout.

2.3 Diagrams in Education

2.3.1 Definition

In 1911 James Maxwell [eb11] defined a diagram as "*a figure drawn in such a manner that the geometrical relations between the parts of the figure illustrate relations between other objects*". This definition of diagrams as an abstract representation is general enough to encompass the many forms that diagrams have assumed throughout history but it is insufficiently concrete to be used as the basis for research.

This section will concentrate on the role of diagrams in education. Although educational diagrams take many forms across multiple educational domains, it is still possible to present a general definition which is specific enough to serve as the starting point for research.

Dodson [Dd99] noted that a diagram is typically comprised of two types of components: *nodes* and *lines*. Many common types of educational diagrams consist of nodes linked by lines, but diagrams can alternatively consist of combinations of lines, nodes overlapping, nodes 'labelled' by other nodes and nodes whose meaning is determined by colour or other distinguishing feature. For a diagram to be comprehensible, its notation must conform to a *convention of meaning* which describes how the elements of the notation are combined to indicate a relationship or function.

2.3.2 History and Scope

Tsintsifas [Ta02] provides an overview of the historical role of diagrams. The word "diagram" is Greek in origin and means literally *to express using lines*. The earliest diagrams were land maps, which demonstrate a low level of abstraction because of the direct relationship between the diagram and the terrain it represents. Later diagrams demonstrate an increased level of abstraction: examples include ancient Greek illustrations of geometric concepts, philosophical representations found in

European texts from the Middle Ages and family trees of genetic lineage. The level of abstraction apparent in diagram representations began to increase markedly in the 17th century with the development of the Cartesian co-ordinate system. Later stages of progress include the development of Calculus and set diagrams.

This work will concentrate on modern diagrams used in education. Diagrams are used in a wide variety of disciplines spanning science, the humanities and art.

In computer science, diagrams are used as an aid to visualisation, for solving computational problems and for design purposes. Von Neumann [GV47] created the "flowchart" notation to aid in the visualisation of algorithms. Statecharts [Hd88], Petri nets [Pc65] and state transition diagrams [BGK+96] assist in the solving of computational problems. Diagrams to assist in the object-oriented design of software projects include UML diagrams [JBR98] and the earlier competitors such as Booch diagrams [Bg93]. Lohse et al [LBW+94] present illustrations of sixty graphical representations, including frequently used notations such as entity-relationship diagrams, data-flow diagrams, Nassi-Schneiderman diagrams and pert charts.

Diagrams are used extensively across engineering disciplines. For example, circuit diagrams for analogue and digital components are used in electrical engineering, while manufacturing blueprints are used in mechanical engineering. Standards databases exist to document the conventions of meaning of these notations [ISO05, ANSI05].

In sports science, articulated body schematics are used to illustrate ideal athletic body positions, examine body stresses, determine ranges of motion and calculate acceleration [MLC03]. A wide variety of diagrams are used across biology to describe biological processes and structures; famously, Watson and Crick [WC53] accompanied their discovery of the DNA double-helix structure with a "*purely diagrammatic*" representation. Interdisciplinary diagram notations with less formal conventions include concept maps [GS95] and mindmaps [Tb93]. More generally, Blackwell and Engelhardt [BE98] have proposed a "taxonomy of taxonomies" for diagrams. Blackwell and Engelhardt present six taxonomic dimensions for diagrams, each of which represents a category of interest in research: representation, message, relation between representation and message, task and process, context and convention and mental representation.

2.3.3 Diagrams in Automated Assessment

Tsintsifas [Ta02] conducted research into diagrams within the context of automated assessment. He noted that approaches to developing diagram editors could be grouped into three broad categories: multi-domain diagram editors, frameworks and diagram editor generators. A multi-domain diagram editor aims to address the editing of a group of related diagram domain notations and is usually specialised to provide editing of a group of related notations in a subject area, for example software engineering. A framework allows developers to create new editors by extending the framework, taking advantage of existing design and implementation; this approach allows great freedom in editor customisation but requires more effort from the developer. A diagram editor generator requires the developer to provide a specification for a diagram notation in a customised grammar. The program then generates a software implementation of a new diagram editor from the specification provided.

Multi-domain diagram editors include Thinglab [Ba79], which allows a description to be followed by the runtime execution of constraints by using the interpreted language Smalltalk. The visual and non-visual attributes of the diagram editors can be related based upon formulas described in Smalltalk. The Templa/Graphica system [Hs90] uses a "design template" to customise a graphical editor, while MetaBuilder [FWW00] allows specifications for a new editor to be provided in the form of a "meta-diagram" which describes the elements, relationships and constraints of the diagram domain. The commercial software package *Microsoft Visio* [En01] also conforms to the multi-domain diagram editor pattern.

Framework approaches include MacApp [App89], ET++ [GMW88], Unidraw [VL89], HotDraw [Tek87] and JHotDraw [BG97]. In each case, to create an editor for a new diagram domain the developer must create specialised classes for each abstraction, building on top of the existing implemented architecture.

Diagram Editor Generators include Minas [Vg95], which relies on the construction of a hypergraph grammar to define a new editor based upon an archive library of graphical components. GenEd [HW96] allows editors for "visual languages" (of which educational diagrams are a subset) to be defined using algebraic specifications. The Penguins system [CM03] allows the realtime creation of editors

for a wide variety of diagram domains using a constraint multiset grammar as the specification for the language. Penguins also allows editors to be created from malformed or incomplete grammars through a system of incremental parsing.

Tsintsifas concluded that existing multi-domain diagram editors, frameworks and diagram editor generators were generally unsuited for use in an assessment context. Multi-domain graphical editors are constrained in scope and lack the features required for automated assessment. Existing frameworks were aimed at programmers and developers, had many extraneous features and were considered overwhelming in an assessment context due to their complex architectures. Similarly, diagram generators were unsuitable because a deep understanding of the mechanics of the generator was required to specify a new domain; non-programming users such as assessment developers could not be expected to attain such specialist domain knowledge simply to set a new exercise domain.

Tsintsifas developed DATsys, an object-oriented framework whose classes make up a reusable design for CBA-oriented diagram editors. Daidalos, Ariadne and Theseus are presented as concrete subclasses for such diagram editors, intended for use by domain developers, exercise developers and students respectively. Representations for diagram domains are specified using Daidalos. Daidalos defines tools for the creation of figures, diagram elements, tools and commands, as well as a selection editor which allows domain libraries of diagram notations to be managed. Developers using Daidalos to author diagram domain notations can define diagram elements in terms of their graphical view, underlying data model and connectivity constraints.

DATsys is fully integrated into the CourseMarker CBA system and makes use of CourseMarker's system of marking tools to provide a generic marking mechanism which will allow any diagram notation to be marked. A drawback of this generality is that the development of marking tools, which are necessary to assess diagram domains, is left to the developer, who must have knowledge of both the domain to be marked and the system of marking tools. DATsys was used as a platform for this research, and a more detailed overview is provided in section 3.3.3.

Thomas et al [TWS05, STW04] concentrated their attention on the "network-like domains" which are common in computer-science education, such as entity-

relationship diagrams and pipelines. The smallest meaningful unit, an "association", is defined as two nodes connected by a line. Student diagrams are assumed to be "imprecise" compared with a model solution, since required features may be missing or incorrectly presented and extraneous features may also be included. Thomas et al concentrate on a tool which conducts a comparison of the associations found in both the model solution and the student solution. The research of Thomas et al is reviewed in more detail in section 3.2.4.

2.3.4 Aesthetics of Educational Diagrams

2.3.4.1 Aesthetic Criteria

Section 2.3.1 noted that most educational diagrams consist of a collection of nodes with lines linking the nodes. A convention of meaning applies to define how the nodes and lines logically interact to relate meaning to the reader. The reader's interpretation of the diagram is also influenced by the aesthetic *layout* of the diagram, such as the physical relationships between the diagram elements. By implementing consistent diagramming practices including clear layouts, reader confusion can be minimised [Dfa04].

Aesthetic principles have been proposed in fields as disparate as fine art and architecture. For the purposes of considering the physical relationships between nodes and lines in educational diagrams, generalised approaches can be found in the fields of graph layout and user interface design. The field of graph layout suggests precise attributes for graphs which can be assessed mathematically, but doubts remain over the merits of the resulting automatic layout algorithms [EG03]. User interface design principles consider the layout of user interface primitives, which can be considered as nodes, on a computer display, but in this context the nodes are not connected by lines. Both of these approaches are examined in more detail below. It is also necessary to consider the domain-dependant layout rules which exist for many common educational diagram types.

When considering aesthetic criteria it is important to bear in mind the point made by Purchase et al [PAC02] that not all criteria are of equal importance. Purchase et al conducted a study into how the aesthetics of UML class and collaboration diagrams, common educational domains, are perceived by readers. "Preference tests", based

upon a user's instinctive preferences between a set of diagrams placed before them, were conducted using sets of technically literate (though not necessarily UML-conversant) volunteers. Volunteers were provided with pairs of diagrams in which one of the pair emphasized a given feature while another did not and were asked to indicate their preferred diagram with reasons. Quantitative data were collated using a points system for volunteer responses while qualitative data, in the form of the stated reasons, were used to search for confounding factors. In this way, the relative importance of six aesthetic measures commonly suggested in the literature, as well as two additional domain-specific layout rules for each of the two types of diagram, were ascertained.

For UML class diagrams the most popular measure, concerned with the minimisation of edge crossings, had a 93% preference level while the least popular measure, concerned with directional indicators on diagram arcs, rated 60%. The most and least popular measures for UML collaboration diagrams were spaced even further apart. Although all measures scored more than 50% preference and hence would appear to be valid, all the measures are not of equal value. Therefore, any educational system which aims to take into account the aesthetic merit of a diagram layout must not only determine the specific criteria to be measured but also the relative importance, or weighting, of the criteria.

2.3.4.2 Criteria from Graph Layout

The field of graph layout classifies graphs into two broad groups: *syntactic graphs*, which are abstract and have no real-world meaning, and *semantic graphs*, which have a real-world meaning and are usually used to convey information within a domain. Petre [Pm95] notes that semantic graphs are subject to "*additional secondary notations*" which tend not to be defined within the formal syntax. Layout features associated with syntactic graphs cannot be transferred *ad hoc* to semantic graphs, including most educational diagrams; to be relevant, a syntactic layout measure must have a real-world application.

The two most commonly cited criteria in graph layout, and two of the easiest to calculate, are the number of "bends" in connection lines and the number of connection lines that cross or overlap other connection lines. Tamassia [Tr87] proposes that an optimal graph has nodes exclusively connected by straight

connection lines and states that curved and segmented connection lines should be minimised. Tamassia et al [TTV00] later presented an algorithm to this end within an automated layout context. Diagramming guidelines support the minimisation of bends in such real-world domains as entity-relationship diagrams [Dfa04].

Reingold and Tilford [RT81] state that trees should avoid the "overlapping" of connection lines, while Stedile [Sa01] acknowledges this as a principle for drawing all graphs. Sugiyama [Sk02] concentrates on minimising the overlapping of both connection lines and nodes. The study by Purchase et al [PAC02] concluded that this was the single most important aesthetic consideration in the presentation of UML diagrams.

Papakostas and Tollis [PT00] outline the concept of graph *orthogonality*. An optimal orthogonal graph has the nodes and connection lines aligned to a regular grid pattern. Nodes should be aligned with grid intersections, while connection lines should lie along the gridlines. Tamassia [Tr87] proposes a similar aesthetic measure, while studies in the real-world domain of user interface design have produced similar measures [NTB00].

Coleman and Stott Parker [CS96] propose a measure which seeks to minimise the physical width of a drawing. Other measures concentrate on the text labels which accompany a graph: text direction should be uniform and the font typeface should be consistent throughout [Pm95, Dfa04].

2.3.4.3 Criteria from User Interface Design

Ngo et al [NTB00, NB01] provide an overview of aesthetic measures from the field of user interface design, presenting fourteen measures which constitute a "*theoretical approach to capture the essence of artists' insights*". The authors acknowledge the criteria as being applicable outside the field of user interface design; their aim is to provide a mathematical model in which the insight of artists is represented by a series of mathematical formulae which assume values between 0 and 1.

The fourteen aesthetic measures proposed by Ngo et al are briefly summarised in Table 2.3. It can be seen that a subset of the aesthetic criteria overlap with the criteria from the graph layout literature outlined in section 2.3.4.2.

Measure	*Description*
Balance	The distribution of "optical weight". Optical weight is calculated from its area, colour and shape. Balance is considered both vertically and horizontally.
Equilibrium	The difference between the centre of mass of the elements and the physical centre of the screen / canvas.
Symmetry	The level of axial duplication. Symmetry is measured horizontally (about the horizontal axis), vertically (about the vertical axis) and radially (about two or more axes which intersect at a central point).
Sequence	The arrangement of objects in a way that facilitates the movement of the eye through the information displayed. In Western culture the eye is trained to move in horizontal lines from top-left to bottom right. The eye moves most easily from big to small, bright to subdued, colour to black-and-white and irregular to regular objects.
Cohesion	The degree of use of similar aspect ratios (the ratio of width to height) in multiple-window systems.
Unity	The appearance of the elements as a visual totality. Elements should be similar in terms of size, shape and colour. The distance between elements and the distance at the margins of the figure should be similar.
Proportion	The comparative relationship between the dimensions of elements and those of 5 aesthetically pleasing proportional shapes: the square, square root of two, golden rectangle, square root of three and double square.
Simplicity	Directness and singleness of form, achieved by optimising the number of elements and minimising visual alignment points.
Density	The proportion of the screen / canvas covered by objects. Screen density levels should be reasonably minimised.
Regularity	The uniformity of elements. Horizontal and vertical alignment points should be standard and consistently spaced. The number of alignment points should be minimised.
Economy	The careful use of elements to get the message across as simply as possible. As few styles, displays, techniques and colours should be used as possible.
Homogeneity	The evenness of distribution of elements across the four quadrants of the screen / canvas. Evenness means that each quadrant should contain nearly equal numbers of elements.
Rhythm	Regular variation, the extent to which elements are systematically ordered. Determined by variations in arrangement, dimension, number and form of the elements.
Order / Complexity	The sum of the previous 13 measures for layout. Complexity refers to a lack of order; extreme complexity and total order may thus be considered opposite ends of the same scale.

Table 2.3: Fourteen aesthetic measures from Ngo et al [NTB00]

2.3.4.4 Domain-specific Layout Criteria

Section 2.3.2 noted the high number of diagram types across a variety of domains. Furthermore, it is a well-known problem that multiple, competing notations may exist for the same purpose, within the same domain. Ambler [As04] argues that well-known notation should always be preferred over esoteric notation. Ambler notes that even *within* a specific diagramming standard, a "kernel notation" of the most well-known features, often consisting of no more than 20% of the available specification, can be used to accomplish a majority of communication. Ambler argues that this kernel notation should be used whenever possible and less-known features avoided.

It is not feasible to provide an overview of all diagrammatic guidelines. Overviews of databases of diagram standards are provided in [ANSI05] and [ISO05]. The research of Purchase et al [PAC02] into the layout of UML diagrams was summarised in section 2.3.4.1; Eichelberger and von Gudenberg [EG03] provide further insight into this domain.

The U.S. government Defense Finance and Accounting Service document provides a set of typical *Diagramming Guidelines* [Dfa04] which specify standard practices and aim to accomplish increased consistency, improved readability and improved pattern recognition through consistency. Introductory comments concentrate on deletion and consolidation of diagrams and standard formats for diagram legends. Subsequent sections provide domain-specific guidelines for Business Process Model (BPM) diagrams, Function Hierarchy Diagrams (FHD), Entity Relationship Diagrams (ERD) and Server Model Diagrams (SMD). Entity Relationship Diagrams are the most frequently encountered of these diagram types within an academic context.

2.3.6 Summary

Section 2.3 introduced the concepts associated with educational diagrams. A diagram can be viewed as a collection of nodes connected by lines. Diagrams have been used as abstractions to represent information for several thousand years and are currently used to illustrate concepts and assist design processes in a large number of academic disciplines. Categories of interest in diagramming research are representation, message, relation between representation and message, task and process, context and convention and mental representation. Systems for creating diagramming editors can

be categorised into the three approaches of multi-domain diagram editors, frameworks and diagram editor generators. In assessment, approaches to diagramming have included a CBA-integrated framework for diagram-based CBA and research into the assessment of "imprecise" student diagrams. Diagrams with an aesthetically good physical layout can help to reduce reader confusion. Criteria approaches applicable to diagramming layouts can be drawn from the fields of graph layout and user interface design aesthetics.

2.4 Chapter Summary

This chapter introduced the areas of CBA, formative assessment and diagramming. Section 2.1 defined CBA in relation to other areas of learning technology in terms of the number and types of processes that are automated. CBA was defined, the motivations for the development of CBA technology were considered, and a brief history of CBA development was provided. CBA has both practical and pedagogical advantages, while its practical limitations are primarily infrastructural. CBA's pedagogical limitations relate to its perceived inability to assess the higher cognitive levels as defined in taxonomies such as Bloom's. Attempts to minimise these pedagogical limitations may include the careful construction of objective questions or the automation of question types traditionally used to assess higher cognitive levels. An overview of the fixed-response and free-response CBA question types was provided.

Section 2.2 introduced formative assessment. Formative assessment has concrete pedagogic benefits and has been shown to improve student learning but it is seen as a resource-intensive assessment mode. Strategies for reducing resource-intensive assessment, including mechanisation strategies, were considered. Criteria were presented by which the effectiveness of formative assessment feedback can be analysed.

Section 2.3 introduced diagrams in education. Diagrams are currently used to illustrate concepts and assist design processes in a large number of academic disciplines. Systems for creating diagramming editors can be categorised into the three approaches of multi-domain diagram editors, frameworks and diagram editor generators. In assessment, approaches to diagramming have included a CBA-integrated framework for diagram-based CBA and research into the assessment of

"imprecise" student diagrams. Diagrams with an aesthetically good physical layout can help to reduce reader confusion. Criteria approaches applicable to diagramming layouts were drawn from the fields of graph layout and user interface design aesthetics.

Chapter 3

CBA approaches for formative assessment and diagrams

Introduction

Previous work on formative assessment using free-response CBA across multiple diagrammatic domains in a generic, extendable way is undocumented in the literature. Formative assessment using CBA techniques has hitherto been largely conducted within fixed-response domains such as multiple-choice questions. Formative assessment in free-response domains is less common in the literature but some work has been documented in domains such as technical essays. Diagrammatic CBA is relatively uncommon in the literature but several systems have been documented. A commonly cited CBA system is CourseMarker, which incorporates the DATsys framework for diagrammatic exercises. CourseMarker is the successor to the successful system Ceilidh and is used as a platform for this work.

Section 3.1 outlines the approaches used in the literature to provide formative assessment using CBA. Most examples in the literature use multiple-choice questions as the domain and can be categorised into those systems which are based around pre-existing software, usually commercial systems, and those systems which were developed entirely by the educators themselves. The approaches are reviewed and the feedback mechanisms are examined in light of the formative feedback guidelines provided in section 2.2.5.

Section 3.2 outlines documented approaches to conducting CBA in diagrammatic domains. Approaches are compared in terms of their flexibility (which is sometimes carefully restricted in terms of student interaction) and their marking mechanisms.

Section 3.3 provides an in-depth examination of the CourseMarker and DATsys systems which were used as a platform for this work. The Ceilidh system is described since it provides important historical and theoretical background. CourseMarker and DATsys are reviewed within the context of providing formative CBA in diagrammatic domains and their current advantages and limitations are discussed.

3.1 Using CBA technology to provide formative assessment

It is neither possible nor necessary to provide an exhaustive catalogue of all examples in which automated assessment has been used for formative purposes. Stephens and

Mascia [SM97] noted the high usage of automated assessment technologies in 1997 and the trend has been for increased use of such technologies in a process which has been described as "inexorable" [Br02]. Instead it is useful to compare examples of various approaches taken to try to automate the process of formative assessment and to contrast their relative merits. Some advantages, such as time-saving, are common to nearly all automated assessment systems and so it is not useful to concentrate on them here. Denton [Dp03], for example, reports considerable time-savings simply by automating the process of returning feedback to students using an email system based upon Microsoft Word.

A naïve comparison of CBA assessments might categorise examples based merely upon whether the assessments used a commercially available package or had been developed 'from scratch' by the academic staff. Considerations such as affordability are important in the implementation of a CBA system and authors of in-house systems routinely bemoan the level of resources required in the development of their systems. However, as section 2.2.1 has previously argued, the primary deliverable of formative assessment is feedback, and the examples considered here demonstrate widely varying levels of feedback using the *same system* as a base for different assessments. Therefore it is clear that any useful review of a formative CBA example must consider both the practical system *and* the pedagogical approach: that is to say, the technical abilities of the system and the level of feedback which is actually provided to students to assist learning. In order to accomplish this, the framework for effective formative feedback outlined in section 2.2.5 will be used as a benchmark when assessing the CBA examples.

Overviews of prominent automated assessment systems already exist. Charman and Elmes [CE98b] provide examples intended to be used by educators wishing to develop their own systems. Rawles et al [RJE02] provide a review of systems in terms of technical capability and ease of incorporation into teaching structures. Symeonidis [Sp06] provides an overview of prominent CBA systems in terms of developmental history, system requirements and automatic marking capability. The aim is not to repeat this material here, but to review the systems in terms of their formative assessment potential.

Section 2.1.6 defined and contrasted fixed-response and free-response CBA. Fixed-response CBA is the easier to develop and therefore examples are more common. Sections 3.1.1 and 3.1.2 consider, respectively, fixed-response and free-response CBA used for formative assessment purposes.

3.1.1 Fixed-response formative CBA: a review

Due to the relative ease of implementation associated with fixed-response CBA, together with the potential for standardised requirements across multiple assessments, fixed-response formative CBA is often accomplished using commercially available software. A brief overview of several such systems was provided in section 2.1; the most commonly documented platform for CBAs is *QuestionMark* [BSP+03], which has been referred to as "*something of an industry standard*" [RJE02] in overviews of the field. Several CBAs based upon commercial platforms such as *QuestionMark* are considered, with emphasis upon the difference in formative feedback delivery. The section then concludes with a review of several systems developed by educational institutions themselves; the advantages and drawbacks of this approach are also briefly considered.

3.1.1.1 Using existing platforms

Charman and Elmes [CE98c] describe the use of *QuestionMark* to conduct formative assessment on a first-year undergraduate module on data analysis in a Geography degree. Short tests are conducted using questions selected randomly from a stratified data bank. The tests typically take 10-15 minutes to complete, but no strict time limit needs to be imposed since the assessment is formative. To motivate students the tests are integrated into the teaching of the course and a two-part assessment strategy is used: several questions typically relate to the results of earlier practical experiments and an end-of-module summative examination is conducted which is similar to the earlier formative tests.

Charman and Elmes note that development costs included employing a research assistant for 2 months to write the questions and feedback for the data bank. Since a commercial package was used there was also an initial outlay to purchase the software. Feedback from the system has allowed teachers to monitor student progress. Charman and Elmes note that there has been little improvement in the very

poorest and the most able students, but that a central group of students labelled "struggling" could be seen to be helped by the system considerably. Student response to the CBA was generally positive: 64% agreed that the system constituted a good way of learning while 56% agreed that the CBA was an improvement over the previous assessment forms.

Opportunities to improve performance are constrained by this system since the tests cannot be retaken, although Charman and Elmes do address this problem partially by allowing students to review their results as a revision resource. Student feedback is returned on a per-question basis and is specific to the student response; this system is not optimal in focusing on student learning and encouraging motivational beliefs, however it provides a concise method for clarifying good performance within the context of the MCQ test. Reflection in learning would seem to benefit strongly from this approach, however, since Charman and Elmes report the "*unexpected*" benefit students now found "*stimulating and interesting*" a module centred around data analysis material which was traditionally regarded as dry.

Greenhow [Gm00] provides an overview of the *Mathletics* system, again built using *QuestionMark* as a platform. Mathletics is used for both formative and summative assessment. Mathletics employs both MCQ and hotspot graphical questions. The approach outlined by Greenhow does constitute a two-part assessment strategy since a summative assessment is conducted using the software after the formative assessments have taken place. Furthermore, Greenhow emphasises that student problems are a focal point for discussion in subsequent student tutorials, facilitating teacher and peer dialogue around learning. However, the key difference between the "formative" assessments conducted using Mathletics and their summative counterpart seems to be the suppression of feedback in the summative tests. The formative assessments are still conducted in formal examination sessions to avoid "cheating" for example, and Greenhow admits that extensive use of Mathletics within a module may result in students having little experience in problem solving; paper-based tests and worksheets are used in conjunction with Mathletics to achieve these aims.

Wybrew [Wl98] describes the use of *Question Mark* to conduct CBA in Health Science modules. The CBA replaces existing MCQ tests which were previously marked using

OMR technology and conducted throughout the module. A two-part assessment strategy is used in which strictly voluntary formative assessments precede a compulsory summative assessment at the end of the module. A comparison of marks between the CBA assessments and the previous OMR assessments shows no difference. Take-up of the formative assessments is low, but this is likely to be as a result of limited access to the Faculty computing facilities on which the courseware is available. A further drawback of the current approach is the fact that the feedback mainly tells the student which answers were correct and wrong. A positive major feature of this example of CBA is that academic staff are simply required to write the assessment itself; technical issues such as converting the questions into the proprietary *Question Mark* format are the responsibility of a separate Unit for Learning, Technology, Research and Assessment within the university.

Other examples of formative Computer Based Assessments based around commercial systems such as *QuestionMark* occur frequently in the literature. Hawkes [Ht98] describes a course of automated assessment in an undergraduate Number Theory course in which the assessment is linked with a sequence of workbooks written specifically for the course; questions are variations of exemplars in the workbooks and only feedback on the exemplar questions is provided.

3.1.1.2 In-house fixed-response CBA systems

Buchanan [Bt00] reports on the use of the web-based formative assessment package *PsyCAL* to assess undergraduates in the first year of a degree in psychology. PsyCAL's infrastructure is composed of CGI scripts written in Perl; students access the system through a web browser. Students are allocated three specific weeks within a 15-week course to assess themselves using the system, although students are free to use the system outside these weeks and many take advantage of this. PsyCAL assesses MCQs exclusively, which are presented to students in short, informal tests typically numbering 20 questions. At the end of the test the student is presented with a list of those questions they answered incorrectly, together with formal references to documents which would help the student to answer those questions. The student is not presented with the correct answers to the questions since this may act as a disincentive to further student research.

Buchanan conducted two studies using PsyCAL in which the level of integration with the module was varied. Both studies demonstrated that the formative assessment tool was useful to student learning. Buchanan notes that the package operates with a test-study-retest cycle and emphasises the importance of "repeated" automated assessment for formative purposes. Buchanan notes that the system was difficult to develop and cautions against starting to develop a system from the beginning if an existing system can be used.

The feedback mechanism described by Buchanan is effective. Feedback is short and student independence is encouraged by the fact that further research is encouraged. Opportunities to improve performance are provided since the student can repeat the test any number of times. It is unclear if the system is inherently motivational but Buchanan reports that 97% of respondents to a questionnaire would be willing to use the package in other modules.

Amelung et al [APR06] describe the *LlsMultipleChoice* extension module for the open-source *Plone* content management system used to assess MCQs in Computer Science modules. LlsMultipleChoice allows the grouping of questions into 'units' and provides for the provision of instant feedback while allowing multiple submissions by students; perhaps the greatest novelty of the system lies in its "localization facility" which allows feedback to be provided in multiple languages (English, German and French).

Hall et al [HRT+98] describe *The Human Brain* CD-ROM, a multimedia tutorial on the human nervous system for use by undergraduates. Much of the project focuses on the delivery of learning materials to students, which can be navigated in non-linear pathways of the student's choosing. Two assessment components are provided: a "quick-test" option, which provides MCQs with typical feedback for incorrect responses and an accumulated score, and the "concept test" component. Learning objectives for each section of the teaching material is defined in terms of "concepts", and each test in the concept test is designed to assess understanding of a particular concept. At the end of a concept test, students are advised of which concepts they are perceived to have poor understanding and are referred to the appropriate teaching materials. The Human Brain CD-ROM links assessment and learning in a similar way to PsyCAL with similar advantages: feedback is short and student

independence is encouraged by the fact that further research is encouraged. Unlike PsyCAL, The Human Brain CD-ROM takes on the responsibilities of providing all teaching materials within the system with a consequent increase in development resources. The Human Brain CD-ROM is based upon the *Scholar's Desktop* CBL platform [BS95], but due to the modified assessment component and the need to generate the teaching materials the authors note that constructing the CD-ROM involved "*a huge amount of resources (both money and academic time)*".

Culverhouse and Burton [CB98] report on the use of *Mastertutor* to assess undergraduates studying Electronics and Electrical Engineering. *Mastertutor* is based upon the format of a "music master-class": the system sets a problem, provides information resources and accepts the student's solution in the form of a questionnaire. The student is then presented with a mark and shown a valid solution in order to define what constituted good performance for the assessment.

The system is used as part of a "feedback cycle" where the solutions are discussed while the problem is still fresh in the student's mind. However, the student is not typically allowed the opportunity to improve their performance through resubmission; the impact of this would be limited in any case since the student has already seen the optimal solution. It is doubtful how much reflection in learning occurs, although dialogue around the assessment is clearly prioritised. The provision of marks to the students raises questions about the extent to which the system is motivational, but it is clear that feedback is delivered in good time and the objectives of the task are exposed.

Paul and Boyle [PB98] describe a CBA system used to assess second year palaeontology undergraduates. The assessment is designed to be simultaneously formative and summative. Frequent assessments are conducted throughout the module, feedback is given and students can be assessed by the system several times. However, marks awarded by the system count summatively within the module; this means that care must be taken to generate different tests each time the student repeats the assessment to avoid students simply repeating the assessment to gain higher marks. This is achieved by selecting questions from a large question bank. To prevent students repeating assessments to re-attain a previous (high) mark, the

highest mark across all submissions for the assessment is counted for summative purposes.

Paul and Boyle note that time is saved by both teaching staff and students and acknowledge the importance of providing feedback in CBAs so that "*students can learn from them*". However, the mix of summative elements with the formative assessment results in some rather awkward compromises. Feedback cannot be acted upon because the next submission involves a different test; furthermore, students know this in advance, so it is unclear how much attention they are likely to pay to the feedback provided to them. Experience with conducting frequent assessments in other domains with a dual formative-summative purpose has shown that students are likely to sit repeated assessments merely as part of a "gambling" strategy to chance on higher marks [BBF+93, Or98]. Paul and Boyle note that marks are not consistently better than before CBA was introduced.

Vendlinski and Stevens [VS02] outline an approach which uses CBA technology to provide information about student learning to educators, relating to the seventh criteria for good formative assessment feedback outlined in section 2.2.5. The *Hazmat* system is based upon courseware called IMMEX, a web-based CBA tool which allows teachers to present domain-specific "simulations" to students. *Hazmat* is used to assess high-school chemistry students, using a simulation in which students are expected to guess the identity of a succession of chemicals by accessing information presented by the system. Feedback to the student informs them of the correctness of their choices. Feedback to educators is more complex. The system keeps a record of the information accessed by each student before they attempted to guess a chemical identity. Vendlinski and Stevens use an artificial neural network to identify groups of similar performances from the data and then further analyse the features of the performances in each group to identify the *strategy* represented by each cluster. The distribution of students across clusters can be calculated and mathematical Markov models used to determine the distribution of students after each student attempts to solve a given number of successive cases within the problem set. The effectiveness of student strategies was determined the probability of producing a correct answer, and Vendlinski and Stevens developed a model to allow the probability of a student changing strategy to be determined based upon the information accessed by the student.

Vendlinski and Stevens' research provides a credible strategy for allowing student understanding of individual *concepts* within a course to be analysed using data generated by fixed-response CBA. This information can then be fed back to educators in order to improve course teaching; Vendlinski and Stevens' research serves as a reminder that teachers should benefit from good formative feedback in addition to students (the 7th criterion of the framework for good formative assessment feedback provided in section 2.2.5). The properties of the *Hazmat* system, however, would appear to be difficult to generalise across many assessment and teaching domains and the construction of a multiple-choice testing environment to use the same techniques would probably appear contrived to students. Furthermore, the amount of effort expended in creating the assessment is very large since not only the assessment but also the teaching materials must be authored specifically for the individual assessment.

Many more attempts to automate formative assessment using bespoke CBA systems have been documented in the literature. Bull [Bj93] provides several useful case studies including the *CALM Project*, which is used to teach mathematics, primarily calculus, to engineering undergraduates. Students can progress through tutorials at their own pace and access formative MCQs throughout. A summative assessment at the conclusion of the module is conducted in a conventional manner.

3.1.1.3 Implications for formative assessment using CBA

The examples reviewed here provide several key lessons when attempting to automate the formative assessment process using CBA software:

- The pedagogic approach to assessment, especially the design of feedback, can be at least as important as the technical capabilities of the CBA system in determining the success of the formative assessment in assisting student learning;

- A two-part assessment strategy, involving using a summative assessment component to act as a motivator for the formative assessment exercises, may increase student participation in the formative assessment process;

- Restricting student access to the CBA system will result in poor student attendance, limiting the formative impact of the assessment;

- Mixing the formative assessment with a simultaneous summative element, rather than using a two-part assessment approach, can confuse the pedagogic approach of the assessment and limit student learning opportunities; Axelsson et al [AMW06] discuss the difficulties of mixing formative and summative assessment aims even in seminar sessions benefiting from high educator participation;

- Constructing CBA systems "from scratch" is time-consuming and expensive and should only be undertaken if commercial systems or existing academic CBA platforms cannot demonstrate required functionality; for fixed-response assessment designs this is now unlikely;

- Formative assessment using CBA often benefits certain student profile groups more than others;

- Feedback should be linked to learning materials;

- Constructing in-house learning materials may be prohibitively resource-intensive; linking feedback to papers, textbooks and website references is an acceptable substitute which can encourage student research;

- Students should be allowed to repeat assessment questions in order to correct mistakes;

- Linking sections of the assessment to specific concepts can assist students in identifying and correcting shortcomings in their understanding;

- Formative assessment should provide feedback to educators as well as students in order that subsequent teaching processes can be improved.

3.1.2 Free-response formative CBA: a review

Section 2.1.6 outlined the reasons for the relative difficulty in developing free-response CBA. This difficulty has resulted in free-response CBA systems being much fewer in number than their fixed-response counterparts and also explains why those

free-response systems which do exist have been created by educational institutions themselves since no major commercial software is available. Section 3.1.2.1 provides an overview of the available systems, while section 3.1.2.2 provides a brief look at the conclusions which can be drawn from the examples. Diagrammatic CBA systems are not featured here since they will be examined in more detail in section 3.2. CourseMarker and DATsys are not featured here since they will be examined in detail in section 3.3.

3.1.2.1 Formative assessment capabilities of free-response CBA systems

Joy et al describe the *BOSS* system [JG04, JL98], a system to allow online submission and automated testing of programming assignments which utilises a hybrid CBA approach in which student submission is automated and student programs can be compared against test data but the assessment and feedback process are carried out manually by the lecturer. Checking the program against test data can be accomplished either by the student to assist in the development of the program or by the educator to assist the assessment process. However, Joy et al argue that a full CBA approach is unable to award fair credit to novel solutions and thus teaches students in a "prescriptive" way. Originally developed, like Ceilidh, as a command line environment, BOSS has been updated several times and now has a client-server architecture with a relational database to store data and a choice of either Java or web-based client for student interaction. BOSS has been used to assess courses in Pascal, UNIX shell programming and C++. Joy et al report considerable practical advantages, especially in administrative matters, a positive student response and a reduced marking time.

BOSS has certain pedagogical advantages. The system of allowing students to run automated tests on their programs before submission allows improvements to be made to the solution and promotes introspective self-assessment. Certain other factors, such as the level of motivation encouraged by the feedback, the clarification of good performance and the concentration on student learning, are unchanged from traditional assessment precisely because they *are* conducted using traditional means. Furthermore, BOSS has proved useful in information collation and is able to provide accurate information to educators. Problems associated with BOSS from a formative assessment standpoint centre around the intervention by the educator at the point of

assessment. Timeliness of feedback to the student is entirely dependent upon the individual educator, rather than guaranteed, and in any case will never successfully rival the near-instant feedback times associated with full CBA systems. Furthermore, given that each submission must be assessed manually, it is unlikely that multiple student submissions on a large scale could ever be feasibly allowed. The implications of this are that feedback might be returned to the student at a time when their solution is no longer fresh in the mind, and that few or no opportunities to repeat the assessment, for the purposes of acting upon feedback, are likely to be allowed. Fundamentally, conducting formative assessment as an iterative cycle in which the student can improve their solution over the course of multiple submissions while receiving several sets of motivational feedback, is not feasible with the BOSS approach. This is a disadvantage recognised by Joy et al [JG04] where they note the formative potential of a fully automated approach such as that taken by CourseMarker.

Jackson and Usher [JU97, Jd00] describe the *ASSYST* system. Like BOSS, the UNIX-based ASSYST uses an approach which is a hybrid of CBA and traditional assessment. The pedagogical advantages and drawbacks in terms of formative assessment potential are therefore similar to those of BOSS.

Daly [Dc99] describes *RoboProf*, an online teaching system structured as a formative, coursebook which presents students with information on programming topics and then automatically assesses student exercises that cover those topics. RoboProf runs under UNIX and interacts with students via a Java applet, is designed to be modular and scalable, allows any number of submissions and does not penalise failure. Daly reports on the use of RoboProf to assess a C++ programming course. RoboProf's assessment mechanism is based upon running student solutions against test data and examining the results in a similar way to that of *Ceilidh*. Student feedback is based around revealing the model solution to the student.

RoboProf clarifies good performance through providing students with the model solution, provides ample opportunities to improve performance through allowing unlimited submissions and provides positive, motivational beliefs through its policy of not penalising failure. On the other hand, reflection and dialogue around learning are likely to be minimised since the student is presented with the model solution.

Furthermore, feedback information is not focused on the student learning process since it is overly problem-specific and provides no motivation to conduct further research. Spacco et al [SHP+06] describe the *Marmoset* system's attempt to overcome these limitations. Sections of the instructor's private test cases are released to the student using a system of time-based tokens. In standard configuration, only three tokens may be redeemed per day. Spacco et al argue that this motivates students to begin work early since more help will be available.

Von Matt [Mu94] describes the *Kassandra* system which is used to assess student programming assignments in Maple and Matlab. Kassandra's primary focus is on summative assessment but the assessment is conducted throughout the course and feedback is provided. Due to the summative impact of the assessment, security is a key feature of Kassandra. Kassandra requires students to modify their code prior to submission, although Winters [Wt04] argues that this is an acceptable requirement in a domain such as computing. Kassandra requires very precise output from student programs.

Kassandra promotes dialogue around learning since students must be aware of the Kassandra system while developing their solutions if they are to modify their code to conform to Kassandra specifications. However, Kassandra fails to provide opportunities to improve performance, to adequately clarify good performance, to encourage positive motivational beliefs or to keep feedback information focused on student learning, all because of the constraints imposed by the summative nature of the assessment process.

A similar approach to Kassandra is documented by Oliver [Or98] and by Douce et al [DLO+05] although not all of the approaches require students to tailor their solutions to the same extent; these approaches are heavily influenced by the Ceilidh system which is examined in detail in section 3.3.1.

Another system heavily influenced by Ceilidh and CourseMarker is the EduComponents system described by Amelung et al [APR06]. Amelung et al acknowledge their experience with systems such as CourseMarker. The EduComponents *ECAutoAssessmentBox* system is written as an extension of the open-source content management system *Plone*; a central priority in the development of the system was achieving integration with the departmental system already used for

the delivery of materials to students. The EduComponents assessment technique was based upon work by Saikkonen et al [SMK01], which demonstrated that automated assessment of functional programming languages can be undertaken by directly comparing the values of functions in the student and model solutions. ECAutoAssessmentBox assesses student programming exercises in Python, Haskell, Scheme, CommonLisp and Prolog. Amelung et al discuss the vital role of formative assessment and report that use of the EduComponents ECAutoAssessmentBox has increased student motivation in programming exercises. However, the system of feedback rests upon awarding the student one of two states, *Accepted* and *Rejected*. Such a system of classification is not motivational to struggling students and, furthermore, the system does not provide the opportunity to resubmit after one of the states has been awarded. Grading and brief feedback is also provided, but since the student cannot resubmit then opportunities to improve, promotion of student dialogue and the encouragement for self-assessment are not provided.

English [Ej04] describes the process of automated assessment of student GUI-based programs in Java using JEWL, a set of Java packages created with the aim of allowing novice programmers to construct GUI programs "*from the 'Hello world' stage onwards*". A "test harness" is used to generate sequences of events which the JEWL event loop allows to be processed as a stream of characters. English reports that the assessment process is as yet in the early stages of development, but that students seem to be motivated by their ability to create Java programs as opposed to command line based programs (which are seen as unrelated to real-world Java programming and therefore dismissed as toys). English reports that only interface functionality can be assessed and considers the drawbacks of being unable to assess the layout of the user interface. The feedback regime is not explicitly described in the literature, although it is implied that students are provided with a report of those features in regard of which their program failed to conform to the program specification. This has obvious drawbacks in terms of motivation; English also reports frequently infuriated student complaints to the system administration but dismisses most concerns as being to do with misread questions. Dialogue around learning is obviously generated, however, and opportunities to improve are provided. Gray and Higgins [GH06] describe a system for the assessment of GUI-based student Java programs using CourseMarker which makes use of the standard CourseMarker feedback mechanism.

3.1.2.2 Implications for formative assessment using CBA

Section 2.1.7 noted the advantages of free-response CBA in terms of the opportunities provided for assessing higher cognitive learning levels. The review of the formative potential of existing free-response CBA systems raises several key implications with relevance to this work:

- Fully automated CBA systems may not account for particularly novel solutions and can be criticised for pedagogic "prescriptiveness";
- A trade-off exists between fully automated CBA approaches and human-assisted approaches which, while more able to cope with student novelty, may result in less timely feedback and fewer chances for the student to improve;
- Requiring precise input from students encourages awareness of the assessment process and has fewer disadvantages in scientific disciplines where questions can be worded as 'specifications';
- Free-response CBA systems can allow the student to construct solutions which they feel are relevant to the real world—this acts as a motivator;
- CBA can be helpful to educators in collating often complex feedback results derived from free-response exercise submissions;
- CBA can provide timely feedback;
- CBA can assist formative assessment through an iterative cycle: test–feedback–retest;
- A system which does not penalise failure can successfully motivate learning;
- Providing model solutions to students may fail to encourage student research — a more effective method is to construct feedback;
- CBA can be used to construct feedback which is not failure specific;
- Allocating simple states or grades focuses on failure and may de-motivate students;

- Questions should be carefully phrased to avoid confusion, since misread questions often infuriate students.

3.1.3 Summary

Section 3.1 reviewed existing approaches to automating the process of automatic assessment. Section 3.1.1 reviewed examples of providing formative assessment using fixed-response CBA systems. The design of the assessment itself is more important to the success of the formative assessment than the technical capability of the system. Some systems are built from scratch by the educator, allowing a flexible and highly targeted approach but at the expense of high development costs. Systems built on top of commercial platforms such as *QuestionMark* require less resources and can provide successful formative assessment so long as the assessment itself, and especially the feedback, is carefully designed by the educator. Section 3.1.2 reviewed free-response CBA systems. Free-response CBA can be used to provide timely feedback and encourage discussion around the learning and assessment processes, but sometimes at the expense of tolerance in marking. Human-assisted approaches are more flexible but do not allow the same potential for timely feedback or multiple student submissions.

3.2 CBA approaches in diagrammatic domains

Tsintsifas' [Ta02] approach to conducting CBA within diagram-based domains aspired to be generic, as opposed to domain-dependent. The DATsys object-oriented framework was developed with the intention that the extensions could be diagram editors in any conceivable domain within a CBA context. CourseMarker's Generic Marking Mechanism was, similarly, designed to be flexible enough to allow any domain to be assessed for which specific marking tools could be constructed. CourseMarker and DATsys are described in detail in section 3.3. Since section 2.3 illustrated that one of the key advantages in assessing diagram domains is their interdisciplinary potential, this work will continue Tsintsifas' generic approach: the aim will be to provide a framework for the formative CBA of diagram-based domains, within which assessment for individual domains can be conducted through extension and parameterisation.

This section considers other approaches to the automated assessment of diagrams which are described in the literature, of which there are four: the *TRAKLA2* system described by Malmi and Korhonen [MK04], the PILOT system described by Bridgeman et al [BGK+00], the diagram comparison system described by Hoggarth and Lockyer [HL98] and a more recent body of work developed by Thomas, Waugh and Smith at the Open University, UK [TWS05], which was published as this work was being undertaken. Each review considers the level to which the work is domain-specific, the user-interactivity of the student interface and the ability of the approach to provide feedback to students. The reviews are in order of generality, with the most domain-specific system first.

3.2.1 TRAKLA2: a review

Malmi and Korhonen [MK04] describe the *TRAKLA2* system, the successor to the earlier *TRAKLA* [HM93]. TRAKLA2 is a domain-specific CBA system used to distribute visual algorithm simulation exercises to Computer Science undergraduates on a Data Structures and Algorithms course. Distribution of the exercises is accomplished over the web as a Java applet; the interactive environment (Figure 3.1) is used by the students to solve the exercise by manipulating the available tools.

Exercises cover such topics as binary search tree insertion and deletion, and insertion into AVL-trees, red-black trees, digital search trees and radix search trees. Typically, a student drags and drops graphical entities (keys, nodes and references) onto the drawing canvas in an attempt to simulate the operations performed by the algorithm defined in the question. Therefore, the user interface can be viewed as a more complex variation on standard graphical hotspot interaction techniques as described in section 2.1.6.1. Malmi and Korhonen are keen to emphasise that the student exercise is individually tailored after each submission — what this means is that certain exercise parameters are randomised in order to alter the student solution. This strategy has been adopted for use with TRAKLA2 to prevent plagiarism.

Student feedback is returned as a mark: the number of correct steps is given as the mark, while the total number of correct steps is given as the highest possible mark. A standardisation system is used for inter-exercise consistency. The student can also request to see the model solution. If this occurs then the model solution is displayed, but grading suspended until the exercise has been re-initialised with different data.

The design of TRAKLA2 emphasises the collection of data regarding student performance. Data is logged every time a student initialises the exercise, asks to be graded, requests the model solution or submits an exercise. The number of submissions allowed is unlimited.

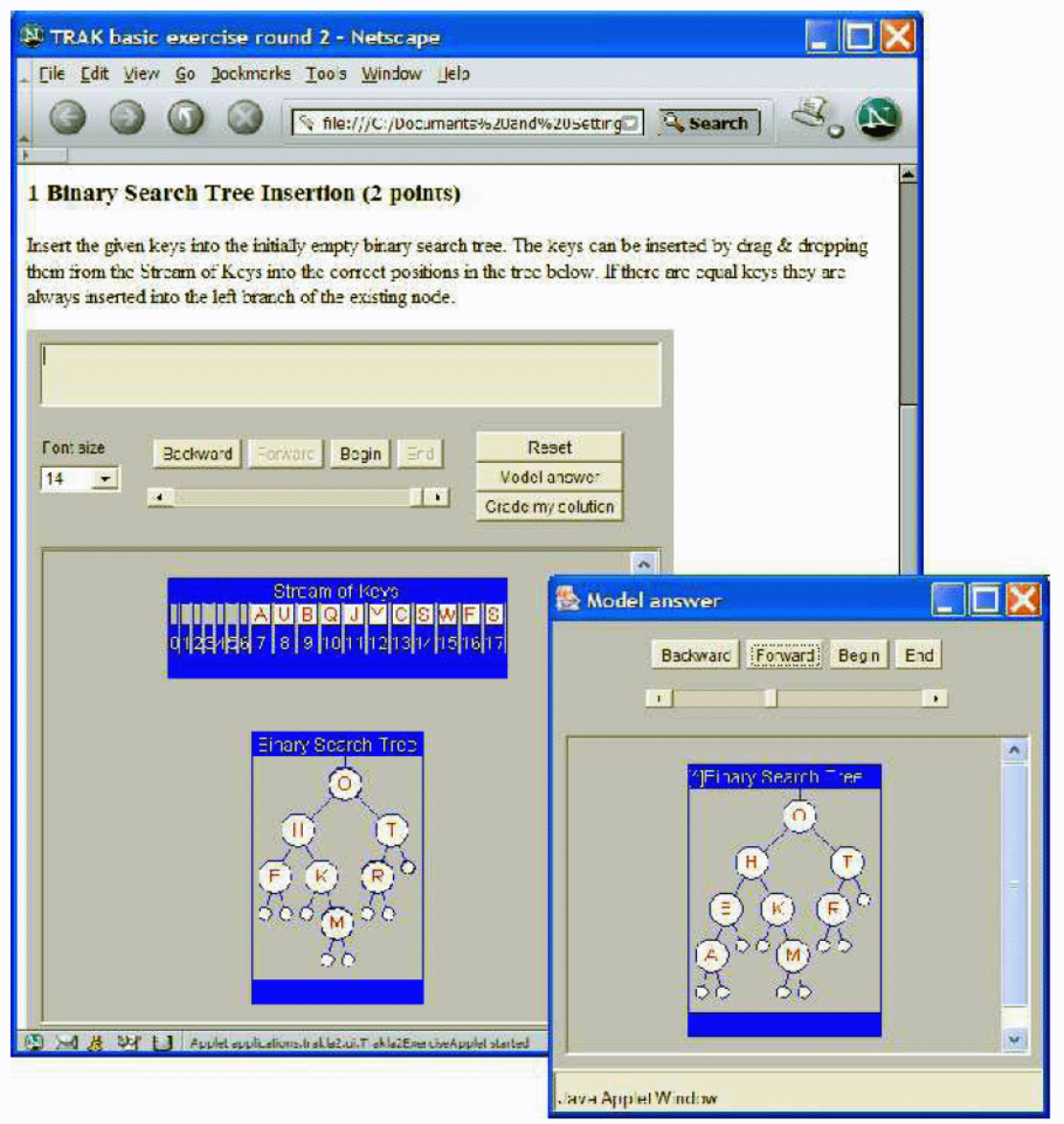

Figure 3.1: TRAKLA2's student applet and model solution window [MK04]

TRAKLA2 provides in-depth information to educators; Malmi and Korhonen's paper [MK04] concentrates substantially on student mark analysis. TRAKLA2 also provides ample opportunities to improve since submissions are unlimited, and clarifies good performance through allowing the student to request the model solution. Malmi and Korhonen emphasise their belief that the randomised exercise elements result in

sustained student interest and increased learning. Unfortunately, this randomisation relies upon the domain and user interaction restrictions of the system to be feasible. The motivational potential of the feedback, in concentrating upon the proportion of the model solution correctly identified by the student, is ambiguous at best. For similar reasons, the feedback fails to provide information focused upon student learning, instead focusing on student mistakes. Malmi and Korhonen fail to document the potential for dialogue around learning in terms of student feedback to educators, although their insistence on guarding against plagiarism seems superfluous for formative assessment. A final examination seems to act as a successful motivator for student participation. Finally, although TRAKLA2 was popular with students and encouraged learning, its context is entirely domain-specific.

3.2.2 PILOT: a review

The PILOT system described by Bridgeman et al [BGK+00] was designed to accomplish three goals: use in class by educators as a demonstration tool to aid exposition, use by students in entering online solutions to randomly generated instances of questions and as a grading tool for formative purposes. PILOT has a degree of commonality with the TRAKLA2 system described in section 3.2.1: PILOT's user client is distributed as a Java applet and the system is used to conduct formative assessment of graph problems such as the minimum spanning of a tree, shortest path algorithms and breadth- and depth-first searches. Furthermore, PILOT aims to reduce plagiarism by only allowing students to solve problems generated "on the spot"; this is an attempt to prevent students from using the system to help them solve their "homework" problems, which are summatively assessed.

Bridgeman et al note the proven usefulness of graph and algorithm visualisation systems in learning; a recent such system is described by Brusilovsky and Loboda [BL06]. To begin an assessment, a student chooses a problem type and a random instance of the problem type is generated. The graph of the problem is then drawn on the screen: this necessitates the use of a Graph Generator system to conduct automatic layout of the graph using algorithms derived from Di Battista et al [BGL+97]. To indicate their solution to the system, the student clicks on the edges, in

order. This generates a list of edges which constitutes the solution. An illustration of an exercise and the student solution is shown in figure 3.2.

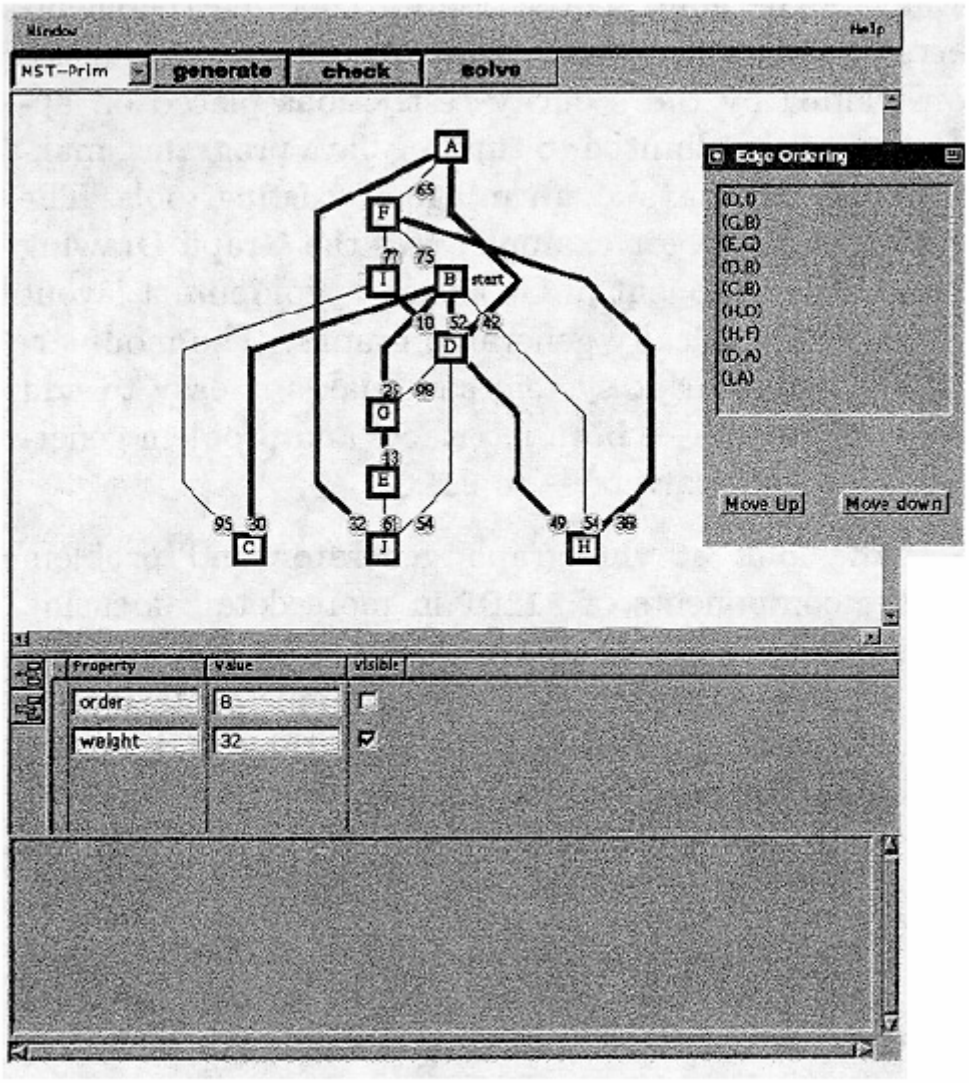

Figure 3.2: Example exercise and student solution using PILOT [BGK+00]

Bridgeman et al emphasise the need to provide partial credit to a student for an "almost" correct solution. The solution checker operates slightly differently depending upon the exercise type: solutions may vary from simply checking the list of edges in the solution tree, to checking the *order* of the list of edges. In the case of non-unique solutions, the checker must evaluate if a student solution which differs from the generated model solution is equally valid — this is accomplished by running a simulation of the graph and comparing the weights of the edges denoted by the two. Feedback is provided in the form of brief comments denoting precise student errors, for example "*Edge (a,c) should be replaced by the lower-weight edge (a,b)*".

The system uses a system of penalty marking in which one mark is deducted from the total for each incorrect edge.

Like TRAKLA2, some of PILOT's most central features are based around domain-specific properties of the exercises, together with limitations to the user interaction. Bridgeman et al's reference to graph layout within a CBA context is interesting, but it is important to note that the context is in the construction *by the program* of aesthetically acceptable graphs which are then manipulated, in a limited way, by the user. PILOT does not assess the aesthetics of student diagrams.

PILOT clarifies good performance to the students by suggesting specific changes to their solutions in terms of edges. While this information concentrates on improving the specific student solution, there is no attempt to motivate students to research on their own by referencing external material or by explaining the *reasons* why the modification to their solution is necessary. By concentrating on student errors, and by adopting a penalty marking scheme, positive motivational beliefs in students are not encouraged. Dialogue around learning and self-assessment is promoted by the tool since students are allowed to discuss their solutions with each other and since PILOT is used as a demonstration tool — again, however, it is unclear how much student improvement is gained through increased understanding as opposed to blindly adopting the system's corrections. Bridgeman et al's insistence that the system not be used to provide help with homework assignments implies worry on their part that students may not learn through PILOT's feedback process. Opportunities to improve are provided through allowing multiple submissions.

3.2.3 Diagram Comparison System: a review

Hoggarth and Lockyer [HL98] developed their diagram comparison system, for use with systems analysis and design diagramming methods, due to their perception that existing CASE tools did not cater for academic users, who would require assistance with the underlying methodology of developing their solutions and well as with the specific usage of the CASE tool. Hoggarth and Lockyer documented a tool which embedded Computer Aided Learning features within a CASE tool; they argued that, "*as a CASE tool fully recognises the content of a software diagram it can provide feedback based on its actual diagrams*". By the definitions outlined in section 2.1.1, the features which are described are actually CBA rather than CAL. The system documented by

Hoggarth and Lockyer is domain-specific: it is used to assess student systems analysis and design diagrams. However, the fact that it is embedded within a CASE tool suggests at least the potential for implementation in further domains, and the comparison mechanism itself is not domain-specific.

As noted in section 2.1.6.2, the verification mechanism involves the student manually tailoring their diagram to match the requirements of the system. This notion of the student labelling their solution to assist the automated assessment process is reminiscent of the Kassandra system described in section 3.1.2.1. The student must specify 'tokens' (the names of diagram components) in their solution, which are matched with tokens in the model solution in order that the system can identify the equivalent diagram components in the two diagrams. The verification mechanism then compares the diagrams as two directional "flows" of modes and connections and notes the differences in ordering between the two. Formative feedback is generated according to three specification criteria: outlining specific student errors, for example the absence of necessary flows in the student diagram; outlining student inconsistencies, such as different symbol order or connections with incorrect directionality and listing the symbol selections between the two diagrams.

The feedback system described by Hoggarth and Lockyer promotes dialogue around learning since students must be aware of the assessment process in order to tailor their solutions to the system, provides opportunities to improve through multiple submissions and provides timely feedback. It could also be argued that comparing the student solution to the model solution helps to clarify good performance. However, the danger inherent in such a feedback mechanism for formative purposes is that the information provided to students is exclusively based around differences between the student's solution and the model solution at the expense of focus on the student learning process itself. Furthermore, the feedback is not inherently motivational since it concentrates mainly on the level of student failure. Such a feedback framework may encourage students to blindly minimise the differences highlighted by the system, at the expense of promoting dialogue around learning. Listing the symbol selection between diagrams is, however, a useful tool in providing a psychological link between the student's submission and the feedback, especially if the student chooses to view their feedback at a later date.

3.2.4 Automatic Marker for Entity Relationship Diagrams: a review

This section examines a body of work undertaken by Thomas, Waugh and Smith at the Open University, UK. Much of the work reviewed in this section was undertaken concurrently with the research outlined by this thesis and it must be emphasised that, for this reason, the strategies and results achieved impacted little on the work outlined in the remainder of this thesis. Significant differences between the two approaches exist: the work described in this thesis provides a theoretical framework for the formative assessment of many diagram domains. Marking of individual domains is achieved through extension and parameterisation. The work described in this section is primarily domain-specific and aims for a deeper understanding of the structure of entity-relationship diagrams for use in a simple marking tool. This section will examine the work undertaken and review its potential for providing formative assessment.

Thomas [Tp04] describes the process of creating a tool to allow students to draw diagrams in an online examination. The drawing tool could be launched simply by the students from within the online examination and the interface was simple (consisting of only "boxes" and "links"). Some students were dissatisfied by the interface (particularly the level of screen scrolling routinely necessitated by the tool) and there was also a reluctance by students to use the system, with which they were not previously familiar, under examination conditions. Thomas concluded that the situation would have been improved had the students been more familiar with the system before the exam. An interesting feature of the results was that students tended to use spatial correlation between boxes, rather than direct links, to indicate intent. For this reason alone it was fortunate that no automated assessment process was used in this trial: it is likely that many students would have received very low marks indeed.

Smith et al [STW04] describe an initial approach to the assessment of *imprecise* diagrams. Student diagrams are said to be imprecise because required features are incorrectly defined or missing, or because extraneous features have been introduced by the student. Smith et al outline a general approach to "interpreting" imprecise diagrams which does not attempt to identify semantic structures, but instead searches for associations between nodes in the student diagram and conducts a

comparison with equivalent associations in a model solution. A mock exam was conducted in which students were asked a simple question whose answer was a pipeline diagram. Similarity measures, generating a value between 0 and 1 for each association, were used along with a system of weighting to generate a final mark. Student solutions were also hand marked and the marks compared. Smith et al noted that the results were encouraging but that it could not be proved that the automatic marker was not significantly different from human markers. This approach is similar to that used by Tsintsifas [Ta02] in the marking of UML diagrams. Thomas et al [TWS05] describe a similar approach to the assessment of entity-relationship diagrams. A tool was used to identify "minimum meaningful units" (MMUs) in a diagram: in entity-relationship diagrams an MMU is a connection between two nodes. Once all MMUs are identified an aggregation stage combines MMUs into higher level features which are then compared with a model solution. Two exercises were presented to participating students, who drew their solution online. Official marking was undertaken manually by tutors. The tool was used afterwards to generate an alternative set of marks which were then compared with the manual marks. In both exercises the correlation between marks was good. Furthermore, correlation for the second, more complex, exercise was later improved by the introduction of a feature to consider synonymous entity names.

Later work [TWS06] attempted to identify higher-order semantic structures through the use of a *cliché library*. The work was domain-specific, again concentrating on entity-relationship diagrams. A *pattern* was defined as a sub-diagram with some of its details omitted (i.e. made generic). Some patterns are considered equivalent, such as a many-to-many relationship and two one-to-many relationships, and it would be useful to be able to substitute these patterns as part of the assessment process. Re-usable patterns were stored for reference and referred to as clichés.

Thomas et al [TWS05, TWS06] describe the construction of a student revision tool. Students are presented with a collection of typical assessment questions in the domain of entity-relationship diagrams. Students draw their answers online; the user interface is shown in figure 3.3. Feedback is provided in terms of a mark and a series of entity-relationship diagrams that form the MMUs of the model solution. The student is also able to display an "interactive" version of the specimen solution. The

tool was later modified to allow the addition of patterns from the student diagram to the cliché library [TWS06] and to suggest pattern substitutions.

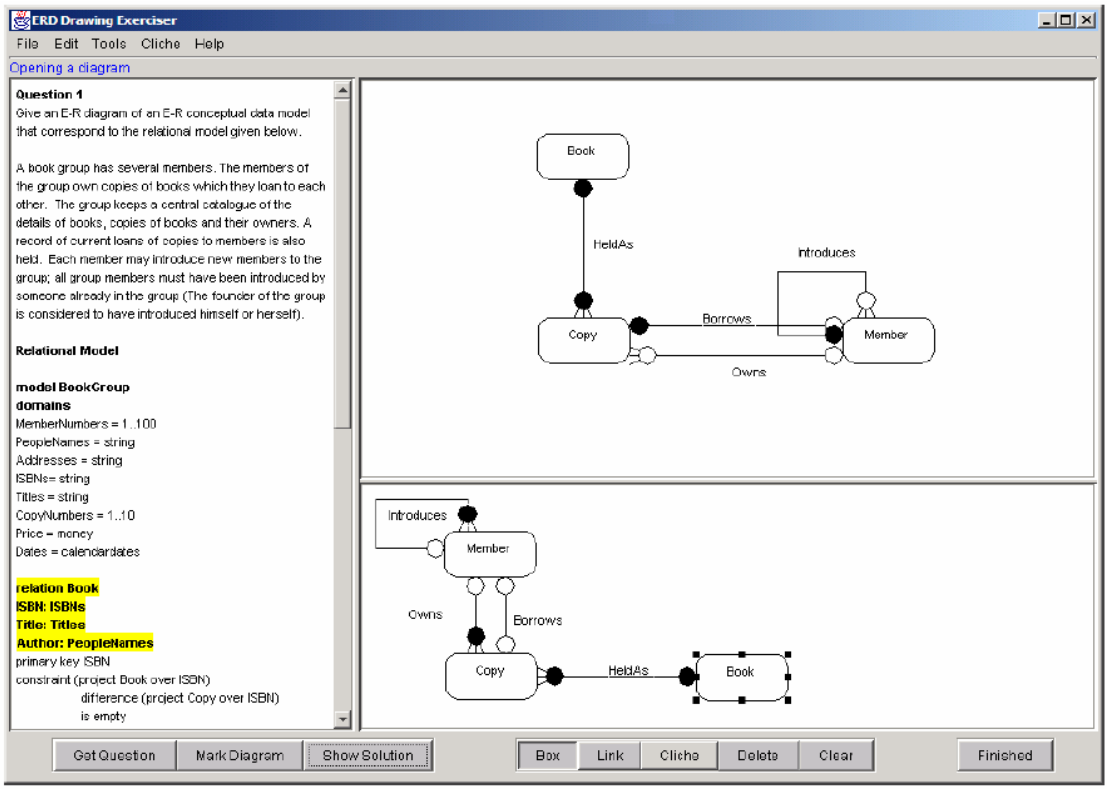

Figure 3.3: The student revision tool [TWS05]

The identification of MMUs and the creation of a cliché library have clear advantages for the automated marking process. Clichés may be equivalent but they are not always equally preferable; therefore, it is possible to award differential marks across equivalent clichés and to suggest substitutions to the student. Good student performance is thus clarified. The identification of MMUs makes understanding the structure of a student diagram possible and allows feedback to be presented as a direct comparison of student and model solutions MMUs, with individual sub-diagrams presented to the student as tailored feedback. Information is, therefore, focused on student learning. Good information is also provided to educators and further clichés added to the library.

The promotion of self-assessment and dialogue around learning may be difficult to achieve since the information provided to students centres directly upon the model solution, allowing them to tailor their solution directly. Little information is provided to motivate student research. Opportunities to improve may therefore not be

maximally effective. The system also concentrates on student differences to the model solution, which is not motivational.

These minor criticisms aside, the work that this section has described provides a rare example of truly free-form diagram-based CBA which provides a deep understanding of the diagram domain under consideration and allows tailored feedback to be provided to the student. The authors concede that their approach has been domain-specific, as opposed to this work which aims to provide a generic framework for formative CBA in diagram-based domains. The authors do express the hope [TWS06] that a similar approach can be focused upon object-oriented design diagrams at a later date.

3.2.5 Summary

Section 3.2 provided an overview of four diagram-based CBA systems. The TRAKLA2 and PILOT systems limit the free-form nature of the CBA by providing a limited number of interactions between the student and the onscreen diagrams. Conversely, the diagram comparison system described in section 3.2.3 and the entity-relationship diagram tool described in section 3.2.4 allow true, free-form CBA to occur. All four described systems are largely domain-specific and all four provide specific comments on student errors as feedback. As the framework for good formative feedback, presented in section 2.2.5, suggests, this may not provide optimal student motivation and discourages student research. The work into patterns and clichés described in section 3.2.4 represents an attempt at deep understanding of a limited diagram domain. Conversely, the DATsys framework, and the CourseMarker CBA system into which it is integrated, provide a framework for generic marking and allow marking tools to be configured by domain experts at a later date. The influential Ceilidh system, its successor CourseMarker and the DATsys framework for diagram editors in a CBA context are reviewed in depth in section 3.3.

3.3 Ceilidh, CourseMarker and DATsys

Section 2.2 emphasised the key differences between formative and summative assessment. From a CBA perspective there are, however, elements which must necessarily be present in both formative and summative assessment, for example

course management facilities. This section will give an overview of automatic assessment systems already developed at the University of Nottingham, primarily for the automated assessment of coursework in programming. The aim is to demonstrate that formative CBA can best be achieved through the extension of an existing CBA platform and that development from scratch would constitute a metaphorical reinvention of the wheel.

Section 3.3.1 begins with an examination of *Ceilidh*. Ceilidh is a system largely of interest to CBA researchers now for historical reasons; however, many key concepts in the later *CourseMarker* system were originally implemented in Ceilidh and their development can best be understood within context. Section 3.3.2 then provides an overview of the CourseMarker system, while section 3.3.3 looks at *DATsys*, the object oriented framework for CBA-based diagram editors.

3.3.1 Ceilidh

Ceilidh was a general purpose courseware system whose main use in practice was in the supporting of courses in student programming [BBF+95]. It was originally developed not as an academic research project but as an ad hoc practical aid for the teaching of programming to large groups of undergraduates; Ceilidh was developed and added to while in actual use and this ensured that its features were designed with practical teaching needs in mind [BBF+93]. Ceilidh supported features to address several problem areas: presentation of course materials, course administration and assessment of submitted student coursework. Ceilidh therefore constitutes a true CBA system according to the definition system outlined in section 2.1.1.

3.3.1.1 Ceilidh's Architecture

In CBA, Ceilidh pioneered the concept of separating the system itself from the courses it administered. The user interface was also separated, the result being that Ceilidh's architecture was a three layer-model as shown in fig 3.4. Foxley et al [FHT+99] emphasise that the interfaces between layers were well-defined, meaning that developers could modify one layer without any of the others being affected. The *database layer* was responsible for storing course information, both for running and administration and including exercise data and student submissions and marks. The

tools layer, which operates on the database layer, was comprised of a wide variety of external marking tools. The *client layer* provided the user interface to the system. Multiple user interfaces were implemented, including a dumb terminal user interface, a command line interface, an X-window interface and a web interface. Ceilidh's concept of User Views meant that each interface was capable of providing different views depending upon the registered type of the user.

The separation of system and data was central to the development of Ceilidh as a general purpose courseware system. Ceilidh was originally developed to assist in the administration of a C programming course; if the C exercises and assessment code had been hard-coded into the system itself, the task of adding new courses later would have been rendered difficult. Instead, the separated architecture allowed courses to be later developed for the assessment of SML, FORTRAN, Pascal, Modula2, SQL, Prolog, Z and UNIX-based software tools. It should be noted that Ceilidh's architecture allowed external developers to feasibly produce their own courses; the result was that a substantial number of courses were developed outside the University of Nottingham.

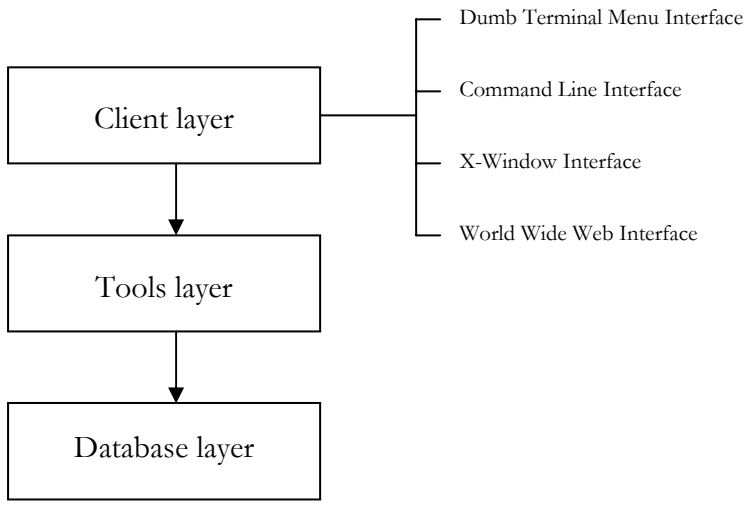

Fig 3.4: architectural overview of the Ceilidh system

3.3.1.2 Ceilidh's Course Structure

Ceilidh was capable of hosting multiple courses simultaneously. Each course had a hierarchical directory structure: within each course directory was a series of subdirectories representing exercises and units (collections of exercises). Ceilidh specified the files which must be present at each level in the directory hierarchy

[BBF96], including files for publishing information (such as unit notes or questions), and skeleton files for student solutions. The flexibility of structure allowed the type of a course to be specified within the course directory and for each exercise to specify the information it will collect upon student submission and the marking tools called upon to assess the submission.

3.3.1.3 Ceilidh's User Views

Users of Ceilidh were split into five groups: students, tutors, teachers, developers and system administrators. Each of these users had different main duties and were consequently presented, upon login, with a different User View of the system. System administrators, for example, had access to every aspect of the system with permissions to modify any file within it while a student would be presented only with those options relevant to the viewing of exercise questions and skeleton files and the submission of solutions, whereupon they would be provided with feedback. In general, for every user interface provided by the system, a course, unit and exercise level functionality specific to the User type were presented through the client layer. This approach has obvious advantages to those using the system. For convenient use, it is necessary that teachers, for example, be provided with the facilities to develop and modify exercises. However, students should not be presented with these facilities upon logging in to the system.

3.3.1.4 Ceilidh's Marking Tools

Marking of student submissions is a complex task comprising many sub-sections related to marking criteria of different types; furthermore, marking coursework within different domains will necessarily involve different operations being performed. Ceilidh's solution to this problem was to encapsulate separate components of the marking process in what were known as *marking tools*. To mark an exercise Ceilidh called all necessary marking tools for the exercise and assigned weights to the numeric values returned. The overall mark assigned by Ceilidh was a composite of the weighted marks. Invocation of marking tools occurred through a *marking action*, a configuration file in which the marking tools to be called, together with the corresponding weight for each, was defined [Ta02]. Marking actions were created for each exercise depending upon which marking tools were required. This innovation was crucial in the development of Ceilidh as an environment in which

multiple exercise types could be assessed. New exercise domains could be assessed providing that marking tools could be written to achieve the task. Marking tools were free to make use of pre-existing software, including UNIX tools; this proved to be of major practical benefit since common operations did not have to be coded from scratch.

Another key abstraction within Ceilidh was the distinction made between *dynamic* and *static* marking tools [FHT+99]. Static metrics were responsible for analysing student source code and examining, for example, typography, complexity and program structure. Dynamic metrics executed the student program and used pre-defined test data, examining program output and allocating marks based on such criteria as dynamic correctness and dynamic efficiency. Both kinds of marking tools made use of an expression recogniser known as Oracles [ZF92], which used an expanded Regular Expression notation to check for the presence (or absence) of defined tokens.

Although these ideas were originally put to the use of assessing coursework in imperative languages, similar ideas were successfully applied in other areas, such as assessing Prolog [MGH98], Z [FSZ97] and UNIX software tools [FHG96]. Apart from these courses and the central ones for imperative programming (C, C++), other courses created for Ceilidh assessed exercises in Pascal, SML and SQL [FHT+99]. Work was also undertaken on the assessment of Object Oriented Analysis and Design in the guise of the TOAD subsystem [FHT+99]; however, Ceilidh was superseded soon afterwards.

3.3.1.5 Review of Ceilidh

Benford et al [BBF+93] provide an overview of their experiences using the Ceilidh system, whilst Tsintsifas [Ta02] documents the necessity of Ceilidh's super-cession by CourseMarker. The Ceilidh system was built as a necessary response to practical circumstances and not as a research project; the authors argue that the result of this is that Ceilidh evolved to meet the actual needs of teachers and students rather than being a hollow prototype. In terms of its effect on students, Ceilidh was seen to have advantages in: **confidence building**, since early simple exercises result in positive feedback which boosts the confidence of students, especially the less able; **providing assistance to weaker students**, since Ceilidh's statistics packages can be used to spot

struggling students earlier than would otherwise be the case and **consciousness raising**, since immediate automated feedback brought about an increased willingness in students to question the marks they were given and the criteria being applied to mark their work, hence improving both the quantity and quality of discussion with students. Ceilidh was also seen to encourage students to manage their workload. Ceilidh's disadvantages for students were in encouraging the phenomena of *perfectionists* (those who would continue to submit even after a satisfactory mark had been achieved in an attempt to gain a mark even closer to 100%, hardly an optimum use of time) and *gamblers* (those who would submit many times with varying modifications in an attempt to 'stumble' upon a good mark rather than considering the problem logically). A feature was introduced making it possible to define a minimum delay between submissions per student at the discretion of the teacher in an attempt to combat these trends.

Positive effects on teaching were the most predictable: Ceilidh resulted in a reduction in marking time and proved to be an efficient course administration system. Negative effects on teaching were the very high raw marks which resulted from the combination of continuous assessment and multiple submissions; this posed problems in differentiating between candidates since the assessment served a summative purpose. Marks were also found to be tightly grouped. The authors argued that this means that Ceilidh's use presents fewer problems where criterion assessment [Kp01] is to be used, i.e. in the first, qualifying year of an undergraduate degree. Where normative assessment is to be used then Ceilidh usage might present additional problems. It is worth noting that in the case of formative assessment, high raw marks indicate success rather than a problem.

Typical usage at other institutions was to use the system but to modify the structure of the courses provided to meet institution-based practices. Marking with Ceilidh is seen to be equitable, incremental and redeemable, while Ceilidh offers facilities to detect plagiarism.

Ceilidh was, however, eventually deemed to have considerable limitations. Foxley et al [FHH+01] point to the fact that Ceilidh was difficult to install and maintain as considerable knowledge of the UNIX operating system was required. The fact that Ceilidh was based in UNIX limited the number of possible installation bases. Also,

while Ceilidh's assessment mechanisms were seen as powerful, its level of feedback to students was limited. Popularity of the system among students was also hampered by the fact that, for many years, Ceilidh's interface was based upon ASCII character terminals (fig 3.5).

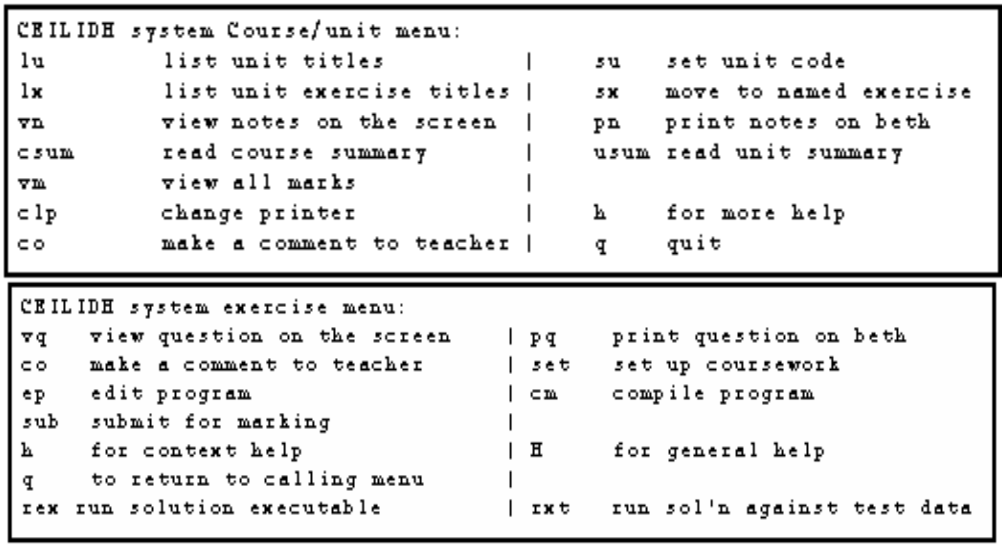

Figure 3.5: Ceilidh's dumb terminal interface [Sp06]

Furthermore, Tsintsifas [Ta02] saw Ceilidh as unsuited to the integration of diagram-based assessment into CBA. Ceilidh system dependencies were seen as too constricting to accommodate the range of exercise types which could be constructed within the domain of diagrams. The Ceilidh system had performance, scalability, extensibility and maintainability problems due to its lack of initial design; its architectural limitations in particular were considered serious enough to decrease the feasibility of diagram-based assessment.

As a result of these identified weaknesses, a complete redesign of the Ceilidh system was undertaken. The result was the *CourseMarker* system, which is examined in detail in section 3.3.2.

3.3.2 CourseMarker

The shortcomings of Ceilidh, as outlined in section 3.3.1.5, eventually led to the creation of a new system to support the full lifecycle of CBA. This system was originally entitled the *Ceilidh CourseMaster System* [FHH+01], often shortened to *CourseMaster*, and was later renamed simply *CourseMarker*. This section will briefly

describe CourseMarker and explain why it provides a suitable platform for the implementation of diagram-based CBA, including for formative assessment. Many of the ideas in the system were based upon the most successful aspects of Ceilidh and so this section will concentrate on those aspects of CourseMarker which are different from, or expanded upon, ideas from Ceilidh rather than re-stating those ideas already considered in section 3.3.1. Furthermore, this section will not concentrate on the DATsys diagramming system which is integrated into CourseMarker; this will be covered in section 3.3.3.

3.3.2.1 CourseMarker's Development Overview

Ceilidh's creation as a direct response to the needs of programming had some advantages, but ultimately the lack of design and coherent planning were major factors in the need to replace the system. It was therefore decided from the outset that CourseMarker would be conceived using object oriented (OO) methods and theory from OO frameworks and design patterns to maximize usability, maintainability and extensibility [FHH+01]. Furthermore, it was decided to develop CourseMarker in the Java object-oriented programming language as this rendered the system platform-neutral, hence increasing the range of potential installation bases over Ceilidh. Certain aspects of CourseMarker rely on UNIX-like "tools", however these can be simulated in Windows systems through the use of the freely-distributed *Cygwin* packages [Cyg98]. The vast majority of Ceilidh functionality was duplicated within CourseMarker. A more extensive comparison of Ceilidh functionality with CourseMarker functionality is provided within [FHH+01].

3.3.2.2 CourseMarker's Architecture

CourseMarker's architecture was expanded from that of Ceilidh in an attempt to reorganise functionality in a more extensible way. Commonalities and variations between the tools and data layer were identified [Ta02], with the commonalities abstracted into class hierarchies and the variation represented by extension points and parameterisation. Seven logical parts were identified and are represented within CourseMarker as servers. The client layer now communicates with Login, Ceilidh, Course, and Submission servers whilst auditing, marking and archiving servers are also included with specific functionality.

The *Login Server* is responsible for registering users, validating sessions and student login and registering when a student has logged out; the *Ceilidh Server* returns the structure of a course, manages the servers and can be used to reload servers at runtime; the *Course Server* returns the list of modules available to the user together with the module information and setup exercises; the *Submission Server* is responsible for submission attempts and receipts, together with the submission of exercises; it communicates with the *Marking Server* for the marking of exercises and the *Archiving Server*, which maintains audit trails, after exercises have been marked.

In common with the CourseMarker design, considerable effort has been expended to ensure that the system, and communications within it, is secure; this is considered especially necessary because CourseMarker is used for summative as well as formative assessment. RMI is used for convenient distribution. CourseMarker supports a range of auditing facilities, generates unique session keys for clients which are validated on every transaction and can use DES password encryption for the transmission of passwords between clients and servers. In their overview of the CAA field, Rawles et al [RJE02] single out CourseMarker as an example of a CAA system which addresses security "*unusually*" well. CourseMarker security is considered in more detail in [HGS+06].

3.3.2.3 CourseMarker's Course Structure

The logical abstraction of a Course as being subdivided into Units, each of which is, in turn, subdivided into Exercises remains unaltered in CourseMarker. Data is organized according to a hierarchical directory structure analogous to this logical abstraction. Course directories contain a subdirectory for each Unit as well as information files such as course notes. Unit directories contain subdirectories for Exercises and other information files. Exercise directories contain exercise files. Symeonidis [Sp02] provides a complete specification of this structure, including the files which must be present at Course, Unit and Exercise level.

3.3.2.4 CourseMarker's User Views

The CourseMarker system has five types of users: students, tutors, teachers, developers and system administrators. Whilst direct access to CourseMarker server commands can still occur through a command-line interface, most users will never

see this in their use of the system. CourseMarker clients have a GUI interface for use by students [FHH+01] and a web interface has also been developed. A CourseMarker web interface has also been developed for the use of system administrators [FHS+01].

3.3.2.5 CourseMarker's Marking Tools and the Generic Marking System

CourseMarker was created concurrently with the diagramming subsystem *DATsys*, described in section 3.3.3, and so the marking system was developed with the full knowledge that a potentially very large number of domains would need to be marked. Tsintsifas states of the system [Ta02] that: "*Devising a prototype mechanism that allows experimentation and creation of novel automatically assessable and across domains diagram CBA is an important deliverable. By using this, metric research for the evaluation of diagram-based coursework could be realistically tested in the context of the classroom.*" With CourseMarker many more domains would have to be marked than with Ceilidh and the potential for the marking of new domains to become a semi-regular occurrence exists if diagram-based CBA becomes widely used. Therefore a marking mechanism had to be designed which would be both extensible and expressive. Furthermore, the marking mechanism had to enable the creation of detailed feedback since this was a perceived weakness of Ceilidh. It was therefore necessary that marking must be more flexible and generic than Ceilidh and able to be configured to mark a large number of domains [Ta02]. Integration of external tools was key to the marking success of Ceilidh and had to be supported here too.

The design of the marking mechanism was based upon Ceilidh's system of marking tools. A Marking Scheme is used to describe the marking of an exercise, calling upon Marking Commands, which in turn use Marking Tools to mark aspects of the solution, return marks and generate a Marking Result which contains feedback to the user which is richer than that returned by Ceilidh. Marking Tool Configurations, which are exercise-specific, exist to specialise the marking tool to the requirements of the exercise. A full conceptual overview of each of these components is provides by [Ta02]. The appearance of rich feedback, based upon the Marking Result, was completed by the use of a GUI representation within the student CourseMarker client [FHS+01], as illustrated in figure 3.6.

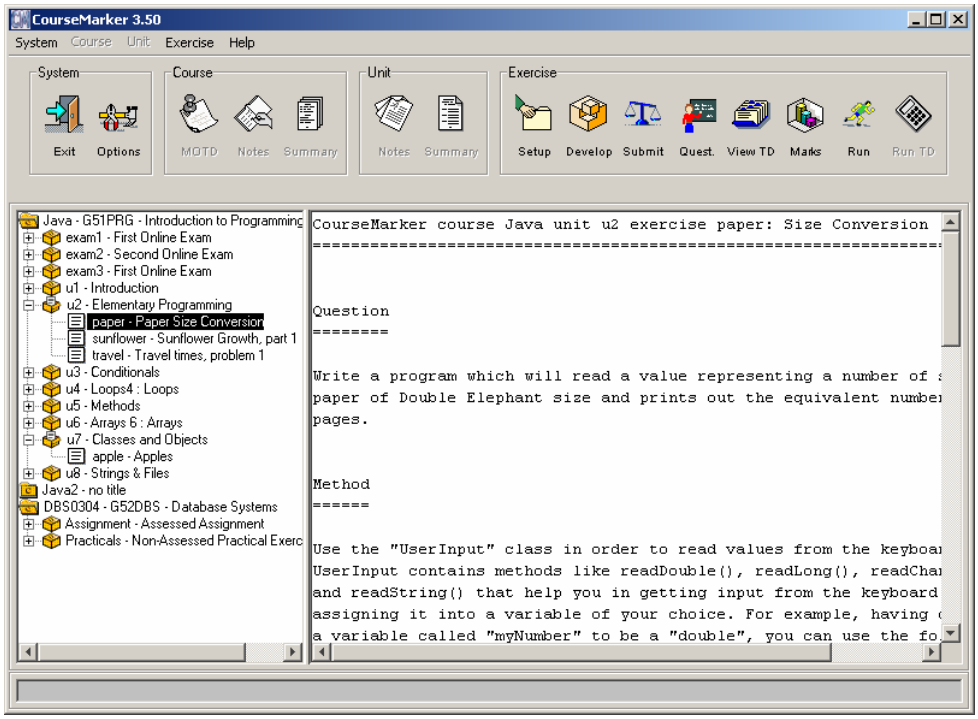

Figure 3.6: The Java CourseMarker client [Sp06]

3.3.2.6 Experiences with CourseMarker

CourseMarker, which is commercially distributed, has been purchased by more than 15 institutions and used in classes of up to 1500 students [Ta02]. This section will consider experiences with CourseMarker at the University of Nottingham, its development base, before considering comments made by those from other institutions.

CourseMarker was first used at Nottingham as a replacement for Ceilidh during the academic year 1998-99. Tsintsifas [Ta02] provides a general evaluation of CourseMarker in March 2002 including an examination of technical improvements over Ceilidh. CourseMarker's primary use at the University of Nottingham is with the assessment of first year courses in Java programming; to this end two courses have been created with exercises regularly updated year-on-year. Courses assessed by CourseMarker involved 150 students at the University in 1998-99; this had risen to 310 students in 2001-02. During 1998-99 (the transition year) some students had used both CourseMarker and Ceilidh depending upon their year of entry and their selected courses. These students were asked to compare their experiences of the two systems, and resoundingly preferred CourseMarker to Ceilidh, mainly due to its

Graphical User Interface and its expanded range of feedback [FHH+01]. Tsintsifas reports on students who were asked simply to evaluate CourseMarker (rather than to compare it with Ceilidh), and states that returned questionnaires indicate that students were largely in favour of the system, especially due to the immediate feedback, availability of multiple submissions and the ability to submit the coursework at their own pace within allotted deadlines.

Teachers and administrators view the system favourably. Teachers "*appreciate the fact that they no longer have to mark hundreds of exercise solutions. Because course administration and monitoring are very effective, even less time is spent on these activities*" [Ta02], whilst administrators find that the system is easier to set up and run than Ceilidh, especially when use is made of the administrator's web interface.

Outside Nottingham, the majority of sites are old Ceilidh users who made the transition to CourseMarker. Tsintsifas [Ta02] quotes positive reports from academic users in Singapore and Glamorgan, UK, illustrating that CourseMarker is viewed as a success outside Nottingham. An evaluation of the usefulness of CourseMarker when compared to other automatic assessment methods is also made by Foster [Fj01]. Foster argues that the benefits of the system are strong; especially praised, once again, is the fully automated, fast marking which makes multiple students submissions feasible. Foster further states that the price of CourseMarker is cheap given the amount of marking time saved. Foster's reservations about the system include: that novel solutions may be penalised by the system; that extra functionality above the question specification will not be rewarded by the system; that the system documentation is regarded as "patchy" and that, as a commercial product, the system is distributed as an executable only, meaning that inspection of the system source or further modifications cannot be undertaken by those purchasing the system. Foster argues that most of the problems he outlines are unsurprising given CourseMarker's nature as a research project which was later simply distributed commercially, and concludes that "[t]*he fact that we are still using CourseMaster, and will continue to do so, is a tribute to the considerable benefit that it does have*."

To summarise, therefore, CourseMarker has been successfully introduced at the University of Nottingham and at a number of external institutions. The amount of workload it carries has increased year-on-year at Nottingham, and reports on the

system by students (both those who had previously used Ceilidh and those who had not) are generally of a positive nature. Teachers and administrators find the system easier to use than Ceilidh, and the system has considerable advantages in terms of marking time saved.

3.3.3 DATsys

DATsys was developed as the main deliverable of Tsintsifas' PhD [Ta02]. Tsintsifas identified the need for diagram-based CBA. CBA applications developed thus far did not address the assessment of diagram-based domains, while existing diagramming packages had not been designed with CBA in mind. This section will first consider the requirements for diagram-based CBA which Tsintsifas identified and which formed the basis for his approach before looking at the main deliverables produced. These deliverables include the *DATsys* framework for diagram-based CBA, the *Daidalos* environment for authoring diagram notations, the *Ariadne* environment for exercise authoring and the *Theseus* customisable student diagram editor [Bb03]. The final deliverables, the Generic Marking Mechanism, has already been examined in section 3.3.2.5. This is an indication of a key point in the development of DATsys, namely that DATsys was not conceived as an addition to CourseMarker to be simply "bolted-on" afterwards; instead, CourseMarker and DATsys were developed in co-ordination with each other. As a result, the need for a Generic Marking Mechanism for the successful CBA of diagrams was identified and implemented in CourseMarker from the beginning and is now used in the assessment of CourseMarker's other CBA domains (primarily programming exercises) as well as for diagram-based CBA. CourseMarker and DATsys are indelibly interlinked, and no true understanding of the one can be achieved without an appreciation of the context of the other.

Tsintsifas identified three major requirements to solve the problem of developing useful diagram-based CBA. These were: the ability to author the editor used by the student to develop solutions in an exercise-specific way during the authoring of the exercise; development of a Generic Marking Mechanism which can be suitably customised to enable the marking of a wide range of diagram types and the integration of a system which meets the previous two requirements into a system which can support the full-lifecycle of Computer Based Assessment. The Generic

Marking System and CourseMarker CBA system, both described in section 3.3.2, were developed to fulfil the second and third requirements respectively.

DATsys was created as an object oriented framework, defined by Gamma et al as "*a set of cooperating classes that make up a reusable design for a specific class of software*" [GHJ+94]. Daidalos, Ariadne and Theseus, examined in the next three subsections respectively, are implemented as concrete subclasses which enable the functionality of the DATsys framework to be used in a CBA context.

3.3.3.1 Daidalos

Daidalos allows the authoring of specifications for diagram notations. It is therefore used by developers to author diagram domains before they can be assessed. Daidalos defines tools for the creation of figures, diagram elements, tools and commands, as well as a selection editor which allows domain libraries of diagram notations to be managed. Tsintsifas argues that Daidalos "*could be considered a meta-diagrammer, as it provides a graphical process for making parts of new diagram editors*" [Ta02].

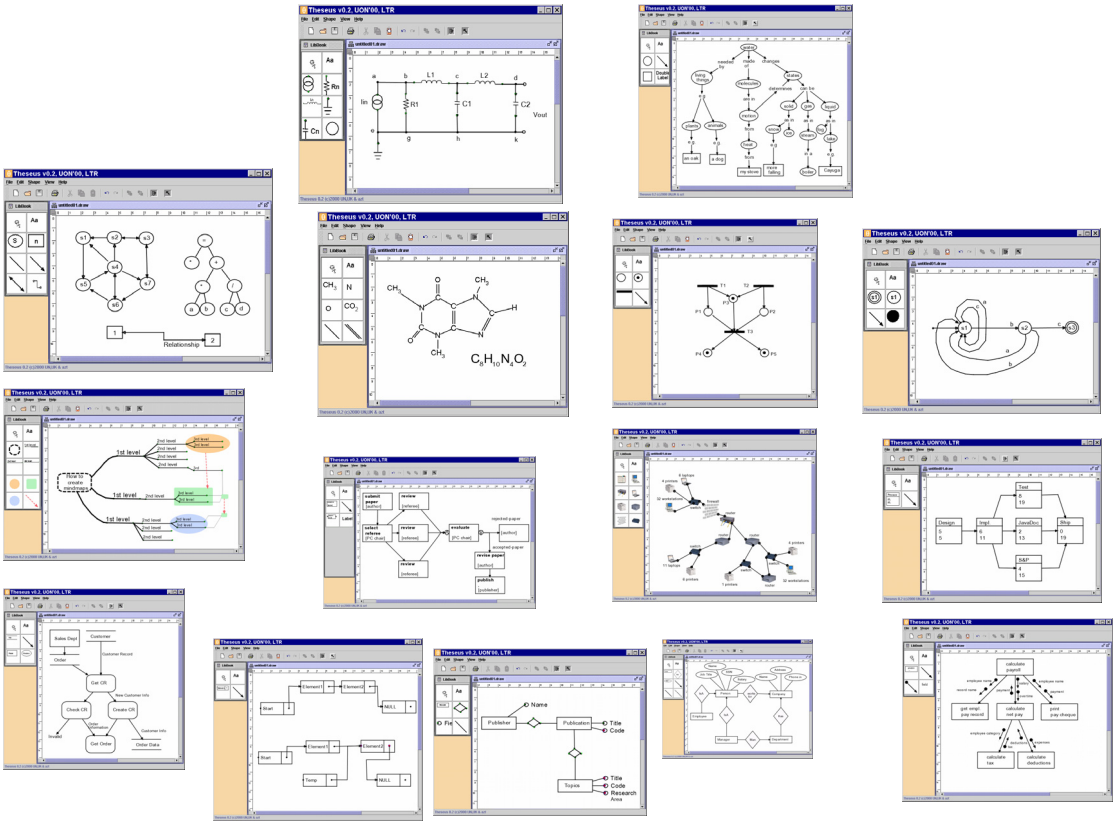

Figure 3.7: A range of diagram notations expressed within DATsys

Developers using Daidalos to author diagram domain notations can define diagram elements (in terms of their graphical view, underlying data model and connectivity constraints), tools and their interaction with the diagram elements, and menu options and the commands they execute. Developers create libraries of tools which are stored in diagram library files (with a `.dlib` extension). Library management functions allow library files to be arranged into groups, with each group representing a diagram domain. A tools graphical view is created by the grouping of graphical primitives on the drawing canvas. The data model for the tool is specified by the addition of typed data fields. The associated connectivity constraints are specified by choosing either perimeter-based connections or pin-based connections (which can themselves be further specialised through specification of connection lines). In this way the representation of a diagram domain is constructed interactively and in an intuitive, graphically-based environment. Exercises within the domain can then be developed within the Ariadne exercise authoring environment. Daidalos is effectively a standalone application with no integration with CourseMarker; it can be used to author a wide variety of diagram notations as evidenced by figure 3.7.

3.3.3.2 Ariadne

Ariadne allows the authoring of specifications for CBA exercises within a diagram domain previously specified within Daidalos. Within Ariadne the student diagram editor, exercise properties and marking scheme can all be specified. Ariadne loads a default group of tool libraries and the existing exercises. If these exercises have already been deployed within CourseMarker then they will be loaded from the course area.

Teachers use Ariadne for the specification of exercises. The student diagram editor is specified in terms of its available tools (taken from tool library files) and available options. Authors select the correct tools from the tool libraries. The marking scheme and marking tool configuration can be edited within text-based windows and configuration for the exercise can be specified. It is possible to draw a model solution upon Ariadne's drawing canvas. Again, therefore, exercises can be developed within an interactive, intuitive graphical environment. Student solutions are then entered online using the configured version of Theseus and then marked according by CourseMarker according to the specification defined by the teacher within Ariadne.

3.3.3.3 Theseus

Theseus is the configurable student diagram editor within which students develop their solution before submission. All of Theseus' features, including the tools available and available options, are defined through configuration. Theseus relies for its configuration upon three configuration files: the first provides the exercise-specific tool library, the second provides the tools to be placed on the toolbar and the third provides configuration for Theseus' execution parameters and working paths.

The tool library, as developed within Daidalos, contains all the tools available to solve the exercise. The students thus place tools on the canvas and attempt to connect them, interacting with the tools, diagram elements, menu options and canvas in the process. Upon completion, the students save the solution as a diagram file (with a `.draw` extension). The students then submit their solution through CourseMarker, which is responsible for marking the solution and returning appropriate feedback.

3.3.3.4 Integration of DATsys with CBA courseware

The level of integration of DATsys with CourseMarker differs across the different sections of DATsys. The Generic Marking Mechanism is fully a part of CourseMarker and is used for the marking of all exercise types, even those with no diagrammatic content. Conversely, Daidalos has been designed to operate completely independently of CourseMarker. Ariadne should have access to CourseMarker's 'CourseArea' directories, where exercise configuration files are stored. Ariadne has knowledge of the files that describe this exercise configuration for exercises in diagram-based domains.

CourseMarker's exercises have a designated 'type' within their configuration, each type being associated, within CourseMarker, with an editor suitable to the exercise domain. Diagramming exercise types have Theseus as their registered editor within CourseMarker; consequently, when students elect to develop their solutions from within CourseMarker the configured Theseus student diagram editor for the exercise is loaded, using the configuration files generated by Ariadne.

When a student solution is saved within Theseus details of all diagram details are saved. These details can be processed by the diagrammatic marking tools which are accessible to the Generic Marking Mechanism. Traversing, translating, converting

and understanding the diagram can all be achieved by these tools. Translating the structure of the diagram involves associating identifiers with each of the nodes within the diagram and the relationships between them. Thus the contents of the full diagram objects are available to the marking tools.

3.3.3.5 Experiences with DATsys

Diagram-based CBA exercises were developed in three domains: logic design, flowchart design and object-oriented design. These exercises were deployed in 1999 to a group of 167 first year computer science undergraduates as part of a module on "Software Tools". Experience has shown that the process of developing exercises is relatively lengthy but straightforward. Tsintsifas describes the complete process of developing the exercises in each of these domains in his thesis [Ta02]. These exercises proved popular with students since the development environment was intuitive and they had been given a brief demonstration of Theseus in a lecture prior to attempting the exercises. The most complex of the exercises within the object oriented design diagram-domain had the unexpected side-effect of causing some students to draw pen-and-paper solutions before transferring this solution online, but in most cases students worked out their solutions and entered them directly into Theseus. Performance at both the client and server level was good; the server was seen to mark up to 15 submissions simultaneously. As a result of this live experience Tsintsifas was able to conclude that diagram-based CBA was both feasible and useful.

Unfortunately, this is the only documented example of diagram-based CBA being used in a live environment prior to this work. Despite the number of institutions which have taken CourseMarker (and previously Ceilidh), none outside Nottingham have yet used diagram-based CBA. It is therefore hoped that the implementation of formative diagram-based CBA as a result of this work will stimulate a wider usage. A key deliverable of this work, indeed, will be to produce examples of working formative diagram-based CBA and test their use in a live environment.

3.3.4 Summary

Ceilidh was an influential CBA system which was responsible for several key innovations. The first was its multi-layer architecture, separating the client layer and

database of course information and submissions away from the architecture of the system and hence allowing multiple courses in disparate domains to be housed simultaneously in a hierarchical structure. The second was its use of marking tools to allow marking of different domains to be specified on a per-domain basis. Ceilidh was widely used and liked, but became difficult to maintain and was eventually superseded by CourseMarker. CourseMarker offers a stable, reliable and secure platform on which to provide CBA courses across multiple domains. Its user interface is attractive and intuitive to students and the system is secure. CourseMarker's feedback mechanism allows feedback comments to be specified for each test conducted in the automated assessment. The DATsys framework for diagram editors in a CBA context provides a flexible platform for the authoring of new diagram notations, the authoring of exercises across multiple domains and the presentation of a configurable development environment for students. Prior to this work, exercises had been developed in three domains but usage had not been extensively reviewed or tested.

3.4 Summary

This chapter provided an overview of CBA systems used for formative assessment and CBA systems used to assess diagram-based domains. Most CBA systems are fixed-response. Section 3.1 first examined a cross-section of fixed response CBA examples, both those built upon established platforms and those built "in-house" from scratch and concluded that the design of the feedback provided to the student is at least as important as the platform upon which the assessment is based. Systems built upon the same platform (e.g. *QuestionMark*) were found to exhibit variation in the quality of formative feedback they provided. Linking to reference materials in feedback was found to motivate subsequent student research. Student motivation could be encouraged further by the use of a two-part assessment strategy, allowing re-submissions and ensuring easy student access to the CBA system. A cross-section of free-response CBA was then examined. A trade-off was noted between fully automating the assessment process or allowing human marker input into stages of the process. Fully automated assessment allows multiple submissions, the fast provision of student feedback and formative assessment opportunities at the expense of academic "prescriptiveness", whereas systems involving human intervention are able to cope better with novel solutions at the expense of speed and formative

assessment opportunity in a test-feedback-retest situation. Section 3.2 examined CBA systems based around diagrammatic domains. All of the systems considered focused on a specialist domain or set of domains. Two of the systems restricted student interaction to the point where their status as free-response CBA systems was debateable. Two of the systems allowed free-responses by students. Section 3.3 began by providing an overview of the historically important CBA system Ceilidh, documenting the continuing advantages and influence of its multi-tier architecture and devolved marking tools. Chapter 3 then provided an in-depth summary of the CourseMarker CBA system and DATsys framework for diagram editors in a CBA context, which will be used as the basis of this work. Chapter 4 documents an attempt to conduct formative assessment using CourseMarker / DATsys, specifically with the intention of identifying the limitations of existing CBA techniques in relation to formative assessment.

Chapter 4

Problems in CBA applied to free-response formative assessment

Introduction

This chapter presents the initial practical research experiment conducted in order to identify those aspects of existing CBA techniques which would need to be extended or adapted in order to meet the criteria of good formative assessment. Section 3.3 described the CourseMarker and DATsys systems. DATsys is a flexible, object-oriented framework for CBA-related diagram editors. CourseMarker is a reliable platform for conducting CBA across a variety of domains. Previous courses assessed using CourseMarker were generally for summative assessment purposes, or else had a dual purpose. This research aimed to conduct formative assessment using CourseMarker / DATsys as a model CBA system. Conclusions could be drawn in terms of positive and negative experiences. The drawbacks would be used to identify those aspects of existing CBA techniques which need to be extended or adapted in order to meet the criteria of good formative assessment, as outlined in Chapter 2.

Coursework involving the construction of entity-relationship diagrams was assessed using CourseMarker / DATsys as part of an undergraduate module in Database Systems and a new marking tool for assessing entity-relationship diagrams within CourseMarker was developed [HB06]. This work aims to develop a framework of best practice for formative assessment across a variety of diagram-based domains rather than be restricted to domain-specific instances. Therefore, an attempt was made to keep the tools constructed as generic as possible to maximise inter-domain potential. Entity-relationship diagrams were assumed to be part of that large section of educational diagrams which are constructed from nodes and the links between them, as per the definition in section 2.3.1. These are the domains which Thomas et al [TWS05] have labelled the "network-like domains". Marking, therefore, consisted of features testing in which the nodes and links of the student diagram were assessed according to features criteria defined by a domain expert. Results were collected and conclusions drawn.

4.1 Assessment Background

Since the experiment aimed to implement a positive formative assessment experience using CBA techniques, it is evident that the assessment should constitute a Computer Based Assessment strategy and adhere to formative assessment practice.

When defining CBA in relation to other areas of learning technology in section 2.1.1, it was emphasised that CBA is the most specialised of the areas considered. The full lifecycle of a CBA exercise includes stages such as authoring the exercise, presenting it to the student, accepting submissions, returning marks and managing the data generated by the system [Ta02]. Since a true CBA system should be committed to automating the entirety of this lifecycle, it is clear that a CBA system is, by definition, non-trivial. CourseMarker and DATsys, described in section 3.3, were developed to manage the full lifecycle of Computer Based Assessment and to allow the CBA of diagram-based domains. Section 2.2.5 outlined a strategy for good formative feedback. It is against these criteria that the CBA assessment experiment feedback and student experience will be measured.

The assessment was conducted as part of a compulsory course in Database Systems taken by second year Computer Science undergraduates at the University of Nottingham. The system attempted to formatively assess student Entity-Relationship diagrams as part of coursework in which students developed their diagrams at an early stage, before moving on to successive tasks involving the construction of SQL query statements (which were further assessed by CourseMarker using methods unrelated to this work).

The coursework constituted a two-part assessment. As defined in section 2.2.3, this means formative assessment with a linked summative element added at the final stage to act as a motivator. An initial problem was presented under purely formative conditions. The students were allowed an unlimited number of submissions and were provided with unlimited help from lab assistants in weekly lab sessions. A second problem was then presented to the students with a summative element: although unlimited submissions to CourseMarker were still allowed, help from lab assistants was limited and students were expected to copy the final diagram into a pre-designated submission form for final, summative marking. This structure was agreed with, and influenced by, the module lecturer. The question texts were also developed by the module lecturer. This ensured, firstly, that the exercises were useful since they had been set by a subject specialist and, secondly, that they did not unconsciously play to the strengths of the system whilst hiding weaknesses.

For this experiment, informal student questionnaires were distributed and tutor observations noted.

4.2 Assessment Construction and Methodology

4.2.1 Assessment Construction

As described in section 3.3.2.5, in CourseMarker a Marking Scheme is used to describe the marking of an exercise. This calls upon Marking Commands, which in turn use Marking Tools to mark aspects of the solution, return marks and generate a Marking Result. For the marking of student submissions a new Marking Command, the `EntityRelationshipCMD`, was created together with a new Marking Tool, the `EntityRelationshipTool`. The approach to marking was based upon an assessment of diagram features, in which the types of nodes and their connections were assessed against criteria provided by the exercise developer. Tsintsifas [Ta02] had developed a similar system for the marking of his trial OO diagramming course; however the approach here was considerably extended to allow the student increased flexibility.

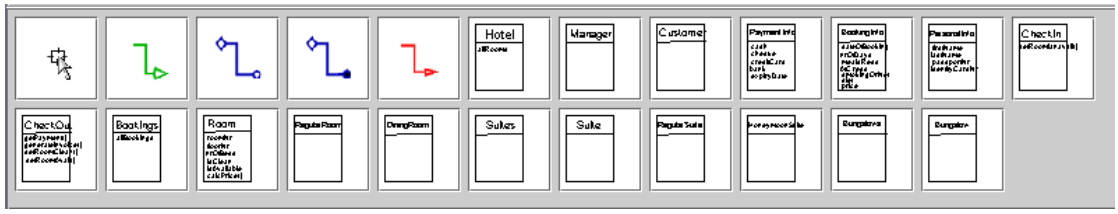

Figure 4.1: Uneditable nodes and distracters in Tsintsifas' OO exercise

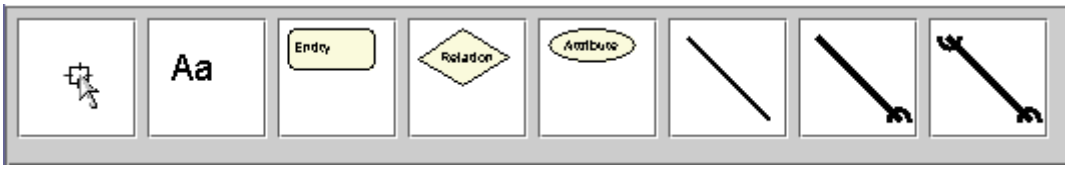

Figure 4.2: Generic nodes in the E-R exercises with editable text

In Tsintsifas' course all possible diagram elements were provided as complete, uneditable entities with incorrect entities also included as distracters, as shown in Figure 4.1. It was felt that in the context of the Entity Relationship diagrams such a method, even with distracters, would serve to provide the students with too much help, especially in light of the fact that an initial problem for the students in the set

coursework was to correctly identify Entities, Attributes and Relationships from the problem description.

Instead, the students were provided with generic diagram elements for Entities, Relationships and Attributes, together with a tool to edit the text within each element on the canvas to the string of their choice. The tool library for these exercises was constructed in Daidalos, and is illustrated in figure 4.2.

Within DATsys, a figure can be composed of any number of primitives (such as lines) together with any number of figures, recursively. Each figure has an attribute `Name` which can be used to distinguish it from other figures. In figure 4.2, from the left, are the standard pointer used to highlight elements on the drawing canvas, followed by figures representing the text tool, Entity figures, Relationship figures, Attribute figures, one-to-one connection lines, one-to-many connection lines and many-to-many connection lines. This tool library was constructed within Daidalos and each type of figure was given a different `Name`. The latter three figures were each defined as connection lines.

The `EntityRelationshipTool` worked from the assumption that each diagram node was a composite figure whose members included a text field called `TextElement`; indeed, this would always be true since the original elements were authored this way in Daidalos. Each node could, therefore be identified in terms of two attributes:

- *Name*: the name of the node, for example *Relation*, *Entity* or *Attribute*;
- *Text Content*: the contents of the editable `TextElement`.

Hence an Entity containing the text "Artist" could be distinguished from both an Entity containing "Album" and an Attribute containing "Artist".

Connection lines, by contrast, were identifiable in terms of three attributes:

- *Name*: the name of the connection, for example *Onetoone*;
- *Start node*: the node connected to the start of the connection line;
- *End node*: the node connected to the end of the connection line.

The procedure for connecting two nodes on the canvas is simple. First, a node is selected from the library and positioned on the canvas with a single click. The text can be edited using the Text tool and the node can be further repositioned by highlighting and dragging. This procedure is repeated for the second node. A connection line is drawn by selecting the line type in the library, and then "dragging" the line between the two nodes by depressing the mouse button on the first node, moving the cursor to the second node and releasing the button. It is important to note in this context that the start and end nodes for each connection line are clearly defined and each connection line is effectively directional for marking purposes.

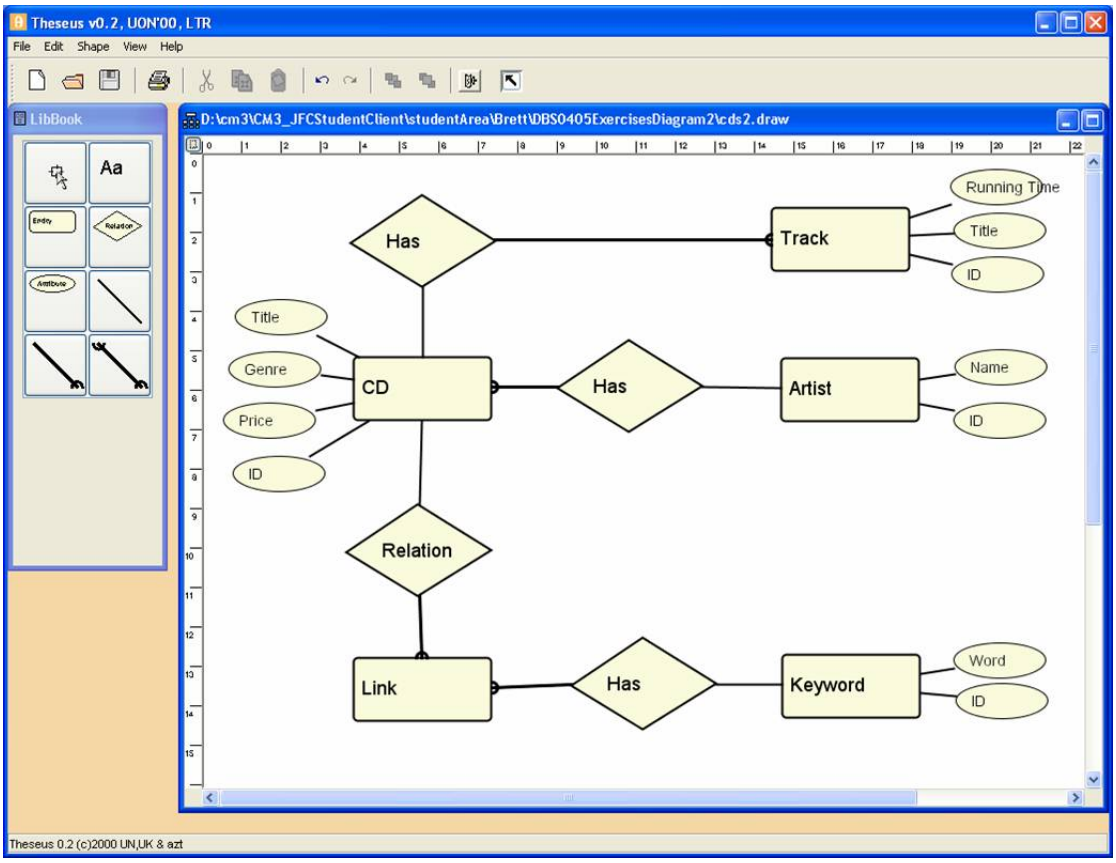

Figure 4.3: An illustrative student ER diagram solution

Potentially, multiple strings could constitute an acceptable Text Content in a student node: an entity may be deemed to be acceptable, for instance, if it contained any of the strings "Artist", "artist", "Artiste" or "artiste". Therefore, the `EntityRelationshipTool` allowed the mark scheme to specify desired text in terms of Oracles [ZF92], an extended notation based upon regular expressions which had already been used successfully in the assessment of programming coursework. A

further keyword, "owt", was introduced to indicate that *any* text would be accepted in a given instance.

A key aim of the design of the tool library was for the drawing of the solution to be intuitive to the student. For this reason the unadorned, straight line connector which officially represented one-to-one connections was also allowed to represent the connection from an Entity to an Attribute, since the two connections are both represented by a simple, straight line. An illustrative example of a student solution is shown in figure 4.3.

The marking tool was invoked by a customised marking scheme (expressed in Java as described by Symeonidis [Sp02]). Both the submitted student diagram and the features specification file, `mark.er`, were passed to the command by the marking scheme. Within the specification file each line represented an individual features test. Each individual features test is represented as follows:

- `Mark weight : Feature expression : Description : Positive feedback : Negative feedback`

The mark weight is an integer denoting the *relative* significance of the test. The description is a string containing a description of the test which is available to the student at the point of feedback. The student receives either the positive feedback or the negative feedback depending on the outcome of the test. Feature expressions are the most complex component of the features test and may take one of the formats described in Table 4.1.

With the exception of *compositeRelationship*, these feature expressions are generic. To author other compatible domain notations in Daidalos one would simply ensure that each diagram element type has a unique name, and that all text which is to be editable by the student is contained within an embedded TextElement. The *compositeRelationship* expression is domain specific because it has knowledge, firstly, of which connection lines across a Relationship element constitute a complete relationship and, secondly, the types of the relationships. Further parameterisation of feature expressions is possible to indicate the *number* of matches within the diagram as illustrated in the examples. If no such parameter is provided then the default is to declare success if one or more matches are found.

Expression	*Format*
exist	`exist [Name]` *Checks that an element with a given Name exists.* **Examples** `exist Relationship` *Checks that at least one relationship exists.* `exist onetoone` *Checks that at least one onetoone connection exists.*
exact	`exact [Name] [Text]` *Checks for an element with given Name and Text Content.* **Example** `exact Entity CD` *Checks for at least one Entity with Text Content "CD".*
connection	`connection [ConnectionLine] [ElementType1] [ElementType2]` *Checks that a direction-specific Connection Line exists from a node with Name Name1 to a node with Name Name2.* **Example** `connection onetoone Entity Relationship` *Checks that at least one onetoone connection line exists from an Entity element to a Relationship element.*
exactConnection	`exactConnection [u|d] [ConnectionLineName] [Name1] [TextContent1] [Name2] [TextContent2]` *Checks for a Connection Line from one element with specified text to another. Direction-specific from element1 to element2 if the parameter is "d"; "u" indicates that the connection may be in either direction.* **Example** `exactConnection u onetoone Entity CD Relationship Produces` *Checks that at least one onetoone connection line joins an Entity displaying the text "CD" to a Relationship displaying the text "Produces".*
composite Relationship	`compositeRelationship [u|d] [RelationshipType] [Name1] [TextContent1] [Relationship] [RelationshipTextContent] [Name2] [TextContent2]` *Checks for a full E-R relationship, with a connection across a relationship between two entities. It does not need the individual ConnectionLines to be specified, just the entire relationship type. If the first parameter is "d" the connection must go from element1 to element2; if "u" it may go either way.* **Example** `compositeRelationship d onetomany Entity CD Relationship Have Entity Track` *Checks for a directional onetomany relationship from an Entity CD, across a Relationship Have, to an Entity Track.*

Table 4.1: Features expressions for the ER exercises

Accepted operators are ==, >, <, <= and >=.

The following are sample features tests from the exercises, of gradually increasing complexity.

Example 1

- `1 : exist Entity : Check for entities in your diagram : Found : You have no entities at all in your diagram! :`

This test checks that a student solution contains at least 1 figure with the Name "Entity". If this is found then 1 mark is awarded and "Found" is given as feedback; if this is not found then 0 marks are awarded and "You have no entities at all in your diagram!" is given as feedback.

Example 2

- `1: exact Entity [Aa]rtist==1 : Check for an Artist entity : Found : NOT found! :`

This test checks that exactly one Entity figure exists whose text is either "Artist" or "artist".

Example 3

- `3: exactConnection u onetoone Entity (CD|cd) Attribute [Pp]rice : Checking an attribute of CD : Found correctly : CD does not possess an essential attribute :`

This test checks if an undirectional one-to-one connection line joins an Entity figure whose text is either "CD" or "cd" with an Attribute figure whose text content is either "Price" or "price". 3 marks are awarded if the desired connection is found.

Example 4

- `5: compositeRelationship d onetomany Entity [Aa]rtist Relationship owt Entity (CD|cd) : Check relationship between Artist and CD : Correct : The type of relationship between Artist and CD is incorrect! :`

This test checks if a directional one-to-many relationship exists which links an Entity figure whose text content is either "Artist" or "artist" with another Entity figure

whose text content is either “CD” or “cd” via a Relationship figure whose text is not examined.

4.2.2 Methodology

In this experiment, a student cohort of 141 undergraduate Computer Science students in their second year was invited to attempt the formative exercises prior to undertaking associated summatively assessed exercises, which were compulsory. The formative exercises were, therefore, the initial stage in a two-part assessment strategy whose purpose was to motivate students. A smaller initial problem set comprised 3 exercises. The first exercise was trivial and designed to allow students to learn to use the system while the other two exercises were progressively more complex. Subsequently, a more substantial problem required students to draw a diagram which would be used as the basis for further questions in the summative assessment.

Data was collected in several ways. Quantitative data was collected using CourseMarker’s Archiving Server and by using Likert scale questions in student surveys. The student solution at *every submission* was captured using CourseMarker’s Archiving Server, together with the associated marks, which were hidden from the student, and the feedback, which was returned to the student. Thus, for each student, it was possible to track changes made between submissions. It was also possible to access information based upon the number of submissions made by each student and how the marks changed as the number of submissions increased.

Likert scale questions within student questionnaires were designed to assess how useful the exercises had been to the student learning process and how enjoyable the experience of using the CBA system had been to the users. Student questionnaires were distributed to students by lab tutors in the final lab session; unfortunately, attendance at this particular weekly session was low.

Qualitative data was collected through the use of open-ended questions at the end of the student surveys and through conducting brief, informal interviews with the lab tutors. The qualitative data was used to explain trends which could be observed in the quantitative data in context. Much of the quantitative data was taken from the marking audits which, by definition, can take into account only those factors which

have been marked. Since a fundamental requirement of the experiment was to determine the success of the automated assessment, it was necessary to consider the observations and experience of the tutors who had led the laboratory sessions.

4.3 Results and Analysis

4.3.1 General Impressions

The two-part assessment strategy successfully ensured a high student motivation: of 141 active students registered on the Database Systems course, 130 (92%) attempted the formative diagramming exercises [HB06]. Although the students themselves were presented with text feedback rather than their percentage scores, the following results provide a good indication of the level of assistance provided by the system. For the smaller initial problem set students made an average of 5 submissions, with first submissions being awarded an average of 49.2% and final submissions an average of 75.1%. For the larger second problem linked to the summative assignment, students made an average of 9 submissions (with 8 students making more than 25 submissions and one student a total of 72!), with initial submissions being awarded an average of 50.7% and final submissions an average of 70.1%.

Completed questionnaires showed that students were pleased by the parameterised Theseus development environment in which they were asked to develop their solutions. Although it was effectively optional, most students chose to directly develop their solutions online — the main exceptions were those most conscientious students who had started to develop their solutions on paper at home as soon as the coursework was announced, and even many of those were persuaded to copy their solutions from paper into Theseus in order to gain feedback. The lab assistants largely corrected most common student misunderstandings quickly; these can be reduced in future by taking care with the wording of questions. Generally, however, students found the instructions clear and the exercises straightforward.

4.3.2 Problems

A major problem from a marking point of view occurred because of the way in which features tests are specified in CourseMarker. Each features test is assessed exactly once and a mark assigned for each submission. Although this had previously seemed adequate for features testing of both programming coursework and the

summative CBA of diagrams by Tsintsifas [Ta02], in the coursework being assessed here there were several equally valid model solutions with slightly differing, mutually exclusive, features. As a result, features tests could only be constructed to search for that subset of features which were common to all model solutions. This scenario is clearly unacceptable. In earlier programming exercises, such cases were rare and were often solved by careful wording of the question specification (or, sometimes, explicit instructions to students).

A second problem was in the lack of marking for diagram appearance. Since the features marking system utilised considered only the diagram elements and the connections between them, it was possible to attain good feedback with a diagram of very poor layout. Indeed, many students took full advantage of this fact, meaning that when unexpected feedback was received it was sometimes difficult for a lab assistant to determine what was wrong with a student diagram due to its poor layout. In fact, the importance of marking diagram appearance had been identified before deployment of the course and was not implemented simply because of time constraints. In the event, experience has confirmed that this is a major issue to be addressed.

The third major problem was in the expressiveness of the feedback. Although considerable effort was undertaken to provide useful feedback for each features test — especially the feedback for the 'negative' case where the student had failed the feature test and assistance was required — it is clear that the feedback did not fully constitute effective formative feedback as defined in section 2.2.5. Specifically, the feedback tended to be too lengthy, since feedback was returned for every features test, and too focused on particular student weaknesses due to its link to a specific features test failure. The feedback will be scrutinised more closely in section 4.3.4.

4.3.3 Marking data

An analysis of results from the course shows that, for the largest exercise, the difference in marks between the earlier submissions is substantially larger than that between later submissions. Figure 4.4 shows how the underlying average student mark improved over the first 9 submissions for those whose total submissions were 12 (the average) or fewer. On average, over the first 9 submissions a gradually improving underlying student average mark converges around the 70% mark.

At this stage the improvement in student marks becomes negligible; this may account for the average number of submissions being 12 since the feedback to the student would have changed little for 2 or 3 consecutive submissions. Since 70% is considered a first-class mark, the feedback to the student was generally positive at the 70% level and so the student would consider their solution adequate.

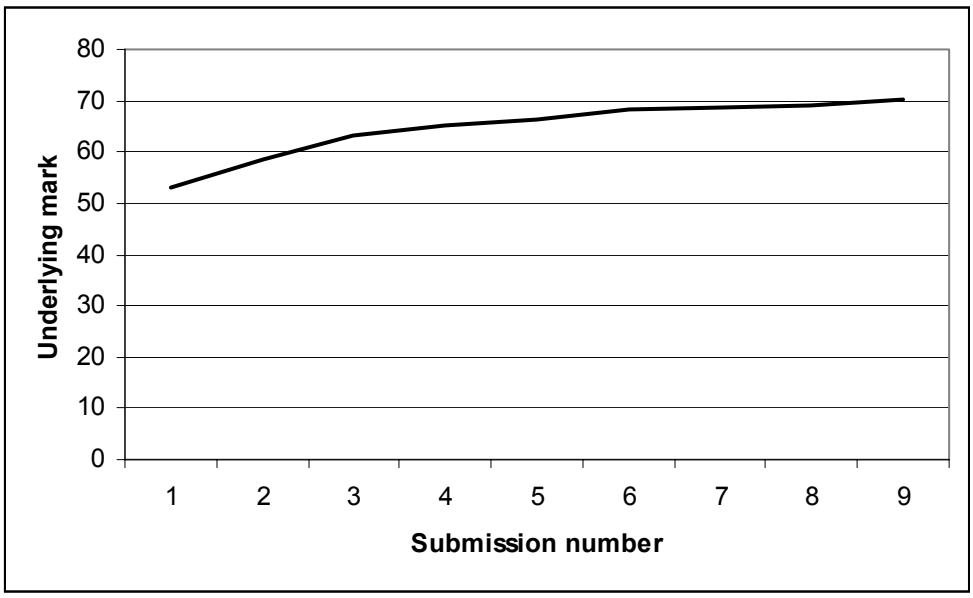

Figure 4.4: First nine submissions of students who submitted 12 times or less

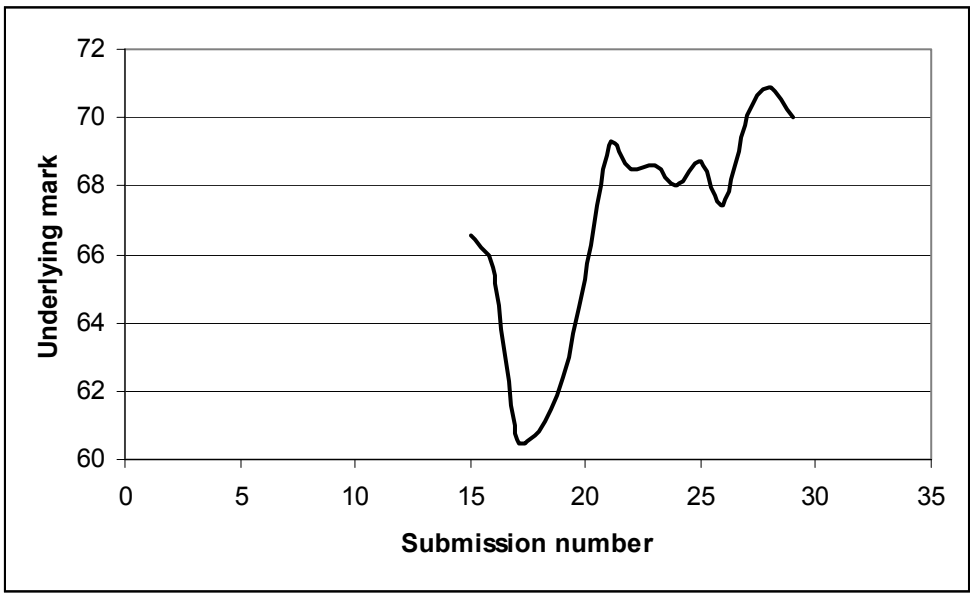

Figure 4.5: Submissions 15 to 30 for those students who submitted more than 12 times

Those students who submitted a great number of times failed to acquire proportionally higher marks. Indeed, those students who submitted greatly more than the average produced a widely fluctuating average mark as shown in figure 4.5. It is likely that these students are the "gamblers and perfectionists" constituency identified by Benford et al [BBF92]. Gamblers randomly submit altered solutions in the hope of chancing on a higher mark and include those students who are interested in the mechanism behind the automated assessment (who were here provided with ample opportunity to experiment). Perfectionists tend to never be satisfied with their feedback and submit more times in the hope of achieving a slightly higher result. It should be noted, however, that the proportion of students falling into these categories in this course was lower than that reported by Benford, and that the nature of this course as a formative assessment rendered such practices academically harmless, if a waste of the student's own time.

4.3.4 Performance as Formative Assessment

The effectiveness of formative assessment can be measured in terms of its primary deliverable: feedback. Section 2.2.5 proposed a good feedback framework for formative assessment and here the system is judged by those criteria.

The interactive CBA system encourages independent learning; although 2 hours of tutor assistance was provided in laboratory sessions weekly, attendance was on a voluntary basis and most students chose to attempt the exercises at times convenient to them as the laboratory was open during daytime hours. The students were free to work independently or could discuss their work with peers, and the feedback provided by the system allowed gradual improvements in the quality of submissions. Tutors were allowed to help the students for the early exercises but such help was discouraged later. Students also found the system itself easy to use and tutor advice was required rarely. As a result, the system fulfilled the objectives of criterion 1 successfully.

Peer dialogue was encouraged in all the formative assessment exercises. A lot of peer-to-peer discussion occurred during lab sessions and likely at other times as well. This has already been documented before in relation to CBA systems in a process which Benford et al [BBF92] labelled *consciousness raising*. CBA encourages debate on the assessment process and students will contest perceived injustice in their feedback

more vociferously than when a human marker was involved. The system therefore fulfilled the objectives of criterion 2 successfully.

The decision to display only feedback to students and to withhold marks caused some initial confusion as to what constituted good performance as some students were used to the convention of grades or percentages. Tutor reassurance overcame this problem, however, and eventually most students came to recognise from the phrasing of the feedback alone that they had submitted a good solution. It is clear, however, that improvement could be achieved in relation to criterion 3 by providing a structured set of feedback examples in the online exercise text alongside a corresponding set of illustrative examples.

Marking is conducted and feedback returned on a timescale which is, in practical terms, instantaneous. Therefore the feedback can be viewed by the student and related to a submission which is still fresh in the mind. Students are allowed unlimited submissions and are therefore provided ample opportunity to act upon their feedback. A student can later choose to consult their feedback for any exercise for which they have a submitted solution. Criteria 4 and 5 are therefore adhered to but the problem of considerable feedback unwieldiness still needs to be addressed, as discussed in section 5.2.

Students were assessed frequently but since this was on a formative basis students were not under pressure to attain high marks immediately. Thus criterion 6 was adhered to. Finally, since CourseMarker has good statistics facilities already in place [FHH+01], student progress was monitored with ease, allowing conclusions to be drawn about future teaching of the material and fulfilling criterion 7.

4.4 Conclusions

The process of conducting formative assessment within diagram-based domains using CBA courseware in this experiment was encouraging, but the adherence to good formative feedback practice and Computer Based Assessment principles was incomplete.

Computer Based Assessment principles were breached because the marking system of CourseMarker was not sufficiently flexible to assess the mutually exclusive

solution cases which arose in the more complex later exercises. This necessitated tutor involvement at the marking stage, since CourseMarker only marked (and therefore could only provide feedback upon) the common subset features.

Formative assessment best practice was not achieved for two distinct reasons. Firstly, the feedback, distributed through a feedback mechanism designed with summative assessment in mind, was lengthy. Many comments provided as feedback related to aspects of the exercise which the student had already successfully completed, while those relating to areas of improvement tended to simply state the student failure in a way which was both too specific and overly negative. It is clear that more targeted and motivational feedback is required for the formative assessment process.

Secondly, the features marking system assessed only the semantics of the diagram. The aesthetic appearance of the diagram was effectively ignored by the system. Section 2.3 outlined the purpose of diagrams: to convey information. Even tutors familiar with the entity-relationship diagram domain often found difficulty in comprehending student diagrams due to poor layout. This fact alone confirms that student diagrams often failed to achieve good practice in this area. The purpose of formative assessment is to aid in the learning process; it is clear that, here, a part of that learning process had been excluded.

4.5 Summary

This chapter described a practical experiment in the automated assessment of entity-relationship diagrams, conducted with the intention of determining the suitability of the CourseMarker / DATsys platform, which section 3.3 had presented as a model CBA architecture, to conduct formative assessment within free-form, diagram-based domains. Overall results were encouraging. The system was popular with students since they appreciated the interactive and intuitive user interface and immediate feedback. Students were provided only with feedback, but the underlying marks increased steadily over multiple submissions, demonstrating student learning.

However, the system demonstrated problems in several key areas. Firstly, the marking scheme was insufficiently expressive to allow for the mutually exclusive solution cases which arose in more complex questions: this resulted in the CBA system being able to mark only common subset features. Secondly, the feedback was

lengthy, insufficiently targeted on the learning process and was not motivational. Thirdly, a key component of learning to draw educational diagrams, i.e. aesthetic considerations, was not addressed by the system with negative consequences in the resultant student diagrams.

Chapter 5 examines the provision of formative CBA within diagram-based domains and outlines the problems which must be overcome in light of the conclusions drawn by the preliminary work in chapter 4.

Chapter 5

Providing a specification for formative CBA in diagram-based domains

Introduction

The process of automating the formative assessment process in free-response diagram-based domains using Computer Based Assessment technology is both feasible and useful. Chapter 4 described an initial research experiment conducted to identify the shortcomings in current CBA techniques, exemplified by the CourseMarker / DATsys system, when related to formative assessment. Proceeding from this point, this chapter identifies those features which must be present for formative, diagram-based CBA to be successful, considers the extensions needed to facilitate that success and outlines a series of specific requirements in each of the identified problem areas. The objective is to argue that CourseMarker / DATsys are a suitable platform for conducting CBA formatively in diagram-based domains and that the system can cater for the full lifecycle of formative CBA if the identified extensions are implemented.

Section 5.1 states the requirements of formative, diagram-based CBA arising from its definition, states the motivation and aims of the work and argues that the requirements can be feasibly achieved by extending the CourseMarker / DATsys CBA system. Current capabilities of the CourseMarker / DATsys system allow it to fulfil some requirements, primarily those shared with summative CBA, while the extensibility of the system makes it a suitable platform for the necessary extensions. The extension requirements are discussed: an extensible system of marking tools to allow the marking of the aesthetic appearance of diagrams; a more flexible features marking system able to consider mutually exclusive alternative solution cases; a system to provide truncated, prioritised feedback and a system of guidance for the construction of formative, free-response exercises, particularly in terms of the creation of feedback.

Section 5.2 outlines measurable requirements which must be fulfilled in order to render successful the extensions in each of the problem areas.

5.1 Objectives

Section 5.1.1 outlines those criteria which must be fulfilled in order to satisfy the definitions of Computer Based Assessment and formative assessment. Section 5.1.2

assesses which of these criteria are already fulfilled by the existing CourseMarker / DATsys system and which criteria necessitate the extension or change of the existing architecture. Section 5.1.3 explains the overall motivation in terms of the resultant questions which must be answered in the context of applying a CBA approach to conducting formative assessment in free-response diagrammatic domains.

5.1.1 Definitions

Figure 1.1 illustrated the scope of this project as the intersection between free-response CBA, formative assessment and diagramming. Section 2.1.1 defined CBA as "*the delivery of materials for teaching and assessment, the input of solutions by the students, an automated assessment process and the delivery of feedback, all achieved through an integrated, coherent online system*". Section 2.2.1 defined formative assessment as "*assessment conducted throughout the learning process, as an integral part of that process, where the central aim is to provide feedback to enable the enhancement of learning*" while section 2.3.1 notes that an educational diagram is a collection of nodes and lines constrained by a convention of meaning whose purpose is to convey information. Section 3.1 noted that a fully automated CBA approach, when compared with other approaches, provides the best potential for time-saving in conducting formative assessment and also allows a realistic test-feedback-retest cycle of iterative learning.

Almond et al [ASM02] summarise the four basic processes present in an assessment cycle. The *Activity Selection Process* selects and sequences tasks with an assessment or instructional focus, including administrative duties. The *Presentation Process* presents the task to the student and captures their response. *Response Processing* identifies and evaluates essential features in the response and records a series of "Observations". Finally, the *Summary Scoring Process* uses the Observations to update the "Scoring Record". Since a CBA approach attempts to automate the entire assessment process it is clear that these processes constitute a *minimum programme* of automation objectives for any CBA system.

As the primary deliverable of the formative assessment process, feedback provides a key to measuring the success of a formative assessment system. Section 2.2.5 outlined a framework for effective feedback within a formative assessment context. Formative assessment should: facilitate the development of self-assessment (reflection) in learning; encourage teacher and peer dialogue around learning; clarify what

constitutes good performance; provide opportunities to improve performance; deliver information focused on student learning; encourage positive motivational beliefs and self-esteem and provide information to educators to shape future teaching.

Based upon the definition of diagrams as a collection of nodes and connection lines, Tsintsifas [Ta02] concluded that a key element of diagram-based CBA was the ability to provide domain coverage while allowing users to manipulate a standard set of tools. Requirements for diagram editors were the level of Human Computer Interaction and the simplicity, intuitiveness and usability of the diagram editors. If a diagram has as its purpose the aim of communicating information, then it is necessary to assess a diagram in terms of two criteria:

- The information provided by the diagram to the recipient must be correct;
- The diagram must be displayed in a way that is aesthetically pleasing, to avoid recipient confusion.

Section 5.1.2 assesses which of these criteria are already fulfilled by the existing CourseMarker / DATsys system and which criteria necessitate the extension or change of the existing architecture.

5.1.2 Identifying the Necessary Extensions

To conduct formative computer-based assessment in diagram-based domains, it is necessary to adhere to requirements and best practice in three areas: CBA, which must fully automate the assessment process in a coherent online system; formative assessment, whose primary purpose is to assist learning and educational diagrams, which constitute a wide variety of node-and-link based domains. This section considers each set of criteria and outlines systematically the shortcomings of CourseMarker / DATsys in fulfilling the requirements.

In chapter 4, the shortcomings of CourseMarker / DATsys were summarised, as the result of a practical experiment, in terms of the ability to encompass mutually exclusive solution cases in marking, the ability to assess the aesthetics of a student diagram, and the provision of concise, prioritised feedback. This section argues that these shortcomings form the core extensions needed to accommodate formative

assessment within diagrammatic domains using CourseMarker / DATsys and are not specific to a specific experiment case. The section also demonstrates why further shortcomings *of a generic nature* would not be identified by further, domain-specific experiments. The approach taken is to consider how CourseMarker / DATsys can fulfil each of the criteria arising from the basic definitions, as outlined in section 5.1.1.

Formative and summative assessment stand opposed in many respects, as outlined in section 2.2. However, the underlying architecture required for the assessment process is similar in many respects; it is this similarity that enhances the feasibility of this project by enabling a system for formative assessment to be built by extending the CourseMarker / DATsys system which was intended for summative assessment purposes. Such similarities were not emphasised by Tsintsifas [Ta02]. Tsintsifas emphasised the practical benefits gained by replacing summative assessment with CBA, without considering that formative assessment was the assessment form most in need of replacement. Tsintsifas believed that security, performance and administration were important only in summative assessment. While student plagiarism is not an issue in formative assessment, it is plain that unauthorised tampering with the system would affect the learning process of others. Furthermore, effective performance and administration are required to provide timely feedback to the students and useful feedback to educators to improve future learning. Tsintsifas further argued that formative assessment could be implemented by discarding the summative marks — disregarding the differences in feedback required, as demonstrated by the experiment outlined in Chapter 4.

5.1.2.1 Fulfilling Computer Based Assessment criteria

To successfully apply CBA technology, it is necessary that the entire assessment process be automated within an integrated system. It is necessary, therefore, to examine the success of CourseMarker in automating the basic assessment process as described in section 5.1.1.

The CourseMarker architecture was described in section 3.3.2. Sequencing of assessment tasks can be specified exactly using a Marking Scheme. Presentation of teaching materials can be achieved through the user clients, for example the Java CourseMarker client illustrated in figure 3.6. The Administrator has a defined role as a User within CourseMarker and administrative tasks are split between the Login

Server, which registers users and validates sessions, the Course Server, which controls module information and sets up exercises, the Submission Server, which receives submissions and issues receipts, the Archiving Server, which maintains audit trails and the Ceilidh server, which manages the other servers and can reload them at runtime. The DATsys architecture, described in section 3.3.3, describes how new domains can be authored with Daidalos and new exercises with Ariadne. The student launches the configured Theseus editor from within the CourseMarker client to draw their solution and then submits through CourseMarker. Hence, the Activity Selection Process, involving the sequencing of tasks and the administrative duties, is well defined in CourseMarker / DATsys.

The Presentation Process in assessment systems presents the problem to the student and then captures the student's response. The student is presented with a problem specification in the CourseMarker client. Upon setting up the exercise, the student can develop their solution within a parameterised Theseus client. Theseus allows the student to interactively "draw" their diagram upon a development canvas. The student can save their drawing by selecting the save function within Theseus. Their drawing is stored in a `.draw` file as a collection of objects, each representing a node or connection, which can later be traversed as an enumeration by the marking tools. Hence the problem is concisely presented to the student and the solution captured in a usable way.

Response Processing examines the features of the solution and catalogues them. Features testing for an exercise is specified on a feature by feature basis. The relative weight of the feature, the definition of the feature being sought and feedback for positive and negative results, are specified line by line. The manner of features specification has survived since Ceilidh with little change. In programming exercises, feature definitions are regular expressions which match the feature being sought. In diagrammatic exercises the feature being sought is defined as in section 4.2. One major problem, identified as a result of the experiment summarised in Chapter 4, is that the features are assumed to be mutually supportive. Marking is therefore accumulated across all features. Previously, in programming exercises using Ceilidh and CourseMarker, students have been "shepherded" into using one particular feature over an alternative option through careful question wording or the threat of being awarded 0 marks overall if a given token is identified. Within formative,

diagram-based exercises, this situation is unsatisfactory. Therefore, it is necessary to enable the marking system to consider mutually exclusive solution cases before cataloguing for the purposes of response.

The Summary Scoring Process uses the Observations recorded by Response Processing to update the score of the exercise. CourseMarker assigns marks based upon a weighted summary of the tests it has carried out, and stores these marks in a structured Marking Result. For the purposes of general assessment, the CourseMarker marking system is logical and efficient. Therefore, the Summary Scoring Process is successfully automated within CourseMarker.

5.1.2.2 Fulfilling Formative Assessment criteria

To successfully facilitate formative assessment, it is necessary to examine how the feedback system of CourseMarker conforms to the framework for formative feedback originally summarised in section 2.2.5.

The feedback mechanism of CourseMarker facilitates student reflection in learning because the feedback process is automated. Students can learn at their own pace and submit at their own convenience since the CourseMarker client is widely accessible to students. Previous courses in programming and diagram-based domains, including the entity-relationship course described in Chapter 4, adopted an approach whereby students were free to choose to attend weekly laboratory sessions, where help from tutors was available, or to attempt the exercises on their own. Experience has shown that tutor advice is required rarely and on peripheral issues; there is no evidence to demonstrate that a course involving the student working through a set of exercises at their own pace without tutor input would be impractical, provided help could be provided (perhaps in the form of a technical support email address) for unforeseen technical issues or comments. Thus, the first criterion outlined in section 2.2.5 can be fulfilled using CourseMarker's present capabilities.

CourseMarker encourages teacher and peer dialogue around learning. Section 2.1.4 outlined student willingness to question CBA marking results. Students are likely to collaborate if working in unsupervised laboratory sessions; if the assessment being conducted is formative, then this can be viewed as a helpful part of the learning process. Section 3.3.1.5 outlined the observation that Ceilidh, the forerunner to

CourseMarker, markedly increased student consciousness of the learning and assessment process. Therefore, the second criterion outlined in section 2.2.5 can be fulfilled using CourseMarker's present capabilities.

The practical experiment in Chapter 4 highlighted problems in clarifying to the student what constitutes good performance. In previous summative assessment, CourseMarker provided feedback as an expandable tree of grades. For formative assessment purposes the grades were removed, but this caused confusion among students as to how much improvement their solution required. To improve this situation will require presenting the questions, solutions and feedback in a clear way to the student. Section 3.1 highlighted that the design of the assessment problem and feedback is as important to the success of the formative assessment as the technical capability of the system used. Therefore, it will be necessary to produce a set of guidelines for exercise developers and teachers to promote good practice in this area.

CourseMarker provides opportunities to improve performance through allowing multiple submissions and providing feedback quickly to students. The fact that the assessment process is entirely automated is key to CourseMarker's innate ability in this area. Therefore, the fourth criterion outlined in section 2.2.5 can be fulfilled using CourseMarker's present capabilities.

CourseMarker fails to deliver information which is sufficiently focused on student learning. CourseMarker's feedback is an exhaustive summary of the feedback comments from each test undertaken during the assessment process, which is often overwhelming in quantity and discourages the student from viewing the exercise as a holistic entity. It is clear that the feedback mechanism of CourseMarker must be modified to provide a smaller number of prioritised comments focused on the student learning process in a motivational way.

Formative assessment using CourseMarker can encourage positive motivational beliefs and self-esteem through providing frequent low-stakes assessment. For this to be successful the feedback itself must be phrased by the exercise developer in a motivational way; this is a focus of the guidelines discussed above. Furthermore, the feedback must be short and prioritised, as discussed within the context of providing feedback focused on student learning.

Lastly, feedback should provide information to educators. This requirement is met fully by the existing CourseMarker system, which allows all submissions by all students, together with associated marks and feedback, to be examined by educators.

5.1.2.3 Fulfilling Computer Based Assessment criteria

Representations for new diagram domains can be authored, for use in DATsys diagram editors, using the Daidalos authoring environment. Section 2.3 explained that educational diagrams are commonly a collection of nodes and links. Nodes and links, and the connectivity between them, can be specified in Daidalos on a domain specific basis. Furthermore, the Generic Marking Mechanism is extendable and designed to accommodate the features marking of new exercise domains, which includes new diagram-based domains.

However, the effectiveness of a diagram in conveying meaning is affected by its aesthetic appearance. A diagram whose physical layout is confusing to the reader is poorer at conveying information than a diagram with identical nodes and connections but a less confusing layout. The assessment of diagram aesthetics in a CBA context is not catered for by CourseMarker / DATsys; indeed, it is undocumented in the literature. This is, therefore, a requirement for formative assessment in this field.

5.1.2.4 Summary

Section 5.1.2 outlined the requirements for CBA, formative assessment and educational diagrams based upon the definitions provided in section 5.1.1. CourseMarker / DATsys constitute a platform which is already able to cater for many of the outlined requirements; it was argued that this was because of the overlap between formative and summative assessment requirements in CBA terms. Outstanding requirements which must be met in order to demonstrate that formative CBA within diagram-based domains can be useful and feasible include the three requirements identified as a result of practical experiments:

- The ability to distinguish between student diagrams of differing aesthetic appearance;

- The ability to consider solutions in which multiple, mutually exclusive cases may be acceptable;

- The ability to truncate the feedback provided to students to reduce confusion — students must be provided with the most relevant comments to their solution.

Furthermore, to accommodate formative assessment criteria it is necessary to provide a set of brief guidelines to educators to assist in *presenting* materials within a free-response CBA context. These guidelines should outline methods for creating positive, motivational feedback and for clearly specifying good practice to students in the specification text.

5.1.3 Aims and Motivation

The aim of this work is to demonstrate that the automation of the formative assessment of diagram-based coursework using CBA courseware is both feasible and useful.

Section 2.2 outlined the numerous pedagogical benefits of formative assessment. Formative assessment encourages openness among students, can be used to assess a great scope of learning outcomes, can help in avoiding mark aggregation and discourages plagiarism. Despite this, formative assessment is in a usage decline because it is seen as *resource intensive*. Section 2.1 pointed out that CBA approaches routinely demonstrate great *resource-savings* whilst free-response domains, such as diagrams, offer the great scope for assessing a wide range of cognitive learning levels. The most basic motivation of this work, therefore, is to answer the question:

- To what extent can CBA techniques be used to reduce the resource required in setting a formatively assessed coursework in a diagram-based domain, marking student submissions and returning feedback, while still adhering to good formative assessment principles?

This question could be alternately phrased thus:

- To what extent would current, successful CBA practices need to be changed to conform to formal formative assessment guidelines?

To answer these questions this work used an initial phase of research to identify shortcomings in the existing courseware. A practical experiment and a consideration of the requirements implied by definition highlighted that three extensions to the courseware must be developed. Since the approach taken is to develop a generic approach to the problem in order to maximise potential usefulness over multiple domains, a guide to educators in presenting domain-specific questions to students using the courseware is also required. A plan is feasible if it can be implemented such that the requirements are fulfilled. Usefulness is achieved when a system provides results which are of benefit to practitioners.

To prove that the automation of the formative assessment of diagram-based coursework using CBA courseware is both feasible and useful, it is necessary to design and implement the extensions, deploy a course of exercises and analyse the results.

The three identified areas of extension are as follows:

- Extending the marking system to assess the aesthetics of student diagrams;
- Extending the marking system to allow mutually exclusive solution cases;
- Changing the system of feedback to provide only the highest priority comments to students.

The marking system should allow the aesthetic appearance of student diagrams to be assessed. The assessment of aesthetic appearance should accommodate a wide range of diagram domains but be able to provide useful insight into the diagram on a domain-specific basis. The aesthetic assessment should provide an analysis of the diagram appearance as a coherent whole. Feedback should be provided to students to indicate aesthetic improvements which would benefit the diagram.

The marking system should be able to accommodate solutions where several, mutually exclusive alternatives are available to the student. A student solution may provide an incomplete attempt to fulfil the solution using one of the solution cases. The marking system should identify which solution case the student is attempting to attain and provide useful feedback which would benefit the diagram. Mutually

exclusive alternatives may constitute multiple nodes and connections which differ between alternate versions of the model solution.

Feedback provided to students should be a truncated version of the feedback generated by the marking system. Feedback comments should be prioritised and the highest priority feedback comments should be returned to students. The idea is to induce an iterative process whereby students are encouraged to successively improve their solution and then re-submit to receive further feedback.

The idea of assessing the aesthetic appearance of student diagrams is novel. It has not previously been documented in the literature, presumably because diagram-based CBA is a research area in infancy. Research interest lies in developing a mechanism flexible enough to encompass multiple diagram domains whilst providing meaningful feedback to students. The feedback provided should be acceptable to human markers as a measure of validity. It should be noted that, although the assessment of diagram aesthetics has been identified as a requirement of formative assessment in this context, aesthetic assessment is likely to be of interest more widely, including in summative assessment using CBA.

The idea of a system of mutually exclusive solution cases is also novel to CBA. Previous CBA research with CourseMarker has relied upon checking for the presence or absence of defined tokens. If multiple model solutions are feasible then the wording of the question has been changed to "force" the student to adhere to one version. This has included explicitly "banning" the use of certain constructions in the question specification. Question marking involving simulation may provide a solution to this problem, but the approach is necessarily domain-specific. Fixed-response CBA does not encounter this problem by the restrictive nature of its design.

The idea of presenting truncated feedback to students within the context of free-response CBA is also novel. Fixed-response CBA such as MCQs often involves only one piece of feedback provided to the student. In free-response exercises such as diagram-based coursework a range of feedback can be generated based upon tests conducted upon the student answer. However, feedback is presented as a concise list of all tests conducted upon the solution. The solution here differs because only high-priority feedback is provided to the student, which can change upon each submission as the student improves the solution; hybrid systems involving human

marking can be used to accomplish the former objective, but it is unlikely that multiple submissions could be assessed due to resource constraints.

5.1.4 Summary

Section 5.1.1 defined the requirements for CBA, formative assessment and educational diagrams, based upon research previously summarised in chapters 2, 3 and 4 of the thesis. Section 5.1.2 assessed whether these requirements were met within the existing CourseMarker / DATsys system and placed the outstanding requirements — the assessment of diagram aesthetics, the assessment of solutions containing mutually exclusive solution cases and the presentation of prioritised, truncated feedback — within context in the general requirements for the areas of CBA, formative assessment and educational diagrams. Section 5.1.3 demonstrated how the proposed extensions relate to the aim and motivation of the thesis as outlined in Chapter 1 and outlined their novelty to the CBA field. Section 5.2 will consider each of the proposed extensions in turn and outline the detailed requirements which each extension must achieve. The scope of guidance required for educators and developers is also examined.

5.2 Detailed Requirements

This work proposes three extensions to the existing CourseMarker / DATsys system, as well as the creation of guidance to developers and educators to ensure successful development of domains and exercises, and solutions to the problem of enabling the formative assessment of diagram-based student coursework to be successfully automated using CBA courseware. Section 5.2.1 outlines detailed requirements necessary to allow student diagrams to be assessed in terms of their aesthetic properties. Section 5.2.2 outlines detailed requirements necessary to extend the system to flexibly mark coursework where mutually exclusive alternate solution cases are allowed. Section 5.2.3 details requirements to allow prioritised, truncated feedback to be delivered to students. Section 5.2.4 outlines the scope of advice needed if the resulting CBA courseware is to be used successfully by developers and educators.

5.2.1 Requirements for assessing the aesthetics of student diagrams

This research aims to enhance the formative assessment of diagrams in a domain independent way. Therefore, it is clear that the layout of diagrams in many domains will need to be assessed in a flexible manner.

Any approach to the assessment of diagram layout which aims to provide one concrete mechanism for the assessment of all diagram domains in a general way will result in assessment of only the most superficial aspects of diagram layout due to the conflicting requirements of different domains. Conversely, a 'blank slate' approach based upon applying an entirely different set of rules on a per-domain or even per-exercise basis will result in an unacceptable level of difficulty to the developer whenever the assessment of a new diagram domain is required.

Within a CBA context it is necessary to enable the marking of aesthetics in new diagram domains with an acceptable amount of development effort, while still providing capability to apply different conventions across disparate domains as required. Concretely, it is necessary to:

- Minimise the effort required to assess the aesthetics of a new diagram domain (to provide a *basis* for aesthetic assessment across common domains) ;
- Allow domain disparities to be accommodated through conducting a different aesthetic assessment (to make the system *extensible*);
- Allow educator preferences and priorities to be reflected (through *parameterisation*);

Since the extensions will be made to the existing CourseMarker / DATsys architecture, it is necessary to ensure compatibility and transparency to users. Specifically, the extensions should be:

- Integrated into the marking system;
- Integrated into the feedback system;
- Able to recognise existing conventions for specifying diagram formats;
- Transparent to students.

Within a formative assessment context the central requirement is feedback. Students should be provided with motivational feedback which is relevant to the shortcoming identified by the assessment procedure. Feedback comments should be short and prioritised; this requirement is further elaborated in section 5.2.3. The central requirement for assessing the aesthetics of student diagrams within the context of formative assessment is, therefore, that feedback is provided which is integrated with the extensions proposed for delivering truncated, prioritised feedback to students.

Within the context of educational diagrams, several requirements are essential to the approach. It is necessary to:

- Provide a basis for assessing educational diagrams generically;
- Provide a platform for extension to accommodate new domains;
- Assess diagrams based upon justified criteria;
- Allow the relative importance of criteria to be specified to take into account the fact that not all criteria contribute equally to the general aesthetic of the diagram, as outlined in section 2.3.4.1.

5.2.2 Requirements for assessing solutions with mutually exclusive alternate solution cases

Once again, the requirements must allow mutually exclusive alternate solution cases to be assessed by the system in a general, domain-independent way while minimising the effort required on the part of the exercise developer. Mutually exclusive alternate solution cases constitute alternate *subsets* of the model solution. It is necessary to distinguish between those parts of the solution which are common to all versions of the model solution and those parts which differ. Consider the simple example in figure 5.1, which shows two versions of a model solution.

Let M be the set of all model solutions. In the simple problem in figure 5.1 there are two model solutions, M_1 and M_2. Let I be the set of features common to all model solutions in M while $D_x = M_x - I$, all features in the model solution which are not common.

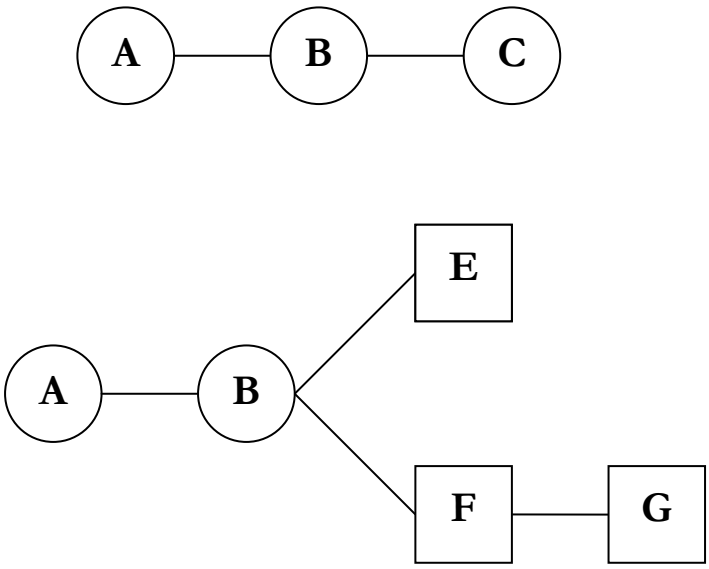

Figure 5.1: Two mutually exclusive model solutions

In figure 5.1, therefore:

$$M_1 = \{A, B, C, (a,b), (b,c)\}$$

$$M_2 = \{A, B, E, F, G, (a,b), (b,e), (b,f), (f,g)\}$$

$$I = \{A, B, (a,b)\}$$

$$D_1 = \{C, (b,c)\}$$

$$D_2 = \{E, F, G, (b,e), (b,f), (f,g)\}$$

If both the commonality and difference across model solutions is to be assessed successfully, a central requirement within the context of CBA is that the exercise developer be allowed to specify which features are common to all model solutions and then to specify the mutually exclusive features. Within this, the former requirement can be satisfied with relative ease since this effectively duplicates current features testing within CourseMarker. Within a real world context, however, the specification of the mutually exclusive solution cases requires flexibility.

In figure 5.1, $D_1 \cap D_2 = \{\ \}$. That is to say that the mutually exclusive solution cases contain *no* common elements. In practice, it cannot be guaranteed that a feature appears in either one *and only one* mutually exclusive solution case or else be common. Furthermore, the marking system of CourseMarker acknowledges, through its system of weighting, that not all features are of equal importance. Similarly, features within a mutually exclusive solution case may vary in importance.

In an educational context, a mutually exclusive solution cases occurs because more than one solution is plausible to the educator. This, in turn, occurs because more than one line of reasoning may be employed to develop a solution. Therefore, it is helpful to determine which features within each mutually exclusive solution cases denote the *reasoning* responsible for the solution case, and which features are dependent upon the original reasoning by virtue of being a logical continuation.

From this, the requirements for assessing solutions with mutually exclusive alternate solution cases flow. From a CBA perspective it is necessary to allow the specification by the exercise developer of the different solution cases. The exercise developer should be able to specify the common cases and each of the mutually exclusive solution cases, including which features within the solution case denote the difference in reasoning, in a way which can be used by the marking mechanism to assess the student solution appropriately. This is necessary to accommodate the generic approach which forms the foundation for this work. An exercise developer should be able to specify assessment criteria across domains with the minimum of development effort and maximum consistency, but the assessment of domain-specific diagrams must still allow sufficient scope to be meaningful.

Furthermore, from a CBA perspective, the proposed modifications must:

- Be integrated into the marking and feedback system system;
- Be seen to maximise transparency to students.

In fact, the first of these objectives leads to the second, since facilitating conventional operation of the marking system and allowing feedback to be delivered consistently with conventional CourseMarker CBA exercises will ensure that the student experience is consistent across CBA domains.

From the perspective of educational diagrams, the extensions must:

- Provide a basis for assessment of a wide variety of educational diagram domains;
- Allow educators to specify criteria for assessment and minimise constraints;
- Allow criterion weighting to account for features of unequal importance.

Specification of common and mutually exclusive features should be consistent across domains. The system should not constrain educators from applying any criteria set of their choice to the exercise; the approach is to supply the exercise developer, who is assumed to be a domain expert, with the facility for CBA rather than to act in a prescriptive manner.

Within a formative assessment context the central requirement is feedback. Students should be provided with motivational feedback which is relevant to the shortcoming identified by the assessment procedure. It is necessary for the assessment process to be able to determine which version of the model solution the student is attempting to attain and to give tailored, motivational feedback based upon the correct, mutually exclusive solution case which will improve the solution. Again, this extension must be integrated into the existing feedback mechanism and be compatible with the mechanism for providing truncated, prioritised feedback outlined in section 5.2.3.

5.2.3 Requirements for prioritising and truncating feedback to students

In order to fulfil formative assessment criteria, the feedback generated by the CourseMarker marking system must be truncated so as not to overwhelm the student. The most relevant information at each submission should be presented to the student while less relevant comments are omitted (possibly to be presented to the student after a later submission). Section 2.2.5 outlined criteria by which formative assessment feedback can be assessed. The primary requirement is to modify the feedback so that these criteria are met.

For this to occur, it is necessary to define a mechanism whereby feedback comments can be prioritised. If the relative priority of feedback comments after each submission

can be successfully defined then the task to delivering only the highest priority comments can be managed. Once the comments are prioritised, the extension must allow the comments to be delivered in a way which meets truncation criteria suited to the individual exercise.

Consequently, from a formative assessment perspective, the central requirements are to allow the prioritisation of comments and the definable truncation of the feedback. Flexibility for the exercise developer is key: it should be possible to define the truncation preferences to be applied by the marking and feedback delivery systems when developing the exercise. It is, furthermore, necessary to ensure that the feedback itself is motivational to students. This last issue will be addressed in the exercise developer guidelines, the scope of which is outlined in section 5.2.4.

Since this last extension is concerned with prioritising and truncating feedback which has already been generated, no direct requirements exist for this extension within the context of educational diagramming.

Within the context of CBA, the central requirements are the integration of the mechanism for prioritising and truncating feedback into the existing CourseMarker marking and feedback systems. To allow the process of assessment and feedback delivery to be automated and online, the mechanism for determining comment priority must operate without human intervention. This implies that comment priority must be able to be determined by information provided by the exercise developer and the results generated by the assessment process from the student submission. Truncation must be also be performed using criteria which are specified during the process of developing the exercise.

5.2.4 Scope of guidance needed for educators and developers

Section 3.1 provided a review of attempts to provide formative assessment capabilities using automated assessment courseware. Although the capabilities of the CBA courseware to provide formative assessment to students is crucial to success by definition, a central conclusion was that the process of development of the exercises themselves, most especially feedback, was also essential if the process of formative assessment was to contribute maximally to the learning process.

It would be neither necessary nor possible to provide a complete guide to educators and developers to the development of exercises in all domains. The pedagogical development of exercises is a research field in its own right and the subject of active debate. Chapter 2 briefly cited examples of research which aimed to develop assessment directly from cognitive taxonomies such as Bloom's taxonomy, but such methods are unproven even at the domain-specific level.

Conversely, a minimal guide to educators and developers whose function is to define the capabilities of the system and the necessary file formats would fail to provide educators with sufficient information to maximise the formative assessment potential of the system. It is necessary to find a "middle-ground", whereby practical advice can be provided to educators and developers within the context of developing formative assessment using the courseware while leaving domain-specific decision-making to be implemented by a specialist.

Chapter 3 noted several good examples of the delivery of good, motivational formative feedback to students using CBA courseware. Phrasing feedback motivationally and referring to learning material or providing research references can encourage further student research and motivate further, useful re-submission. From an educational point of view, it is important to illustrate how good formative feedback can be developed within the context of CBA. It is, furthermore, important to emphasise that CBA feedback comments are linked with assessment criteria, with the result that prioritisation of the underlying assessment criteria must be defined in such a way that the feedback comments are encountered by the student in a useful order which reflects the learning curve of the domain materials.

For developers it is necessary to make explicit the relationships between the pedagogic priorities of the formative assessment and the functions of the courseware responsible for their implementation. Specifically, guidance must be given to allow the developer to successfully:

- Indicate the relative priorities of assessment criteria within the exercise;
- Specify and integrate new aesthetic measures or configure existing measures;

- Facilitate the assessment of multiple model solutions by defining their commonality and variation points;

- Specify the method of feedback truncation to be used for the exercise.

5.2.5 Summary

Section 5.2 provided detailed specifications of the requirements which each of the proposed extensions must attain if the existing drawbacks outlined in section 5.1 are to be overcome. Section 5.2.1 outlined detailed requirements for diagram aesthetics to be assessed by courseware, including the need for a flexible and extensible system which, nevertheless, provides a basis for assessing the aesthetic properties of a wide range of education diagram types. Section 5.2.2 considered the requirements in marking exercises with multiple model solutions through considering the commonality across and variation between solutions. Requirements for educators to define those key features which denote the different reasoning between model solutions were outlined. Section 5.2.3 considered the provision of useful formative feedback in terms of the necessity to allow the prioritisation of comments and a system of defining the level of truncation applied to the features-linked feedback comments. Section 5.2.4 considered the scope of advice needed for educators and developers to develop formative exercises using the courseware: practical advice on the development of motivational feedback, plus an explanation of the relationships between the pedagogic priorities of the formative assessment and the functions of the courseware responsible for their implementation, are required.

5.3 Summary

Chapter 5 built upon the shortcomings of the CourseMarker / DATsys CBA system which were identified in Chapter 4. Section 5.1 discussed the requirements inherent in conducting formative, diagram-based CBA, demonstrated that the courseware can accommodate some of the requirements, especially those shared with conducting summative CBA, and placed the proposed extensions to the CBA courseware within context. The topics of fulfilling the criteria linked with CBA, formative assessment and educational diagrams were each considered in turn.

Section 5.2 developed detailed requirements for each of the extensions, considered within the context of CBA, formative assessment and educational diagrams. The

requirements for assessing aesthetic diagram criteria, accommodating mutually exclusive solution cases and prioritising and truncating the feedback of the system were each considered in turn. The scope of guidance to be provided to educators and developers was also discussed. Based upon these guidelines, the extensions can be designed. Based upon the requirements outlined within this chapter, Chapter 5 documents the design decisions made in the context of each of the extensions and the integration into the existing courseware.

Chapter 6

Designing the extensions

Introduction

This chapter describes a solution to the problem of automating the formative assessment of diagram-based coursework using CBA courseware. It describes the process of designing the extensions to the courseware which were identified as necessary in Chapter 5: assessing the aesthetics of student diagrams, considering solutions with mutually exclusive alternate solution cases and prioritising and truncating the student feedback. The design meets the detailed requirements identified in section 5.2.

Section 6.1 presents a high-level overview of the design. The issues of ensuring that the design meets the identified requirements and that the components are successfully integrated are discussed and a brief overview of the approach in each area is provided. Section 6.2 describes the design of the extension which enables the assessment of the aesthetics of student diagrams. A series of aesthetic measures are chosen to represent commonality across educational diagram domains, while domain-specific structural measures can be implemented through extension. The hierarchy and weighting system is described, together with the individual aesthetic measures. Section 6.3 outlines the extension which deals with solutions which possess mutually exclusive alternate solution cases. The approach is based upon the notion that some uncommon features, designated *harbingers*, define the difference in reasoning between the mutually exclusive cases. Other features, which are present in all model solutions, are designated common. Responsibility for defining the solution is defined and the integration into the features testing system is discussed. Section 6.4 presents the extension responsible for prioritising and truncating the feedback provided to students. The system of prioritisation is described and responsibilities for users defined. A configurable system for truncating the prioritised results is presented.

6.1 High Level Overview

The central purpose which motivates the design of the extensions described in this chapter is to allow research to be conducted with the aim of proving that formative CBA in diagram-based domains is feasible and useful. This assessment is feasible if the extensions can be successfully designed and implemented and if exercises in educational diagram domains can be developed with realistic levels of time and effort. The assessment is useful if the exercises assist the process of student learning.

The design assumes that the eventual courseware will be used by three distinct categories of users:

- *Developers*, who are responsible for developing new diagram domains and carry responsibility for configuring the marking tools and specifying the tests to be conducted;
- *Teachers*, who are domain experts who can design exercises, including exercise specifications to be presented to students, possible model solutions and useful feedback;
- *Students*, whose learning process is the focus of the formative assessment.

The approach is intended to facilitate a domain-independent environment where new domains can be assessed through specification and extension, carried out by developers. Chapter 4 outlined the reasons for using the CourseMarker / DATsys system as a development base. The design is, therefore, able to take advantage of considerable existing design infrastructure. It is incumbent upon the extensions, however, to integrate with the existing architecture in order to provide a smooth, coherent experience for the student users.

The design must consider the trade-off between, on the one hand, providing a realistic basis for formative assessment without overwhelming developmental requirements and, on the other hand, restricting the cross-domain potential for assessment through allowing insufficient flexibility in extension. At each stage, the intention is to provide a concrete basis useful for common, node-link type, educational diagram domains whilst specifying flexibility through extension points and parameterisation.

6.1.1 Requirements

Section 5.2 outlined set of detailed requirements which must be fulfilled by each of the extensions if success in formative assessment is to be achieved. The design must be shown to explicitly meet each of these requirements in turn. Generally, the requirements aim to ensure that the resultant courseware achieves an optimal trade-off between flexibility and developmental effort, integrates seamlessly into the existing architecture and provides a clearly defined role for each of the system's users.

6.1.2 High Level Design

This section aims to provide a brief overview of the design strategy for each of the extensions. The aim is to provide an introduction of the strategy used to ensure that the extensions are effective and meet the requirements set out in section 5.2. A high-level overview of the integration between the extensions is then considered in section 6.1.3 before the detailed design decisions are discussed in section 6.2 to 6.4.

6.1.2.1 Assessing the aesthetics of student diagrams

The design of the extension to allow the assessment of the aesthetic properties of student diagrams is based around the aggregation of input from a series of measures which each examine distinct aesthetic properties of the student diagram. Each measure is applied to the diagram, returning a scaled numeric mark and a piece of motivational feedback. Some diagram domains are subject to domain-specific aesthetic rules. For this reason, a key distinction is made between *aesthetic measures* and *structural measures*. These are defined as follows:

- *Aesthetic measures* are domain-independent and based upon the relationships between the nodes and links within the diagram and the drawing canvas on which the diagram has been created;

- *Structural measures* are domain-specific and based upon knowledge of the rules governing the relationships between types of links and nodes, as defined by the convention of meaning associated with the educational diagram domain.

These two distinct types of measures encapsulate the commonalities across and differences between domains of educational diagrams.

Aesthetic measures provide a basis for the marking of general diagram layout across a range of educational diagram domains. Existing aesthetic measures are based upon mathematical graph layout criteria and studies of aesthetics in graphical user interface design. New aesthetic measures may be added by developers upon discovery, but this process is likely to be irregular. The task of the educator with regard to aesthetic measures is to specify the relative importance of the aesthetic criteria through the allocation of weights to the aesthetic measures. This process reflects the fact that, although certain measures of aesthetics are applicable across educational diagram domains, their importance varies across domains.

Structural measures provide the means by which the marking of the layout of student diagrams can be extended to accommodate domain-specific requirements specified within the convention of meaning of an educational diagram domain. Structural measures are identified by the educator when a new domain is to be assessed. The educator is also responsible for defining the relative importance of the new structural measure through the allocation of a numeric weight. Developers create new structural measures as required, each time a new educational diagram domain is to be assessed.

Aesthetic and structural measures both constitute marking tools with similar aims. The distinction between the two is pedagogical. The way in which the measures are implemented in each case is similar, although a hierarchy is used to make the distinction between the two clear for the purposes of avoiding confusion.

The marking scheme is responsible for calling marking tools and providing parameters. Both aesthetic and structural measures must accept three parameters:

- The student diagram to be assessed;
- The relative weight of the measure;
- A *leniency value*.

The student diagram to be assessed is represented within DATsys as a `Diagram` object. The relative weight of the measure is provided as a real number. The leniency value is used for linear scaling purposes, based upon the maximum value for the measure which the educator can reasonably expect the student to obtain.

The criteria against which student diagrams are judged may be derived from theoretical formulae which assume an ideal diagram scenario. Due to circumstances beyond the control of the student, therefore, it may be impossible to obtain a score of 100% from one or more measures due to the nature of the nodes and links (which are defined by the developer) or the circumstances of the model solution (which is defined by the educator). If the assessment is to be valid (as defined in table 2.1) then these external assessment qualities, which are not based upon a reflection of the ability of the student, should be maximally excluded. Therefore, it is incumbent upon the developer to determine, by considering the model solutions, the base level in each criterion which it is reasonable to expect the student to attain. This value can be used, in a process of linear scaling, to ensure that this value is "scaled up" to 100%, with other values being proportionately scaled through a linear process.

Since developers may develop new measures, both aesthetic and structural, to extend the functionality of the CBA courseware further, it is necessary to define a standard to which all measures must conform. An interface is used, therefore, which all aesthetic and structural measures must implement. The interface enforces the acceptance of the three parameters and the return of a `MarkingResult` object to enable integration with the CourseMarker marking system.

Therefore, the architecture of the extension consists of a package `layout`, which is located in the CourseMarker marking system and which contains the interface `LayoutToolInterface`. The package `layout.aesthetic` contains classes which implement the aesthetic measures, while the package `layout.structural` is provided to developers to add domain-specific structural measures.

The exercise marking scheme is used to call the aesthetic measures. If structural measures are present then they too are invoked by the marking scheme. The student drawing, the relative weight of the measure and the leniency value are passed as

parameters to the measures, which each return a `MarkingResult` containing the weighted mark and motivational feedback.

Section 6.2 describes in more detail the concrete design of this extension. The linear scaling system is described and the `LayoutToolInterface` interface is fully defined. Suitable aesthetic measures are chosen from criteria in the fields of graph layout and GUI design, and their transformation into marking tools in the `layout.aesthetic` package is outlined. The commonality in design and implementation between aesthetic and structural measures is discussed and their different usage explained. Finally, the integration into the existing CourseMarker architecture is described and the intersection points made clear. The design is shown to arise from the detailed requirements listed in section 5.2.1.

6.1.2.2 Assessing solutions with mutually exclusive alternate solution cases

The design of the extension to allow the assessment of solutions with mutually exclusive alternate solution cases arising from the acceptability of multiple model solutions is based upon the notion of allowing the definition of *features set cases*. The features set cases are derived from the acceptable model solutions. The educator defines the acceptable model solutions for the exercise as part of the process of creating the exercise and outlines, to the extent possible, those features which denote the difference in reasoning which resulted in the model solution case. The developer takes the model solutions and identifies those elements (nodes and links) which are common to all model solutions, defined in section 5.2.2 as I. Features tests which search for the elements in I and the relationships between them are constructed by the developer. These tests constitute the first features set case, FT_0.

Subsequent features set cases, $FT_1 ... FT_x$, contain features tests whose success depends on the presence of the elements and links present within a mutually exclusive solution case, defined in section 5.2.2 as D_x. These features tests are, by definition, *uncommon* features which are not present in all model solutions. The first features test within each features set case $FT_1 ... FT_x$ ideally denotes a feature test whose search criteria checks for an element or combination of elements which is unique to the *specific* mutually exclusive alternate solution case. This feature test is known as the *distinction test* since it is used to distinguish between alternate mutually

exclusive solution cases. The element or combination of elements in the model solution case which is used to distinguish the case from all others is called the *harbinger*.

In the ideal situation an element or combination of elements which represent the *line of reasoning* that resulted in the student arriving at that particular version of the model solution is identified by the educator and used by the developer for the distinction test; in this case a *perfect harbinger* has been found. This situation is ideal since the feedback can be focused on the specific line of reasoning associated with the specific model solution.

It is possible that the educator is unable to describe a precise line of distinct reasoning which leads to each version of the model solution. The design can still be used to assess student solutions where multiple model solutions are plausible without it. The minimum requirement for the distinction test is that it searches for an element or combination of elements which is *unique* to the specific model solution; such an element or combination of elements must exist or, by definition, the features set case is not assessing a mutually exclusive alternate solution case. Such an element or combination of elements is an *imperfect harbinger*.

After creating the teaching materials, exercises and one or more model solutions, the educator is responsible for highlighting those unique elements within each model solution which will be used to determine the distinction test. The educator also prioritises the features and is responsible for generating positive, motivational feedback. This responsibility lies with the educator across a wide variety of automated assessment cases; the task is onerous, but since the exercises may be repeatedly re-used in a formative assessment context the time can be justified. Guidance for generating positive, motivational feedback is outlined in chapter 7. Features tests associated with common elements, FT_0, are placed into the first features tests file, `[exercisename].ft0`. Features test cases $FT_1 ... FT_x$ are placed into features test files `[exercisename].ft[x]` with the features tests representing the distinction test defined as the first features test in each file. The structure of individual features tests remains, as previously defined in chapter 4, as follows:

- `Mark weight : Feature expression : Description : Positive feedback : Negative feedback`

The `DiagramFeaturesTool` marking tool is based upon the same principles as the `EntityRelationshipTool` tool described in section 4.2. Nodes are identifiable by their Name and Text Content while links are identifiable by their Name, Start Node and End Node. The features expression types *exist*, *exact*, *connection* and *exactConnection* have usage and meaning consistent with EntityRelationshipTool. The feature expression type *compositeRelationship* is included for backward compatibility purposes but its usage is discouraged due to its domain-specific nature.

The exercise marking scheme is used to call the `DiagramFeaturesTool` once for each features test file. The student drawing file and the correct features test file are passed to the `DiagramFeaturesTool` as parameters as each call is made. The `DiagramFeaturesTool` returns a `MarkingCompositeResult` at each call. The remaining task is to parse the `MarkingCompositeResult` tree to determine which of the mutually exclusive solution cases the student solution attempts to emulate. The remaining `MarkingCompositeResult` objects can be pruned from the tree accordingly.

The process of parsing the marking tree to determine the best solution case to consider and then truncating the tree to remove the other cases is accomplished as part of the responsibility of the `PrioritiseTruncateTool` described in section 6.1.2.3. Since much of the functionality of truncating and prioritising the tree duplicates the functionality required to prioritise and truncate the student feedback in general, the construction of a separate tool was not justified. The allocation of responsibility for prioritising the alternate mutually exclusive solution cases to the `PrioritiseTruncateTool` constitutes the primary relationship and interdependency between the extension to enable the assessment of mutually exclusive solution cases and the extension to prioritise and truncate student feedback. The integration of the extensions is summarised more generally in section 6.1.3.

Section 6.3 describes in more detail the concrete design of this extension. The responsibility of users, the defining of features and the use of the marking scheme to search for features are described. Finally, the integration into the existing

CourseMarker architecture is described and the intersection points made clear. The design is shown to arise from the detailed requirements listed in section 5.2.2.

6.1.2.3 Prioritising and truncating feedback to students

The design of the extension to allow the delivery of prioritised and truncated feedback to students is based around providing an extensible, flexible and configurable mechanism for the developer to encapsulate different methods of prioritisation and truncation, based upon the wishes of the educator.

The problem of prioritising and truncating the feedback to students can be divided into four smaller tasks, each of which may be accomplished in a number of different ways. The four smaller tasks are:

1. Establishing which of the competing mutually exclusive solution cases has the highest priority and deciding what course of action to take with regard to the feedback generated by the other mutually exclusive solution cases;

2. Prioritising the feedback provided by all features tests;

3. Prioritising the feedback generated by the aesthetic and structural measures;

4. Truncating the feedback.

To the solving of each of these smaller tasks, multiple strategies could be applied depending upon context-dependent factors such as the nature of the domain, the details of the assessment, the type of the students and the preferences of the educator.

A strategy for solving the first problem could be examining the distinction test for each mutually exclusive solution case, determining the distinction test with the highest score and pruning all other mutually exclusive solution cases from the feedback tree. An alternative strategy could examine other features tests within the mutually exclusive solution case to determine if features from multiple cases were being confused by the student.

A strategy to solve the second problem could sort the feedback from the common elements features tests FT_0 together with the remaining mutually exclusive solution

cases to determine the most important feedback overall. Alternately, a strategy could prioritise comments from FT_0 in the event of a low overall mark and introduce feedback from mutually exclusive solution cases only when the student solution passes a given threshold.

A strategy to solve the third problem could inter-mingle aesthetic and structural measures and determine the highest priority comments overall, or try to prioritise structural measures in the early stages to ensure domain correctness before emphasising feedback comments from aesthetic measures later once a threshold is reached.

Truncating the feedback could involve retaining a specific number of features comments and a specific number of layout comments, both specified by the educator. Alternately, the topmost percentage of comments above a threshold could be provided to the student.

The design makes use of the object-oriented Design Pattern known as *Strategy* [GHJ+94]. The intent of the *Strategy* pattern is to define a family of interchangeable algorithms, allowing the algorithm to vary independently from the clients which make use of it.

The *Strategy* pattern has three participants: the *Strategy*, the *Concrete Strategy* and the *Context*. The *Strategy* defines an interface common to all supported algorithms, while a *Concrete Strategy* implements a specific algorithm whilst conforming to the *Strategy* interface. The Context is configured with a *Concrete Strategy* object, maintains a reference to a *Strategy* object and may define an interface which lets *Strategy* access its data.

In this case the four tasks to be completed in the prioritisation and truncation of student feedback are translated into four *Strategy* interfaces. These are, in order of the problems listed above, the `SolutionCaseStrategy`, `FeaturesSortStrategy`, `AestheticsSortStrategy` and `TruncationStrategy` interfaces.

The `PrioritiseTruncateTool` acts as the Context to all four strategies. The tool is configured with four objects representing the configured algorithm to be employed at each of the four stages of the prioritisation and truncation process.

The educator specifies the methodology to be used at each stage of the prioritisation and truncation process. The developer then develops a *Concrete Strategy* for each stage containing an algorithm which encapsulates the methodology defined by the educator, and which conforms to the *Strategy* interface responsible for the specific stage of the process. The educator can then specify parameterisation on a course-specific (or, if desired, exercise-specific) basis.

For example, an educator could decide on a truncation strategy for stage 4 of the process which involves retaining a specified number of feature-related feedback comments and another specified number of layout-related feedback comments. This methodology could be used by the developer to develop a *Concrete Strategy* called `OrdinateTruncationConcreteStrategy` which conforms to the `TruncationStrategy` interface. For a given exercise, the educator could decide to retain precisely the 2 most relevant feature-related feedback comments and the 2 most relevant layout-related feedback comments. This information is used, in the form of parameters, in the construction of a new `OrdinateTruncationConcreteStrategy` object which, in turn, is used as one of the four *Concrete Strategies* necessary to configure the `PrioritiseTruncateTool`.

Section 6.4 describes in more detail the concrete design of this extension. The `PrioritiseTruncateTool` is defined, along with the four *Strategies* used for configuration. The responsibility of users is made explicit, the integration into the existing CourseMarker architecture is described and the intersection points are made clear. The design is shown to arise from the detailed requirements listed in section 5.2.3.

6.1.3 Extension Integration

Figure 6.1 illustrates a high-level view of the relationships between the extensions discussed in this section. The student solution is marked through a Marking Scheme which invokes configured instances of each extension in turn.

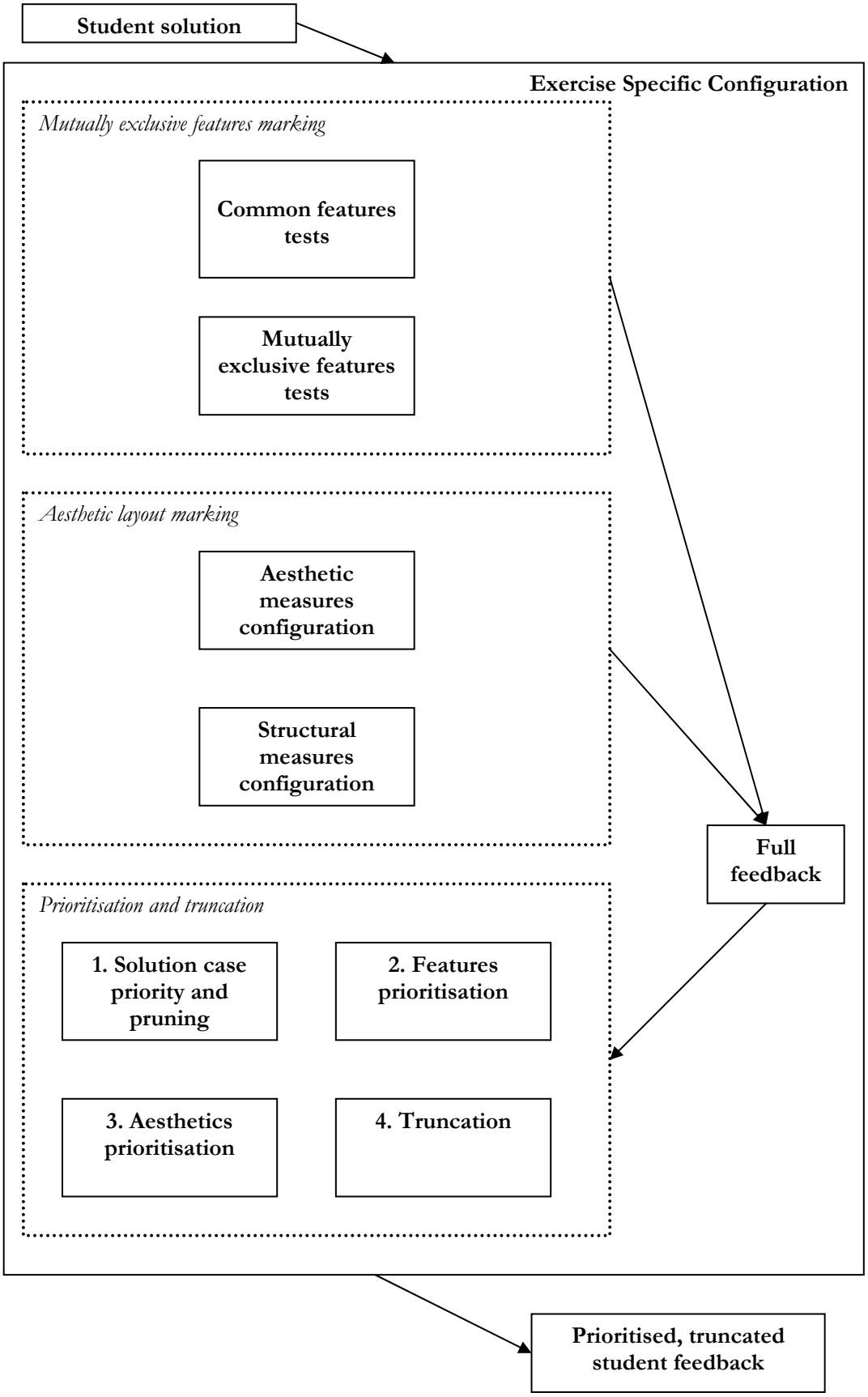

Figure 6.1: A high-level view of the relationships between the extensions

A Composite Marking Result is created by the Marking Scheme to hold the feedback returned by each extension. A Composite Marking Result operates using a tree-like structure designed to be intuitive to the student when feedback is presented. Each node on the Composite Marking Result may be either a Marking Leaf Result, which contains a mark value, weight value, description and feedback comment or, recursively, another Composite Marking Result.

The first two extensions may be invoked in either order. The extension which solves the problem of assessing multiple, mutually exclusive solution cases within the student solution, runs features tests for the common features and *each case* of the mutually exclusive features identified by the marking scheme. A Composite Marking Result is generated upon each run-through of the features test tool `DiagramFeaturesTool`. Each of the Composite Marking Results contains a Marking Leaf Result for each features test specified in the test case. If there are no mutually exclusive solution cases then only one Composite Marking Result is generated, that for the common features.

The extension for aesthetic layout marking runs the aesthetic measures and the structural measures specified in the configuration. Two Composite Marking Results are added to the feedback: the first contains the Marking Leaf Result feedback generated by the aesthetic measures, whilst the second contains that generated by the structural measures. If no structural measures are invoked then this Composite Marking Result is empty.

The extension for the prioritisation and truncation of student feedback is invoked last. The `PrioritiseTruncateTool` is configured using four legal *Concrete Strategies*, each of which is applied in order. The first strategy prioritises the mutually exclusive solution cases to decide which mutually exclusive solution case is most relevant to the student solution. Depending upon the strategy, all other mutually exclusive solution cases may be subsequently ignored by pruning the appropriate Composite Marking Result branches. The second and third strategies prioritise the feedback branches provided by the features tests and the aesthetic layout tests respectively. The final strategy truncates the feedback.

The resultant Composite Marking Result, modified at each stage, is presented to the student using CourseMarker's existing feedback mechanism.

6.1.4 Summary

This section provided an overview in words of the design strategy for each of the three courseware extensions proposed within this work. The assessment of the aesthetic layout of student diagrams is accomplished through distinguishing between aesthetic measures, which are domain-independent and structural measures, which are domain-specific. An interface is defined to ensure compatibility between the measures. The assessment of student solutions where multiple model solutions are viable, containing mutually exclusive solution cases, is accomplished by identifying the commonality and variation between model solutions. Common features are assessed using a first solution case, while subsequent solution cases assess the remaining features within each model solution. The fundamental difference between solution cases is identified by a *harbinger*. Prioritisation and truncation of feedback is accomplished through specifying four sub-tasks. The algorithm for each sub-task can be specified differently within context but must implement an interface for compatibility. The design for this third extension was based upon the *Strategy* design pattern.

Finally, the section provided an overview of the relationships between the three extensions.

This section aimed to provide an overview and a sense of context to each of the extensions. The subsequent sections within this chapter offer further detail, as required, on the specific design decisions taken within the context of each extension, for the purposes of ensuring that the specific requirements identified in chapter 5 are fulfilled.

6.2 Assessing the aesthetic layout of student diagrams: resolving the design issues

Section 6.1.2.1 outlined the approach to assessing the layout of student diagrams. The approach is based upon implementing marking tools to assess a wide variety of marking criteria. Broadly, these criteria can be divided into aesthetic measures, which are domain-independent, and structural measures, which are domain-specific. A hierarchy was described which separates the aesthetic and structural measures into separate packages. The marking tools for each marking criteria must implement

an interface and are invoked by the exercise marking scheme. This section demonstrates the link between the design decisions and the detailed criteria for the extension provided in section 5.2.1. Concrete design decisions for the class hierarchy, top-level interface, scaling mechanism, aesthetic measures and structural measures are made explicit, such that implementation may be achieved.

6.2.1 Linking the design to the requirements

Section 5.2.1 identified the key requirement in assessing the aesthetics of student diagrams as the task of ensuring that diagrams in many domains can be assessed in a flexible manner. The system of aesthetic and structural measures achieves this through considering the domain context of each marking criterion. Aesthetic measures are domain-independent and can be called upon to assess diagrams from many educational domains, while the educator is able to specify further, domain-specific, criteria to be implemented as structural measures.

The design also minimises the effort required to assess the aesthetics of a new diagram domain. Only the *unique* features of a diagram domain need be adapted into structural measures. Most common attributes can be assessed by the existing aesthetic measures, which provide a basis for all domains. It is plausible that, in many cases, no unique layout rules exist for a new educational domain, in which case the sole task is to specify the relative weights of the existing aesthetic measures. Structural measures should not be regarded as indispensable for every diagram domain — they simply act as an extension point to allow unique domain disparities to be accommodated. Educator priorities may be expressed through changing the relative weights of the aesthetic and structural measures. There is also the further option of defining prioritisation strategies between aesthetic and structural measures, as discussed in section 6.1.2.3.

The extension is integrated into the existing marking and feedback system. Aesthetic and structural measures are CourseMarker marking tools which can be called from within the exercise marking scheme. The measures return their results as a CourseMarker standard marking result which can be returned to the student using the existing feedback mechanism. In practice, the marking result will be subject to modification by the extension to enable the delivery of prioritised, truncated feedback before it is presented to the student, but the process is still transparent from

the student's point of view. The aesthetic and structural measures recognise the existing conventions for specifying diagram formats (in Daidalos) through the mechanism of enumerating the diagram components, in the same way as existing diagram features marking tools.

The extension provides a basis for assessing educational diagrams generically through the implementation of aesthetic measures, while structural measures constitute the platform for extension. Aesthetic measures for diagrams are based upon justified criteria. Aesthetic measures are drawn from documented aesthetic criteria in the fields of graph layout and user interface design with demonstrated real-world application. Structural measures can be specified on a domain-specific basis by a domain expert. The relative importance of the criteria, based upon either research, anecdotal evidence or simply the "gut feeling" of the educator, can be specified through the system of weighting.

6.2.2 Hierarchy

Figure 6.2 illustrates the hierarchy of the extension. The `layout` package is positioned at the top-level of the hierarchy, while both the `aesthetic` and `structural` package occupy one level beneath the layout package. The `LayoutToolInterface` interface is located within the `layout` package.

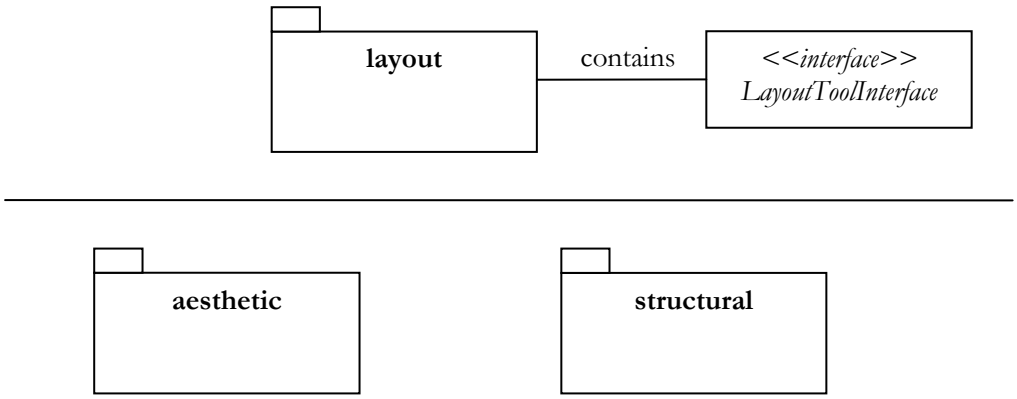

Figure 6.2: The hierarchy of the aesthetic layout extension

Figure 6.3 illustrates the locations of Marking Tools representing both aesthetic and structural measures. Marking tools for aesthetic measures are located within the `aesthetic` package whilst marking tools for structural measures are located within

the `structural` package. Both aesthetic and structural measures must implement the `LayoutToolInterface` interface.

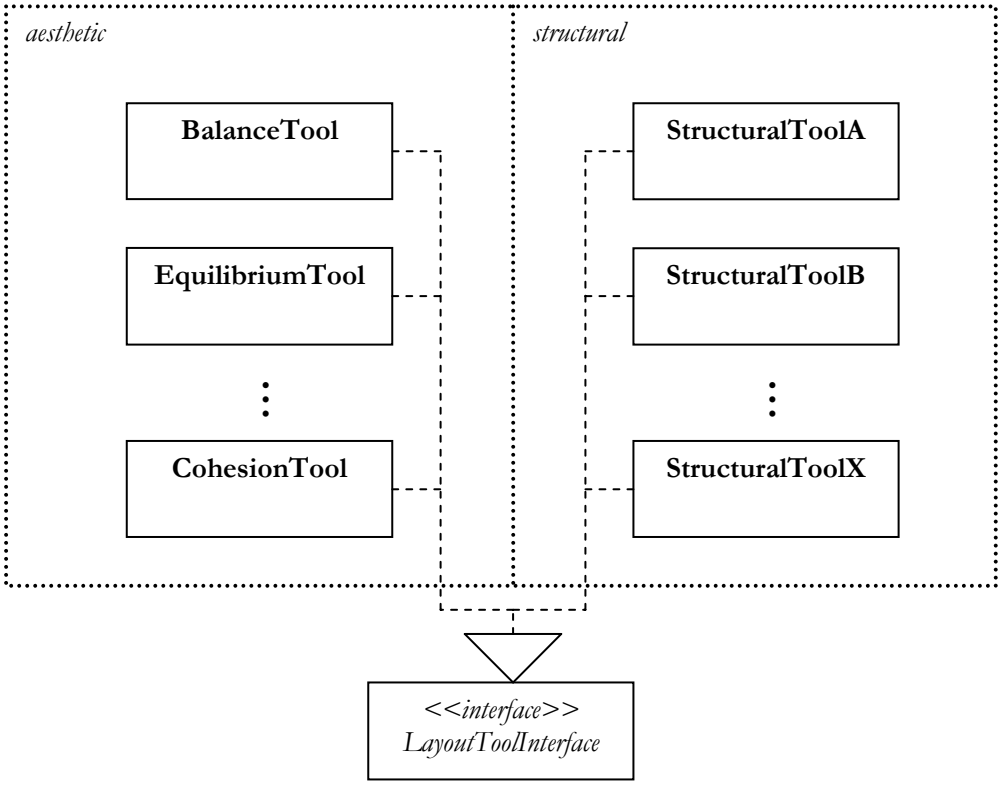

Figure 6.3: Aesthetic and structural measures implement LayoutToolInterface

6.2.3 Interface

The `LayoutToolInterface` interface is shown in figure 6.4. The interface contains one method which must be implemented: `mark`.

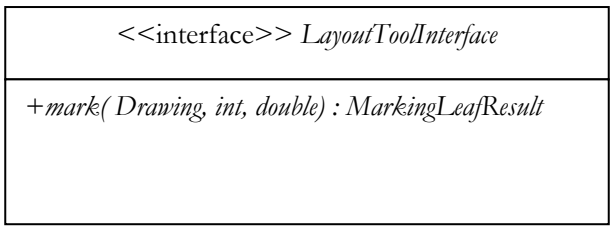

Figure 6.4: The LayoutToolInterface interface

The `mark` method requires three parameters: the student diagram to be assessed, the relative weight and the leniency value for scaling purposes. The method returns a `MarkingLeafResult`. The `MarkingLeafResult` is defined within the

CourseMarker marking system as an extension of `TMarkingResult` which encapsulates the following data:

- *markvalue*, an integer representing the final, scaled mark returned by the test;
- *weight*, an integer representing the relative weight of the feedback result;
- *description*, a String holding the description of the test;
- *feedback*, a String holding the feedback returned by the test.

6.2.4 Scaling

Aesthetic and structural measures use scaling to translate the raw score into a suitable mark to return embedded in the `MarkingLeafResult`. The linear scaling mechanism, which has been in use since the time of Ceilidh [ZF94], requires but a trivial modification to take into account the fact that aesthetic and structural measures return scores between 0 and 1. Figure 6.5 illustrates the simple relationship between the raw score and the scaled mark.

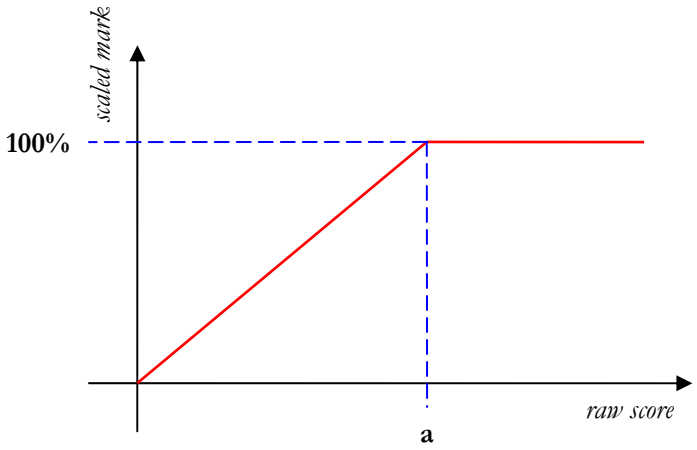

Figure 6.5: The relationship between the raw score and the scaled mark

The value *a* represents the leniency factor, the maximum raw score which the educator feels it is reasonable to expect the student to achieve for this measure. All raw scores above the leniency factor are scaled to 100%, while scores below the leniency factor are scaled to a percentage of *a*.

6.2.5 Aesthetic measures

Section 2.3.4 introduced the concept of educational diagram aesthetics and examined aesthetic criteria from the fields of graph layout and user interface design. Furthermore, section 2.3.4 explained that graph layout criteria from syntactic graphs must be shown to be justified in a real-world context before being used to assess educational diagrams. The process of choosing aesthetic criteria, from which to develop aesthetic measures, was based upon the notion that several requirements must be considered:

- The criterion must not be domain-specific and must have the potential for relevance across a wide variety of educational diagram domains;
- The criterion must be able to be expressed algorithmically such that a numeric value in the range 0 to 1 can be assigned to a student diagram to indicate student compliance;
- The criterion must not require the student to conduct complex modification to their solution for the purposes of "assisting" the algorithm to be successfully applied.

On the basis of these requirements, eleven criteria were chosen to be implemented as aesthetic measures. Two of these, the principles of non-intersection and non-interception, were taken from the field of graph layout. The remaining nine — balance, equilibrium, unity, proportion, simplicity, density, economy, homogeneity and cohesion — were taken from the field of user interface design. The criteria fulfil the first two requirements completely. The third requirement is not fulfilled by several of the criteria taken from the field of user interface design, which requires the student to modify their solution slightly, prior to submission.

Subsequent sub-sections document the process of transforming these criteria into aesthetic measures. Section 6.2.5.1 documents the process of designing the aesthetic measures to assess non-intersection and non-interception, including the need to identify a suitable formula, provide a clear design for the class, scale the raw score and conform to the `LayoutToolInterface` interface. Existing formulae exist for calculating compliance to the criteria taken from graphical user interface design. Section 6.2.5.2 documents the process of creating an aesthetic measure to assess the

property of equilibrium, while section 6.2.5.3 provides an overview of creating the other, similar aesthetic measures. Finally, section 6.2.5.4 describes and justifies the student modifications which must be made if several of the measures inspired by graphical user interface design are to be effectively measured and demonstrates that the required modification is insufficiently major to disrupt student learning.

6.2.5.1 The aesthetic measures for non-interception and non-intersection

The first necessary step in the process of designing an aesthetic measure is the identification of an appropriate method to determine compliance with the criterion numerically. Non-interception, as discussed in section 2.3.4.2, refers to minimising the number of lines in the diagram which cross over other lines. Non-intersection refers to minimising the number of nodes which intersect other nodes. In this case equation 6.1 is sufficient to determine non-interception, where c is the number of valid lines that intercept another and l is the number of valid lines on the canvas.

$$M_{non-\text{int}erception} = 1 - \frac{c}{l}$$

Equation 6.1: The non-interception measure

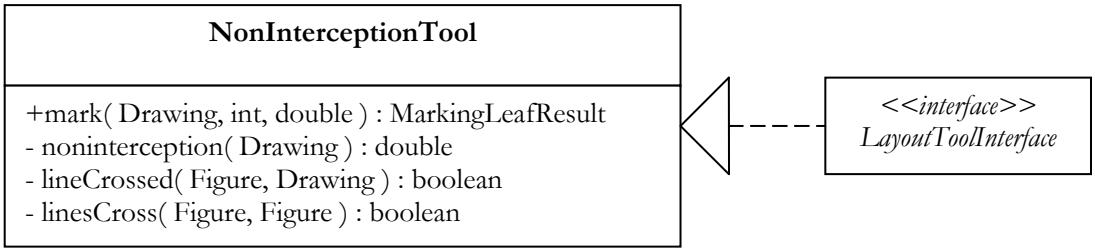

Figure 6.6: The design of the non-interception tool

The design for the `NonInterceptionTool` is simple. The `linesCross` method returns true if two line figures cross each other. The `lineCrossed` method returns true if a line is crossed by any other line in the drawing by repeatedly invoking `linesCross`. The `noninterception` method counts the number of lines and the number of lines which are crossed before applying equation 6.5 to obtain the raw score. The `mark` method, which must be defined in order to implement the `LayoutToolInterface` interface, invokes `noninterception` to obtain the raw score, applies scaling to obtain the mark and calls the `MarkingLeafResult`

constructor, supplying the scaled mark and weight, the internal description and the associated feedback before returning the `MarkingLeafResult`. Figure 6.6 presents an overview of the `NonInterceptionTool` using this design.

Similarly, the design of the non-intersection tool is based upon equation 6.2, in which o is the number of nodes that intersect at least one other node, while t is the total number of valid nodes on the canvas. Non-intersection refers to minimising the number of valid nodes in the diagram which overlap. The `NonIntersectionTool` operates similarly to the `NonInterceptionTool`, and is summarised in figure 6.7.

$$M_{non-\text{int}\,er\,\text{sec}\,tion} = 1 - \frac{o}{t}$$

Equation 6.2: The non-intersection measure

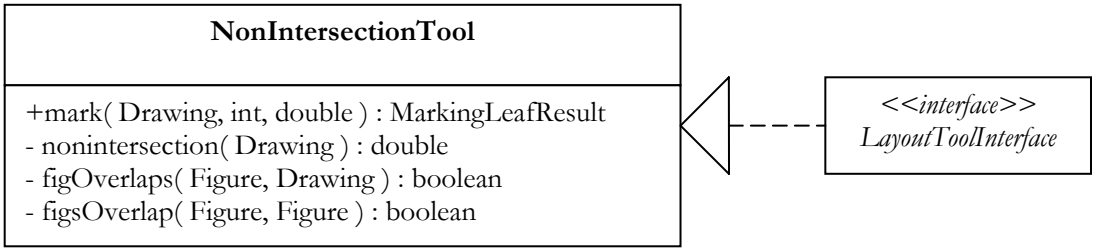

Figure 6.7: The design of the non-intersection tool

6.2.5.2 The aesthetic measure for equilibrium

This section outlines the process of designing the aesthetic measure for equilibrium. The process of designing the aesthetic measures for balance, unity, proportion, simplicity, density, economy, homogeneity and cohesion was very similar to the process of designing the aesthetic measure for equilibrium; section 6.2.5.3 discusses the design process of these aesthetic measures, based upon the process described here.

Table 2.3 has previously given a brief description of the equilibrium criterion as "*The difference between the centre of mass of the elements and the physical centre of the screen / canvas*". Ngo et al [NTB00] provide an extended definition of equilibrium, together with formulae to enable equilibrium to be calculated. These formulae are reproduced here as equations 6.3, 6.4 and 6.5.

$$EM = 1 - \frac{|EM_x| + |EM_y|}{2}$$

Equation 6.3: Equilibrium

$$EM_x = \frac{2\sum_i^n a_i(x_i - x_c)}{nb_{frame}\sum_i^n a_i}$$

Equation 6.4: *x*-axis equilibrium component

$$EM_y = \frac{2\sum_i^n a_i(y_i - y_c)}{nh_{frame}\sum_i^n a_i}$$

Equation 6.5: *y*-axis equilibrium component

In equations 6.3, 6.4 and 6.5, EM is the equilibrium measure, EM_x is the *x*-axis equilibrium component, EM_y is the *y*-axis equilibrium component, (x_i, y_i) and (x_c, y_c) are the co-ordinates of the centres of object i and the frame, a_i is the area of object i, b_{frame} and h_{frame} are the width and height of the frame and n is the number of objects on the frame.

The design process for the aesthetic measure for equilibrium is similar to that for the non-interception and non-intersection measures discussed in section 6.2.5.1, but the need to develop mathematical formulae to enable the calculation of the measure numerically is obviated by the existence of such formulae in the existing literature. A `mark` method calls a method `equilibrium` which invokes methods to calculate the *x*- and *y*-axis equilibrium components, and so on. The design of the aesthetic measures for criteria based upon user interface design principles is thus rendered a straightforward, if laborious, process.

The `Figure` objects embedded in each `Drawing` object can be accessed by means of a `FigureEnumeration`. Each `Figure` contains 'getter' methods for attributes including its centre and size. Consequently, the only modification required to complete the required calculation is the specification of the diagram's border. This procedure, and the pedagogical issues surrounding it, is outlined in section 6.2.5.4.

6.2.5.3 The aesthetic measures for balance, unity, proportion, simplicity, density, economy, homogeneity and cohesion

The aesthetic measures for balance, unity, proportion, simplicity, density, economy, homogeneity and cohesion were developed in the same way as the aesthetic measure for equilibrium. Mathematical formulae allowing these measures to be determined and compliance expressed numerically are already available in the literature. Ngo et

al [NTB00] provide an overview of interface aesthetics. The aesthetic measures for balance, unity, proportion, simplicity, homogeneity and cohesion were based around the equations presented in [NTB00]. The aesthetic measures for density and economy were based upon the equations published in [NB00]. Section 6.2.5.4 justifies the diagram modifications which must be made by the student if several of the measures inspired by graphical user interface design are to be effectively measured.

6.2.5.4 The need for students to adapt their solutions

Section 3.3 emphasised that a key benefit of the DATsys framework and the Theseus student diagram editor was the ability to allow the student to draw their solution onto a canvas in an interactive and intuitive way. It would have been possible for students to specify their diagram solution using other means, for example a proprietary text-based notation entered through a text editor, but this would have created an extra layer of abstraction between the student and the solution, hence unnecessarily hindering the learning process. Section 3.1.2.1 described the Kassandra system, where students were indeed expected to adapt their solutions to the requirements of the marking system.

It is clear that the level of disruption to the student learning process is related to the amount of modification which the student is required to perform. The student learning process will also be impacted less if the modification can be understood easily by the student, rather than involving requirements which are not understood by the student and are viewed as "abstract".

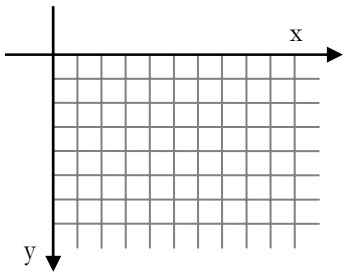

Figure 6.8: The co-ordinate system in DATsys diagram editors

In order for the aesthetic measures based upon graphical user interface design principles to be successfully calculated, it is necessary to define the *boundaries* of the student diagram. This is necessary if certain properties of the diagram, such as its

centre, are to be calculated. Within the DATsys framework, drawings are allowed an unbounded size, based upon a grid system of co-ordinates, as illustrated in figure 6.8. Students can use both scroll and zoom facilities to traverse large diagrams. One possible solution was to impose a canvas size upon the student for each exercise — this solution, however, is prescriptive to the student and fails to take into account that different diagram sizes may be required for different model solutions. The solution adopted, therefore, was to allow the student to specify the boundaries of their own diagrams by drawing a `BorderRectangle` around their solution, prior to submission. The extent of the student modifications is illustrated in figure 6.9.

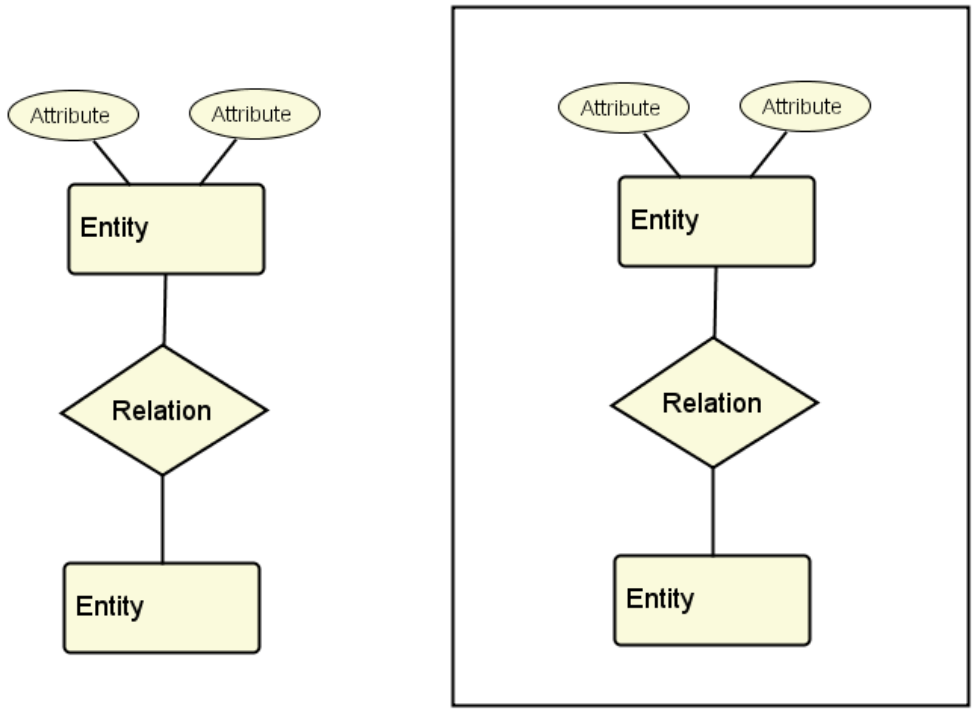

Figure 6.9: Original student solution and student solution with modification

In figure 6.9 an illustrative student solution is shown both before and after modification. For the student to make the modification, they must select the `BorderRectangle` tool from the library and use it to draw the rectangular boundaries of their solution. Only figures within the boundaries of the `BorderRectangle` will be considered by the marking process. This feature also allows students to leave reminder notes for their own purposes (for example, to remind them of why they chose features in their solution should they choose to view

their solution again at a later date), by simply placing the comments or other objects outside the boundaries of the `BorderRectangle`, where they will be ignored for marking purposes. Since this modification is simple, easy to understand theoretically and can be carried out within the student environment Theseus, it is clear that it is unlikely to impact upon the student learning process.

6.2.6 Structural measures

The design process for structural measures is precisely the same as for aesthetic measures. Structural measures must implement the `LayoutToolInterface` interface, must be based upon a criterion that can be expressed algorithmically such that a numeric value in the range 0 to 1 can be assigned to a student diagram to indicate student compliance and must not require the student to conduct complex modification to their solution for the purposes of "assisting" the algorithm to be successfully applied. Structural measures are invoked by the exercise marking scheme is the same way as aesthetic measures.

The primary difference between structural and aesthetic measures is pedagogic. Structural measures should measure some criterion that is domain-specific. Only exercises within the domain associated with the structural measure would invoke the measure in their marking scheme. The `MarkingLeafResult` returned by a structural measure is assigned to a different `MarkingCompositeResult` to those returned by aesthetic measures to allow the `PrioritiseTruncateTool` to distinguish between the two when prioritising and truncating student feedback.

Many domains will not require structural measures to be designed and implemented since there may be no domain-specific layout rules, allowing the layout of student diagrams within the educational domain to be assessed adequately by the aesthetic measures alone. Facility for structural measures is provided as an extension point to allow the layout of student diagrams in non-typical educational diagram domains to be addressed by educators and developers at a later date. If a new educational diagram domain does not require structural measures to be assessed, then development effort has been successfully minimised. If an educational domain requires structural measures, then the task of the educator is to specify one or more structural measures which can be implemented by the developer. The developer creates one class for each structural measure, which must be located in the

`layout.structural` package and which must implement the `LayoutToolInterface` interface. The `mark` method of the class is then invoked within the marking scheme of exercises within the educational diagram domain.

6.2.7 Summary

This section outlined the specific design decisions made to allow the implementation of the design outlined in section 6.1.2.1 to occur. The design decisions were linked to the detailed requirements and a hierarchy of packages and classes was defined. The interface which all aesthetic and structural measures must implement was defined and the design of the aesthetic measures was illustrated at length. Finally, the design similarity of structural and aesthetic measures was explained and the difference in usage emphasised. Section 6.3 outlines the specific design decisions made to allow the assessment of solutions with mutually exclusive alternate solution cases to occur.

6.3 Assessing solutions with mutually exclusive alternate solution cases: resolving the design issues

Section 6.1.2.2 outlined the approach to assessing solutions with mutually exclusive alternate solution cases. The approach is based upon identifying the common and uncommon elements within the acceptable model solutions. Features tests based around these elements are then constructed in features test cases. The 0^{th} features test case contains all features test cases based around the common elements, while subsequent features test cases are based around those uncommon elements present in a model solution. Therefore, if x model solutions have been designated acceptable by the educator, then $x+1$ features test cases will be required.

Although the functionality introduced by this extension is key to allowing the formative assessment of student coursework in free-form, diagram-based domains, the design and implementation process for this extension was the least demanding of the three extensions discussed in this work. The design is able to build upon existing functionality within the CourseMarker marking system. This section demonstrates the link between the design decisions and the detailed requirements for the extension presented in section 5.2.2. The process of implementing the features test cases using the generic `DiagramFeaturesTool` and invoking the test cases from within the marking scheme is described. The key process of identifying suitable *harbingers*

within each alternate model solution, and defining a distinction test based upon each *harbinger*, is outlined. Finally, possible methods of distinguishing between solution cases in order to prioritise feedback, after the marking process has been undertaken, are proposed and the decision to incorporate this stage of the marking process as a strategy within the `PrioritiseTruncateTool` is justified.

6.3.1 Linking the design to the requirements

Section 5.2.2 outlined the requirements for assessing solutions with mutually exclusive alternate solution cases. The way in which the alternate solution cases arise from the acceptability of multiple model solutions was outlined and the need to specify the common and uncommon features across the model solutions was explained and the requirements in the areas of CBA, educational diagrams and formative assessment were shown to arise from this situation.

The requirement for the exercise developer to specify the different solution cases is accomplished by allowing the specification of the common features, and the uncommon features tests associated with each model solution, to be achieved through the use of separate features test files. The specification of features tests has been common to CBA since the days of Ceilidh. Section 4.2 provided a description of features testing within the context of diagram-based CBA using CourseMarker. The features tests can be specified using the four generic features expressions *exist*, *exact*, *connection* and *exactConnection* which are implemented within the generic `DiagramFeaturesTool`. It is possible to extend the tool to introduce new features expressions as required, but this does not constitute part of the extension since the ability to create new marking tools is a historic ability of Ceilidh and CourseMarker [Sp06]. The novelty of the extension, and the focus here, lies in the ability to allow alternate *cases* of feature tests to be assessed, rather than in the specification of the features tests themselves. The `DiagramFeaturesTool` allows generic features tests, such as checking for the existence of nodes and the links between them, to be conducted in order to fulfil the requirement that the exercise developer should be able to specify assessment criteria across domains with a minimum of development effort and maximum consistency.

The extension is integrated into the marking and feedback system. Marking tools can be invoked by the exercise marking scheme. They return feedback by returning

marking results. Transparency from the student perspective is achieved by returning the marking result using the existing feedback delivery system. Integration with the `PrioritiseTruncateTool` allows raw feedback to be modified and truncated through the parameterisation of one tool, minimising development effort for the exercise developer and reducing the need for parallel tools and development hierarchies.

The extensions provide a basis for the assessment of a wide variety of educational diagram domains through the use of a generic mechanism to allow marking tools, such as the domain-independent `DiagramFeaturesTool`, to be executed on multiple occasions. This allows the features tests to be assessed independently of context and `MarkingLeafResult` objects to be composed for later examination within the context of the `PrioritiseTruncateTool`. The DiagramFeaturesTool allows educators to specify criteria for assessment as features tests. The system for specifying weighting to account for measures of unequal importance survives intact from previous CourseMarker features testing standards, but the usage is changed to reflect formative, rather than summative, assessment priorities.

The task of determining which model solution the student is attempting to attain is solved by the first strategy of the `PrioritiseTruncateTool`. This strategy is examined in section 6.4. Again, the design decision to integrate this functionality into the `PrioritiseTruncateTool` minimises development effort on the part of the exercise developer. It also facilitates the requirement to achieve compatibility between the feedback provided by this extension and the mechanism to allow the prioritisation and truncation of student feedback outlined in section 6.4.

6.3.2 A tool for generic features testing of diagrams

Section 6.3.1 explained that the `DiagramFeaturesTool` is not a key component of the extension and, furthermore, pointed out that any other CourseMarker marking tool which supported features-based testing in a manner compliant with the domain to be assessed could be substituted in place of the `DiagramFeaturesTool`. It is still necessary to provide a brief overview of the `DiagramFeaturesTool`, however, for several key reasons. Firstly, the `DiagramFeaturesTool` is generic. Therefore, it provides a basis for features testing to be conducted across a wide variety of educational diagram domains. It is in keeping with the design decisions taken

throughout this work that the commonality across domains be used to provide a *basis* for formative assessment across domains, while still allowing future flexibility by allowing extensions to be made by developers. Secondly, providing a platform for features testing is an essential prerequisite if the assessment of alternate features testing cases is to occur.

The `DiagramFeaturesTool` is based closely upon the `EntityRelationshipTool` used in the initial experiment described in Chapter 4. However, where the `EntityRelationshipTool` was constructed on an *ad hoc* basis, within a strict time frame, the `DiagramFeaturesTool` benefits from a clear design perspective which is intended to guide developers in the process of creating their own features testing tools, should this be required in the future.

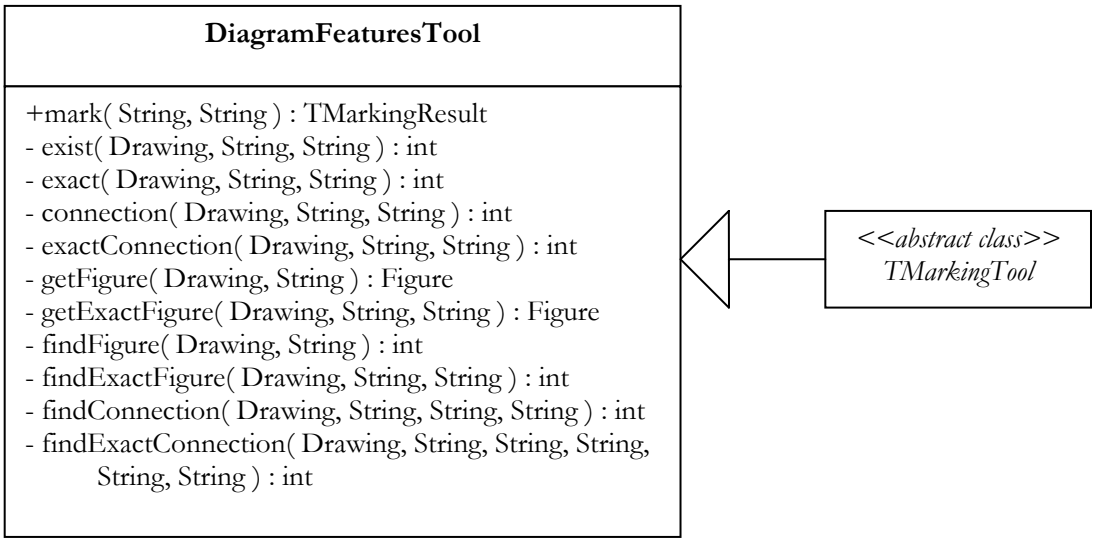

Figure 6.10: The DiagramFeaturesTool

The `DiagramFeaturesTool` extends the abstract class `TMarkingTool`. Several utility methods are present: `findFigure` returns the number of figures within a drawing which match the given figure name, `getFigure` returns the first figure which matches the given figure name, whilst `findExactFigure` and `getExactFigure` perform analogous functions based upon both a figure name and displayed text. `findConnection` and `findExactConnection` return the connection lines specified by name, and name and display text, respectively. Methods `exist`, `exact`, `connection` and `exactConnection` generate an enumeration of figures based upon the student diagram and return the number of

times the specified condition was matched. Finally, the public method `mark` acts to draw the functionality together. The String of the features test is parsed, with relevant information stored in variables. A case statement is used to call `exist`, `exact`, `connection` and `exactConnection`, dependent upon context and then to check if the feature test has been met, based upon the five accepted operators described in section 4.2. Based upon this, a new `TMarkingResult` is created and returned. Figure 6.10 shows the design of the `DiagramFeaturesTool`.

The `DiagramFeaturesTool` is invoked by the exercise marking scheme. The features tests are as described in section 6.1.2.2.

6.3.3 Designing the process of assessment for mutually exclusive solution cases

The assessment of student solutions in which multiple model solutions are deemed acceptable by the educator is a two stage process. The first stage involves assessing each of the features tests cases using a suitable marking tool such as the `DiagramFeaturesTool`. The second stage involves using a strategy to decide which of the model solutions the student is attempting to achieve and modifying the feedback accordingly. This section examines the first stage of this process. The second stage of the process is discussed in section 6.3.4, with the resultant design decisions being used as a `SolutionCaseStrategy` in section 6.4.

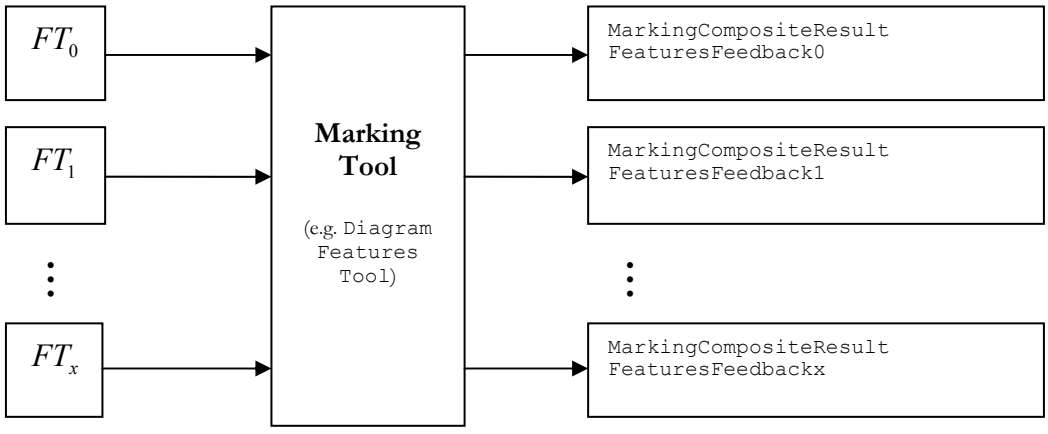

Figure 6.11: Marking multiple features test cases

Figure 6.11 illustrates the process of assessing multiple feature test cases. The marking tool, for example the `DiagramFeaturesTool`, is invoked repeatedly by the exercise marking scheme. A new MarkingCompositeResult is generated for each

features test case. Within each `MarkingCompositeResult`, a `MarkingLeafResult` node is used to store the mark, weight, description and feedback returned by the marking tool for each individual features test within the test case. The description of each `MarkingCompositeResult` is set to allow the common features test set, and each of the mutually exclusive solution cases, to be identified when the process of comparing the results generated by each of the solution cases is undertaken by the `PrioritiseTruncateTool`.

6.3.4 Harbingers and the distinction test

Section 6.1.2.2 has already discussed the role of *harbingers*, solution elements or combinations of elements which exist in only one model solution. *Harbingers* are used to construct the distinction test for each mutually exclusive solution case. The distinction test is a features test which should only succeed if the *harbinger* elements are found. This technique provides valuable assistance in the process of ascertaining which of the model solutions the student is attempting to attain. Indeed, some strategies to distinguish between solution cases may rely entirely upon the detection of *harbingers* through the distinction test.

The relative importance of features tests is generally indicated through the system of weighting. The distinction test, however, may or may not be educationally important to the student learning process. It is, instead, useful because it is useful to the features marking process within the context of assessing mutually exclusive solution cases. For this reason, the identification of the distinction test is carried out using a mechanism unconnected with the weighting system. In each mutually exclusive solution case, the distinction test is always the first features tests within the test set. This convention is carried through the marking tool; the feedback generated by the distinction test will always be held by the first `MarkingLeafResult` held within the `MarkingComposite` result for the mutually exclusive test case. Components of the `DiagramFeaturesTool` which implement the `SolutionCaseStrategy` may choose to use this information when deciding between solution cases. The common features tests, FT_0, do not have associated *harbinger* elements and so a distinction test is inappropriate.

6.3.5 Strategies for distinguishing between mutually exclusive solution cases

Given the disparity between educational diagram domains, it is unrealistic to expect that any one strategy can be successful in a generic way with regard to the assessment of mutually exclusive solution cases. This work has assumed that the fact that multiple model solutions may be feasible as a response to a given problem specification indicates that multiple conventions within the domain may be applied to solve the problem in varying ways, or that conventions may be inconsistently understood or applied within the domain in general. Indeed, since educational diagram domains attempt to teach principles to design problems to which there exists no single, deterministic solution, then this problem is likely to be permanent. If the occurrence of multiple model solutions is due to disparity within a domain, then, similarly, it is unrealistic to expect that similarity in the learning process *across* domains can be achieved.

Expert educators rely on domain knowledge to ascertain which model solution a student was attempting to construct. Within a CBA context, therefore, it is necessary to allow the strategy to distinguish between mutually exclusive solution cases to be determined by an expert.

However, the approach of this work has been to provide a *basis* for assessment, while allowing extension by subsequent developers. In this case, therefore, the following, simple, algorithm will be used as the basis for distinction:

- IF one distinction test succeeds (returns a mark > 0) AND all others fail (return 0) then identify the solution case associated with the successful distinction test;

- ELSE identify the distinction test with the highest average mark for features tests overall;

The concrete design of this strategy will be expanded in section 6.4, within the context of designing the `PrioritiseTruncateTool`.

6.3.6 Summary

This section outlined the specific design decisions made to allow the implementation of the design outlined in section 6.1.2.2 to occur. The design decisions were linked to the detailed requirements. The design is able to build upon existing functionality within the CourseMarker courseware. The design of `DiagramFeaturesTool` tool, for the features testing of diagram in a domain-independent way, was demonstrated and the process of repeatedly invoking the tool through the exercise marking scheme, subsequently storing the feedback results in a separate `MarkingCompositeResult` for each features test case, was outlined. The decision to incorporate the process of distinguishing between mutually exclusive solution cases into the `PrioritiseTruncateTool` was justified and the importance of *harbingers* and the distinction test to the approach was emphasised. It is unrealistic to expect any one strategy for distinguishing between mutually exclusive solution cases to be successful across educational diagram domains, so a basic strategy was proposed to provide a default basis on which to operate, while the potential for later expansion was emphasised. The process of implementing such a strategy is outlined in section 6.4, which documents the design of the `PrioritiseTruncateTool` for the prioritisation and truncation of student feedback.

6.4 Prioritising and truncating student feedback: resolving the design issues

Section 6.1.2.3 outlined the approach to the prioritisation and truncation of student feedback. This process was divided into four sub-tasks: examining the features feedback and deciding what course of action to take with regard to that provided by the mutually exclusive solution cases, prioritising all features test feedback, prioritising all layout feedback generated by the aesthetic and structural measures and, finally, truncating the feedback prior to its delivery to the student.

The design of the `PrioritiseTruncateTool` is based upon the *Strategy* design pattern. The tool acts as a context to four strategies, one for each of the sub-problems. Each strategy acts as an interface; concrete strategies to solving each of the four sub-problems must implement the interface associated with the sub-problem. The interfaces are used to define the rules associated with concrete strategies so that the

`PrioritiseTruncateTool` can operate smoothly using a variety of implemented strategies.

This section begins by linking the design decision to the detailed requirements for the extension set out in section 5.2.3. The section then establishes the design of the `PrioritiseTruncateTool`, followed by the design for each of the interfaces responsible for regulating the strategies for each sub-problem: respectively, those interfaces representing `SolutionCaseStrategy`, `FeaturesSortStrategy`, `AestheticsSortStrategy` and `TruncationStrategy` requirements.

6.4.1 Linking the design to the requirements

Section 5.2.3 outlined the requirements for prioritising and truncating feedback to students. The central criteria were to define a mechanism whereby the prioritisation followed by the truncation of the feedback could be achieved. Furthermore, flexibility for the educator and developer was required such that prioritisation and truncation could be configured to preference. The design achieves these requirements by outlining four sub-problems and allowing the strategy used to solve the sub-problems to be defined by the educator and implemented by the developer. The first three of the sub-problems are associated with the prioritisation of feedback comments, whilst the fourth sub-problem is associated with the truncation of the comments based upon prioritisation.

Requirements from the field of CBA were identified in terms of ensuring compatibility with the existing CourseMarker marking and feedback systems. In fact, the tool does not affect the functioning of the CourseMarker marking system in any way since it is designed to operate upon the feedback generated by the marking tools *after* their operation has been completed. The `PrioritiseTruncateTool` which acts as the context for the design acts transparently within the context of providing feedback. Within CourseMarker, the generation of feedback is associated with marking tools which are invoked through the exercise marking scheme. The delivery of the generated feedback to the student also occurs through the marking scheme. The `PrioritiseTruncateTool` is invoked *between* these two actions. The new course of events sees the feedback being generated, modified by the `PrioritiseTruncateTool` and then returned to the student by CourseMarker in the conventional manner.

6.4.2 The PrioritiseTruncateTool

The `PrioritiseTruncateTool` acts as the context for each of the strategies which represent the four sub-problems. The `PrioritiseTruncateTool` contains one method, `streamline`, which accesses the student feedback object together with objects representing concrete implementations of each of the four strategies, through parameterisation. The design uses the approach of having the context (the `PrioritiseTruncateTool`) pass the data as parameters to each of the strategy operation since this keeps the context decoupled from each of the strategies. Concrete strategies extend the `SolutionCaseTool`, `FeaturesSortTool`, `AestheticsSortTool` and `TruncationTool` respectively. The relationships between these tools and the `SolutionCaseStrategy`, `FeaturesSortStrategy`, `AestheticsSortStrategy` and `TruncationStrategy` interfaces is outlined in section 6.4.3. The `streamline` method passes the feedback object to each concrete strategy in turn before finally returning the student feedback. The design for the `PrioritiseTruncateTool` is summarised in figure 6.12.

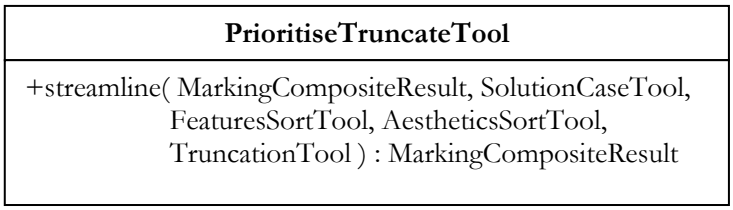

Figure 6.12: The PrioritiseTruncateTool

6.4.3 The strategy interfaces and abstract classes

The design of the strategy interfaces remains simple to allow maximum flexibility to the educator and developer. One design plan might have been to have the interfaces define specific methods for each of the operations which the strategy might reasonably expect to overcome (for example, the `SolutionCaseTool` might be expected to define methods to operate on marking results with only one solution case). However, this design would necessitate an inflexible design for the `PrioritiseTruncateTool`, which would be responsible for distinguishing between each plausible scenario based upon an examination of the marking result, and unnecessary dependency between the `PrioritiseTruncateTool`, which acts

as a context for the strategies, and the individual strategies themselves. The result would be an inflexible system with limited development flexibility.

To maximise flexibility, each interface requires that only one method be implemented: `modify`. The modify method is parameterised by the current feedback result, the `MarkingCompositeResult`, and returns a new `MarkingCompositeResult` representing the feedback after the concrete strategy has been applied. Each interface has an associated abstract class. These abstract classes must be extended by the concrete marking results in order that parameterisation of the `PrioritiseTruncateTool` may occur, thus enforcing implementation of the interfaces. The strategy interfaces, together with the associated abstract classes, are illustrated in figure 6.13.

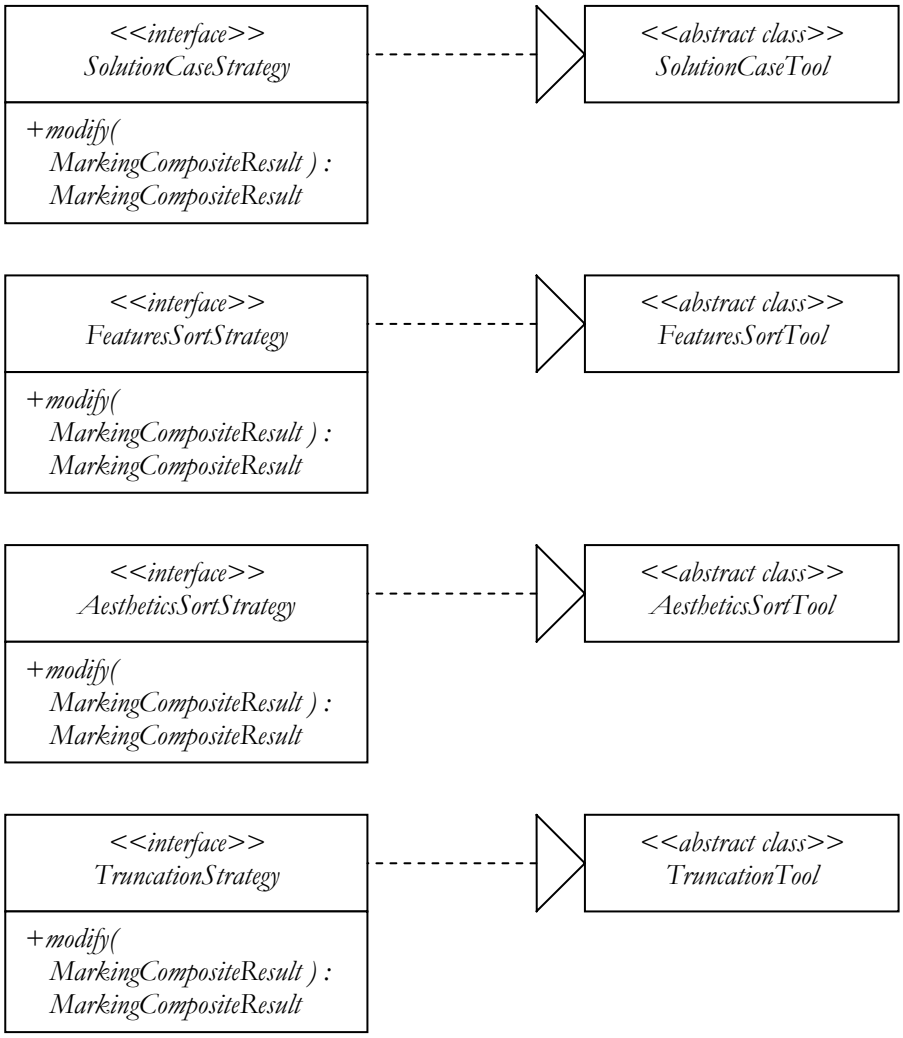

Figure 6.13: Strategy interfaces for the four sub-problems

6.4.4 Providing a basis

This work has consistently argued that, while flexibility in extension for educators and developers must be a high priority in the design of the extensions proposed by this work, it is also necessary to provide a *basis* for assessment to occur through the implementation of default assessment behaviour. Section 6.3.5 outlined a concrete strategy which could be used as the basis for solving the first sub-problem in the prioritisation and truncation of student feedback. It is, however, necessary to propose approaches to solving each of the three remaining sub-problems which can be used in implementation.

A concrete features sort strategy will be implemented which ignores the distinction between the common features test feedback and the feedback from the remaining mutually exclusive solution case. The two `MarkingCompositeResult` branches will be merged into one and sorted according to the prioritisation equation presented as equation 6.6.

A concrete aesthetics sort strategy will be implemented similarly. Feedback from aesthetic measures and structural measures will be combined and sorted by priority according to equation 6.6.

A concrete truncation strategy will be implemented where the highest priority n features feedback comments will be retained along with the highest priority m aesthetic layout feedback comments. The values n and m can be specified through parameterisation within the exercise marking scheme. All other feedback results will be pruned from the feedback tree.

Equation 6.6 relates the priority P, of a `MarkingLeafResult` x to its weight w and percentage mark m. Priority increases in proportion to both the weight of the comment and the level of error of the student.

$$P_x = w_x(100 - m_x)$$

Equation 6.6: Calculating the priority of a MarkingLeafResult

6.4.5 Summary

This section outlined the specific design decisions made to allow the implementation of the design outlined in section 6.1.2.3 to occur. The design decisions were linked to the detailed requirements. The design of the `PrioritiseTruncateTool` was defined, and a simple definition was provided and justified for each of the four sub-problems associated with the process of prioritising and truncating the feedback provided to students. A basis for implementing examples of concrete strategies to solve each of the four sub-problems was described.

6.5 Summary

This chapter outlined the design for the three extensions to the CBA courseware proposed by this work. Section 6.1 provided a high-level overview of the design for each of the extensions in words, and illustrated the high-level integration between the extensions. Section 6.2 outlined the design for the extension to allow the aesthetic layout of student diagrams to be assessed. A package structure was introduced for aesthetic and structural measures and the design of the measures themselves explained. A number of criteria from the fields of user interface design and graph layout were chosen to be implemented as aesthetic measures. Section 6.3 outlined the design for the extension to allow mutually exclusive solution cases to be assessed. A generic diagram features tool was described and the process of repeatedly invoking the features tool for each features test cases was described. The features test cases themselves were discussed and the importance of defining a distinction test for mutually exclusive solution cases based upon *harbingers* in the different model solution versions was emphasised. Section 6.4 outlined the design for the extension to undertake the prioritisation and truncation of student feedback. The process was divided into four sub-problems and a strategy interface was presented for each sub-problem. Concrete strategies for each of the four sub-problems were presented.

Chapter 7 provides an overview of implementation issues based upon the design decisions presented in this chapter, together with a summary of guidance for the use of teachers and educators, whose scope was identified in section 5.2.4.

Chapter 7

Issues in implementation and advice for educators and developers

Introduction

This chapter provides an overview of the issues arising from the implementation of the extensions and their integration into the CourseMarker architecture. The chapter also presents advice useful for developers and educators in the formative assessment of new domains and the setting of exercises, which builds upon existing theory and the documentation available for CourseMarker / DATsys.

Section 7.1 considers the implementation issues. The objectives of the implementation are outlined, software quality is considered and issues arising from integrating the extensions into the existing CourseMarker architecture are explained. A brief overview of the implementation of each of the extensions is provided. For each extension, the point of integration into CourseMarker is defined and reference is made to the design described in chapter 6.

Section 7.2 presents advice for developers and educators in developing CBA for formative assessment in new diagram-based domains and the setting of exercises. References to existing development materials, such as CourseMarker documentation, are provided and the differences between developing traditional materials for CourseMarker for summative assessment purposes and developing formative assessment materials for CourseMarker / DATsys which use the extensions provided as a result of this work are explained.

7.1 Implementation Issues

Chapter 6 documented the design process for each of the extensions and linked the design to the detailed requirements specification. This section outlines the key elements in the implementation of the design. Section 7.1.1 outlines the objectives in terms of the requirements which the implementation must fulfil. Section 7.1.2 discusses the issues arising from implementing the system as a set of extensions integrated into the CourseMarker architecture. Finally, sections 7.1.3 to 7.1.5 describe how the designs for each of the three extensions were implemented and the points of integration into CourseMarker.

7.1.1 Objectives

The main objective of this research is to investigate the feasibility and usefulness of automating the process of providing formative assessment in free-form, diagram-based domains using CBA courseware. Implementing the design is fundamental if the feasibility and usefulness is to be evaluated. The purpose of the implementation is to meet the following goals:

- To implement the extension to allow student diagrams to be assessed in terms of their aesthetic layout;
- To implement the extension to allow features testing to accommodate mutually exclusive solution features;
- To implement the extension to allow the prioritisation and truncation of student feedback;
- To address software quality issues;
- To integrate the extensions into CourseMarker, which provides a realistic, extensible framework for the full lifecyle of CBA.

The first three goals will be addressed in section 7.1.3 to 7.1.5 respectively.

The existing CourseMarker infrastructure has proven reliability, maintainability, portability and extensibility. The marking mechanism for CourseMarker is stable and integrated with course management, assessment material delivery, feedback and other facilities. Section 7.1.2 argues that integration into the CourseMarker architecture ensures that key software quality issues such as reliability, robustness, maintainability and portability are addressed automatically, as long as the integration is successful and the design of the extensions themselves is sound. For each of the extensions, therefore, it is necessary to demonstrate that the required functionality has been implemented, that the extension can be maintained and extended in accordance with software quality principles.

7.1.2 Integration into CourseMarker

CourseMarker provides an existing architecture, with a design emphasising explicit extension points, interfaces and standards to which components integrated into the architecture must conform. CourseMarker was implemented in Java 2, a language which is simple, object-oriented, distributed, interpreted, robust, secure, architectural-neutral, portable, high performance, multi-threaded and dynamic [GJS97]. Tsintsifas [Ta02] argued that choosing Java 2 as the implementation language would allow the development of a "*better deliverable*".

The design for the extensions outlined in chapter 6 made explicit its intention to integrate the extensions into the existing courseware architecture. Therefore, the design process was influenced by the need for integration from the outset. Integrating the extensions into CourseMarker has several major advantages. Key CBA concepts such as the storage of administration data, security of access and the delivery of materials and feedback, are already implemented as part of a proven design. Since these components have been successfully tested and repeatedly used in a live situation already, the design of the extensions was able to be simplified to the extensions themselves and their integration with the surrounding courseware. The other requirements of a CBA system, which were already implemented, could be removed from consideration. This is an advantage that carries through to the implementation stage.

Designing extensions to existing systems can be restrictive to the design process. In this case, however, the effect was minimised due to the fact that CourseMarker was designed with extensibility as a primary requirement.

For these reasons, the implementation of the extensions from their design was straightforward. Where implementation problems did occur, they were usually trivial.

7.1.3 Assessing the aesthetic layout of student diagrams: implementing the design

The hierarchy of packages described in section 6.2.2 is implemented within the package of marking tools, `com.ltr.cm.marking.tool`. The `LayoutToolInterface` therefore assumes the position at:

- com.ltr.cm.marking.tool.layout.*LayoutToolInterface*

The aesthetic and structural packages reside within the layout package. The 11 aesthetic measures, such as NonInterceptionTool and EquilibriumTool, are placed within the aesthetic package. The structural package is initially empty. The Figure interface, in com.ltr.daidalos.framework, imposes methods for all Figure objects which return the "size", "center" and other attributes of the object. These method calls are the basis behind the algorithms within the aesthetic measures, which are therefore implemented simply. Implementation closely follows design. Only two noteworthy issues arise.

Firstly, for the marking tools to be integrated successfully into CourseMarker, an associated marking command must be created for each. The marking command is responsible for retrieving the user's solution based upon the filename specified in the marking scheme and the project code of the user. The marking command is also responsible for dealing with errors, for example if the user's file cannot be found. The marking command calls the marking tool and returns the result to the feedback system. It is necessary to create a marking command for each marking tool. Implementation thus requires that 11 marking commands are created and housed in the com.ltr.cm.marking.cmd package. By convention, a marking command has a similar name to the associated marking tool. For example, the EquilibriumTool has the associated marking command EquilibriumCMD. The creation of marking commands is a standard process requiring no further design exertion; it can be accomplished most simply by making a copy of an existing marking command and editing both the name of the command and the name of the marking tool referenced within.

Secondly, the BorderRectangle necessary for the operation of many of the aesthetic measures, as illustrated in figure 6.9, is assumed to have been defined within Daidalos for each educational diagram domain to be assessed. BorderRectangle is treated as a reserved keyword. A simple utility method isValidFigure is introduced to the LayoutToolInterface which returns true if a figure is entirely contained within the BorderRectangle and false otherwise.

The process of invoking and parameterising the aesthetic and structural measures and storing the results within a `MarkingCompositeResult` object is illustrated in section 7.2.

7.1.4 Assessing solutions with mutually exclusive solution cases: implementing the design

The `DiagramFeaturesTool` for the generic features testing of diagrams is implemented within the package `com.ltr.cm.marking.tool`. An associated marking command, `DiagramFeaturesCMD`, is implemented within the package `com.ltr.cm.marking.cmd`. Implementation of the `DiagramFeaturesTool` is straightforward because the functionality was based upon the earlier `EntityRelationshipTool`, which had already been implemented. A clearer method structure was, however, imposed by the design of the tool.

Methods to facilitate a traversable enumeration of all figures within a drawing are imposed by the `Drawing` interface within the `com.ltr.daidalos.framework` package.

The process of repeatedly assessing the mutually exclusive solution cases is based upon repeatedly invoking the `DiagramFeaturesTool`, through the `DiagramFeaturesCMD`, within the exercise marking scheme. Therefore, since the distinction between solution cases is achieved by the `PrioritiseTruncateTool`, this stage of the implementation is the least demanding.

The process of repeatedly invoking and parameterising the `DiagramFeaturesTool` and storing the results within `MarkingCompositeResult` objects is illustrated in section 7.2.

7.1.5 Prioritising and truncating student feedback: implementing the design

The `PrioritiseTruncateTool` marking tool is implemented within a package `prioritisetruncate` located with the other marking tools at `com.ltr.cm.marking.tool`. The four simple interfaces `SolutionCaseStrategy`, `FeaturesSortStrategy`, `AestheticsSortStrategy` and `TruncationStrategy`, together with their associated abstract classes, are also implemented within

`com.ltr.cm.marking.tool.prioritisetruncate`; four further sub-packages `solutioncasestrategies`, `featuressortstrategies`, `aestheticssortstrategies` and `truncationstrategies` are introduced for the purpose of grouping together the implemented concrete strategies in a consistent way which is convenient for the domain developer.

Four concrete strategies are implemented. The reasons for this are three-fold. Firstly, the implementation is consistent with the approach of the work, which attempts to provide a basis for formative assessment through implementing appropriate default behaviour as well as providing a foundation for future extension. Secondly, the implementation of concrete strategies facilitates an analysis of the usefulness of the extension, a key requirement of the implementation. Thirdly, within a CBA context the most suitable way for a developer to create new components is by modifying existing ones. Therefore, the implemented concrete strategies provide a useful template for future expansion by developers.

The `DistinctionFirstSolutionCaseTool` extends `SolutionCaseTool` and is located within the `solutioncasestrategies` package. It is based upon the algorithm outlined in section 6.3.5.

The `MergeEqualFeaturesSortTool` extends `FeaturesSortTool` and is located within the `featuressortstrategies` package. The `MergeEqualAestheticsSortTool` extends `AestheticsSortTool` and is located within the `aestheticssortstrategies` package. The `PriorityBothTruncationTool` extends `TruncationTool` and is located within the `truncationstrategies` package. These three concrete strategies are based upon the algorithms described in section 6.4.4.

Associated marking commands were implemented for each of the concrete marking tools in the same way as in sections 7.1.2 and 7.1.3.

The process of invoking and parameterising the `PrioritiseTruncateTool` delivering the resulting feedback, which has been prioritised and truncated, to the student, is described in section 7.2.

7.1.6 Summary

Section 7.1 outlined the key elements in the implementation of the design. The objectives of the implementation were explained. The integration into the existing CourseMarker architecture was discussed and its implications for the implementation outlined. The design of the three extensions was influenced from the very beginning by the need to be integrated into the CourseMarker architecture. This facilitated a smooth implementation process, an overview of which was provided for each of the three extensions. Section 7.2 presents advice for developers and educators in developing CBA for formative assessment in new diagram-based domains and the setting of exercises.

7.2 Advice for developers and educators

Section 5.2.4 argued that guidance for developers and educators was an essential requirement if the implementation of the extensions was to result in successful formative assessment being carried out. Designing the assessment format to take full advantage of the capabilities of an automated assessment system is a prerequisite for successful assessment, a fact illustrated in section 3.1, where formative assessment examples utilising the same courseware were shown to vary in their level of success.

This section presents essential guidance for educators and developers. In doing so, the section achieves several objectives. Firstly, it demonstrates the mechanisms which allow educators and developers to use the extensions to deploy formative assessment exercises in a feasible and useful way. Secondly, it provides a documentation overview which illustrates the practical implementation of exercises using the extensions. In doing so, the practical integration between the extensions themselves, and between the extensions and the existing courseware infrastructure, is made explicit. Thirdly, it provides a useful overview to allow existing CourseMarker users to appreciate the new functionality available for formative assessment purposes for the purposes of migrating their exercises to make use of the extensions, where appropriate.

Section 7.1 provides guidance for developers. Documentation for the development of CourseMarker / DATsys exercises is already available and reference to this is made. The guidance to developers concentrates on those aspects of domain and exercise

development which are either changed or completely new when setting formative exercises using the extensions outlined within this work. Section 7.2 provides guidance for educators. The literature on formative assessment is plentiful and references to relevant introductory texts are provided. The section concentrates on the development of assessment materials which utilise the CourseMarker extensions to facilitate the achievement of formative assessment best practice. The section also notes the differences in conceptual assessment design between the formative assessment exercises and the previous, summative, CourseMarker exercise assessment schemes.

The roles of educators and developers have been previously defined. Conceptually, the educator develops assessment materials while the developer facilitates the necessary extensions to the courseware and may be responsible for translating the assessment materials into correct CourseMarker exercises. However, it is certainly useful for those in both roles to be *aware* of the material in its entirety, since co-ordination and mutual understanding across roles will best facilitate exercises which make the most effective use of the courseware environment.

7.2.1 Guidance for developers

7.2.1.1 Prerequisites

The process of setting up an exercise using CourseMarker is outlined in [Sp02]. The document provides an overview of the directory structure for CourseMarker exercises and summarises the files required at the course, unit and exercise levels of the structure. The structure of the necessary administration files is specified, such as `save.txt`, which is responsible for defining the student files retrieved by the server prior to the marking process, `setup.txt`, which is responsible for defining the files placed in the student directory when the exercise is set up, and so on. The definition of features for features testing is covered, including the Oracles notation for the features expressions used to define the search. The use of marking schemes expressed in Java is explained and a sample `mark.java` file is listed in full. The function of the various marking commands is explained. Finally the batch file `mrun.bat`, used to compile the marking scheme, is explained.

The use of Daidalos to author new diagram notations is a simple, intuitive process described in [Ta02]. It would be helpful for the developer to familiarise themselves with CourseMarker conventions before attempting to set exercises. This section explains the differences and extensions to the exercise format necessary to implement formative assessment exercises in diagram-based domains.

7.2.1.2 Expressing features testing regimes to assess mutually exclusive solution cases

Features marking continues to use the same format as in [Sp02]. Feature expressions vary depending upon the marking tool called; the simple, generic `DiagramFeaturesTool` supports four types of features expression (exist, exact, connection and exactConnection) with the same parameterisation as for common features tests. Features test cases are expressed in separate features marking files. The first stage of the process is to express the common features tests in the file `[ExerciseName].ft0`. The second stage is to express the mutually exclusive alternate cases in subsequently numbered features files. The first features test within each file should examine the distinction test. The third stage is to assess all cases in turn by invoking the marking tool within the exercise mark scheme.

mark.ft0:

```
5 : exact CircleNode A : Feedback1 : Feedback2
4 : exact CircleNode B : Feedback1 : Feedback2
4: exactConnection Link CircleNode A CircleNode B : Feedback1 : Feedback2
```

mark.ft1:

```
5 : exact CircleNode C : Feedback1 : Feedback2
7 : exist SquareNode==0 : Feedback1 : Feedback2
```

mark.ft2:

```
5 : exact SquareNode E : Feedback1 : Feedback2
5 : exactConnection Link CircleNode B SquareNode E : Feedback1 : Feedback2
4 : exact SquareNode F : Feedback1 : Feedback2
3 : exact SquareNode G : Feedback1 : Feedback2
```

Figure 7.1: Features tests organised into cases

Figure 7.1 outlines a very simple example of three features files which might be used to assess the exemplar problem in figure 5.1. The domain has three types of figure: `CircleNode`, `SquareNode` and `Link`.

The `mark.ft0` features file contains features tests examining those elements common to all model solutions, while the `mark.ft1` and `mark.ft2` files represent mutually exclusive solution cases. The first features tests within `mark.ft1` and `mark.ft2` — *`exact CircleNode C`* and *`exact SquareNode E`* respectively — denote the distinction test for each case.

The weight and feedback should be determined by the educator according to the guidance presented in section 7.2.2. Feedback assists the student learning process within a domain and therefore, by definition, relies upon domain knowledge to be useful to the student. As an example, the unsuccessful feedback (Feedback2) from `mark.ft1` could explain to the student why the type of solution represented by that case precludes the existence of any `SquareNode` nodes.

The way the features test cases are assessed through invocation of the marking tool by the marking scheme is examined in section 7.2.1.5.

7.2.1.3 Layout tools

All layout tools must implement the `LayoutToolInterface` interface and are placed in either the package `aesthetic` or `structural` depending upon the nature of the tool. Layout tools must implement the method mark, which takes the student drawing, relative weight and leniency value as arguments and returns a `MarkingLeafResult`. The majority of methods related to the calculations of layout tools are located in the `Drawing` and `Figure` interfaces of DATsys. Method calls can return the co-ordinates of the location of the centre, width, height etc. of a figure. Enumerating the figures within a drawing object for the purposes of traversal is a repetitive task which varies little between diagram marking tools. As with all CourseMarker exercise components, the best way to implement a new layout tool is to copy an existing tool and make necessary modifications, rather than attempting to implement "from scratch". In most cases, the mathematical formulae for layout measures can be translated directly into algorithms for implementation into layout marking tools.

Once the raw score from the algorithm is returned, scaling should be undertaken using the leniency value. The new mark value is used to parameterise the constructor

of `MarkingLeafResult`, along with the description and feedback returned by the tool, and the weight, which is unchanged during the marking process.

The process of invocating and parameterising the layout tools is examined further in section 7.2.1.5.

7.2.1.4 Prioritisation and truncation strategies

Like layout tools, prioritisation and truncation strategy tools implement specific interface methods, deduce their input data in a standard way from the provided parameters and return an object, in this case a `MarkingCompositeResult`, to conform to the interface. Similarly, the most straightforward way to implement a new prioritisation or truncation strategy tool is to copy an existing tool of the correct type and modify the central algorithm to conform to the strategy outlined by the educator.

There are four types of prioritisation and truncation strategy tools which may be implemented.

Solution Case Strategy marking tools extend the `SolutionCaseTool` class and must decide how to distinguish between mutually exclusive alternate solution case feedback and prune the feedback tree accordingly. `MarkingCompositeResult` objects containing feedback from the common feature case will contain the substring "common" in their description, whilst those representing mutually exclusive alternate solution cases will have the substring "exclusive*x*", where *x* was the solution case number.

Features Sort Strategy marking tools extend the `FeaturesSortTool` class and must implement an algorithm to apply criteria for sorting to features test feedback for the purposes of prioritising feedback leaf nodes, and deciding how to prioritise features results from the common and mutually exclusive cases relative to each other.

Aesthetics Sort Strategy marking tools must apply an algorithm for sorting and criteria for prioritisation, this time between the feedback from aesthetic and structural marking tools. Aesthetics Sort Strategy marking tools extend the `AestheticsSortTool` class.

Truncation Strategy marking tools must implement an algorithm to truncate the overall `MarkingCompositeResult` in order that feedback can be returned to the user. Truncation Strategy marking tools extend the `TruncationTool` class.

Prioritisation and truncation strategy tools must be placed in the appropriate package within `com.ltr.cm.marking.tool.prioritisetruncate`. Solution Case Strategy marking tools are located within the `solutioncasestrategies` package, Features Sort Strategy marking tools within the `featuressortstrategies` package, Aesthetics Sort Strategy marking tools within the `aestheticssortstrategies` package and Truncation Strategy marking tools within the `truncationstrategies` package.

The process of invocating and parameterising the `PrioritiseTruncateTool` with marking tools from each strategy area is examined further in section 7.2.1.5.

7.2.1.5 The marking scheme

Figure 7.2 shows an example of an exercise marking scheme. Twelve points of interest are noted on the figure for reference.

The CourseMarker course structure hierarchically stores exercises in directories which conceptually represent courses, units and exercises. The package structure reflects the directory structure of the course. Marking commands and tools must be imported to make them available to the marking scheme class (point 1). The `markExercise()` method (point 2) returns a `TMarkingResult` object; this is the point of integration to CourseMarker's feedback delivery facilities, since it is the `TMarkingResult` object which is used to populate the graphical feedback tree representation. String and integer range boundaries can be specified to configure the look of the feedback tree to the student.

```
package MyCourse.MyUnitOfExercises.SimpleExercise;

import com.ltr.cm.marking.*;
import com.ltr.cm.marking.cmd.*;
```

```
import
    com.ltr.cm.marking.tool.prioritisetruncate.solutioncasestrategies.*;
import
    com.ltr.cm.marking.tool.prioritisetruncate.featuressortstrategies.*;
import
    com.ltr.cm.marking.tool.prioritisetruncate.aestheticssortstrategies.*;
import com.ltr.cm.marking.tool.prioritisetruncate.truncationstrategies.*;
import com.ltr.cm.marking.tool.layout.aesthetics.*;

public class mark extends TBaseMarkScheme {

public TMarkingResult markExercise() {

String[] strRange = {"Rotten", "Poor", "Good", "Excellent"};
int[]    intRange = {  40    ,   50  ,   80  ,     100    };


TMarkingResult am1 = execute(new NonIntersectionCMD( "Simple.draw", 4,
    0.5 );
am1.setWeight(4);
am1.setFeedbackRange(strRange, intRange);

TMarkingResult am2 = execute(new EquilibriumCMD( "Simple.draw", 3, 0.21
    );
am2.setWeight(6);
am2.setFeedbackRange(strRange, intRange);

MarkingCompositeResult amcr = new MarkingCompositeResult("Aesthetics
    Measures");
amcr.addChild(am1);
amcr.addChild(am2);

TMarkingResult feat0 = execute(new DiagramFeaturesCMD( "Simple.draw",
    "SimpleExercise.ft0" );
feat0.setFeedbackRange(strRange, intRange);

MarkingCompositeResult f0mcr = new MarkingCompositeResult("Common
    Features");
f0mcr.addChild(feat0);

TMarkingResult feat1 = execute(new DiagramFeaturesCMD( "Simple.draw",
    "SimpleExercise.ft1" );
feat1.setFeedbackRange(strRange, intRange);

MarkingCompositeResult f1mcr = new MarkingCompositeResult("Exclusive1");
f1mcr.addChild(feat1);

MarkingCompositeResult rawtree = new MarkingCompositeResult("Main");

rawtree.addChild(amcr);
rawtree.addChild(f0mcr);
rawtree.addChild(f1mcr);

MarkingCompositeResult feedback = new PrioritiseTruncateTool( new
    DistinctionFirstSolutionCaseTool(), new MergeEqualFeaturesSortTool(),
    new MergeEqualAestheticsSortTool(), new PriorityBothTruncateTool() );

return feedback;
}}
```

1 2 3 4 5 6 7 8 9 10 11 12

Figure 7.2: A simple marking scheme for a formative exercise

The `NonIntersectionCMD` is called (point 3) and the feedback generated by the associated NonIntersectionTool is given the variable name `am1`. The

`NonIntersectionTool` returns a `MarkingCompositeResult`, which is an implementation of the `TMarkingResult` interface. Various properties of the `TMarkingResult`, such as the weight, can still be manually set, if required.

The `EquilibriumCMD` is invoked using the same mechanism (point 4); in a complete marking scheme all eleven aesthetic measures, plus any required structural measures, would be invoked in this way. Here they are omitted for the sake of brevity. Next, a new `MarkingCompositeResult` is generated to store the feedback generated by all aesthetic measures (point 5). Structural measures are dealt with in the same way as the aesthetic measures. Each structural measure is invoked and its marking result is stored as a `TMarkingResult`. Once all structural measures have returned their results, they are all added as child nodes to a `MarkingCompositeResult` for later prioritisation and truncation.

The next stages demonstrate the marking of features test cases. The `DiagramFeaturesTool` is invoked (point 6) to carry out the features assessment upon the common features tests expressed in the file `SimpleExercise.ft0`. A composite marking result for common features is created with common features feedback as its child nodes (point 7). The process is repeated for the first mutually exclusive features case (points 8 and 9). Generally, a marking scheme would invoke *at least 2* mutually exclusive features cases; if only one model solution is acceptable, then the use of mutually exclusive features testing is superfluous.

A composite marking result to encompass all feedback is generated (point 10), with the composite marking results for the aesthetic and structural measures (in this case, there are no structural measures) and the features test cases being added as children. The `PrioritiseTruncateTool` is then invoked, parameterised by four concrete strategies (point 11). The `PrioritiseTruncateTool` utilises each of the strategies in turn to generate a new `MarkingCompositeResult`. Finally (point 12), the new `MarkingCompositeResult` is returned as feedback.

7.2.2 Guidance for developers

7.2.2.1 Prerequisites

Conceiving and constructing assessment materials is a non-trivial task. Assessment materials for formative assessment benefit from a great potential for re-use across

many academic sessions since plagiarism between students is not an issue. Unfortunately, due to the care which must be taken in defining the feedback and the amount of time necessary to consider alternative model solutions, formative assessment materials require considerable development time. Resource-savings are therefore optimised by creating exercises which can be re-used.

Formative assessment is intended to assist student learning. Therefore, exercises based upon logical application of domain principles are to be preferred over deliberately misleading questions, especially in the early stage of a course of exercises. A briefing on formative assessment principles is provided in [Kp01]. The primary deliverable associated with formative assessment is feedback, rather than assessment marks or grades. For this reason, great care must be taken in constructing the feedback for the exercises. The nature of feedback for formative assessment is discussed in [JMM+04]. Good formative assessment using CBA courseware depends upon a successful interaction between the assessment materials and the courseware itself. Thus, an examination of CBA exercises using CourseMarker and the guidance for developers provided in section 7.2.1 is likely to prove useful. Knowledge of the way in which features testing operates in existing CBA exercises is a pre-requisite to specifying features testing using the mutually exclusive features test cases allowed by the extensions.

This section outlines the issues in constructing assessment materials to utilise the formative assessment potential offered by the new extensions described in this work.

7.2.2.2 Identifying harbingers and specifying distinction tests

Given an assessment specification to which there is more than one possible model solution, the first task is to identify those elements which are common to all model solutions and to construct features tests which assess the solution based upon those elements, or combinations of those elements, alone.

For each model solution, it is next necessary to consider those elements which are uncommon. It is necessary to emphasise that, although features tests search for features expressions which, in turn, are developed around the idea of searching for desired elements, the precise nature of the relationship between features tests and elements varies across both domains and the preferences of educators.

Features tests within the mutually exclusive solution cases will, therefore, be based upon testing for the presence (or absence) of combinations of both common and uncommon elements in the model solutions which allows the pedagogic understanding of the student to be assessed and meaningful advice, in the form of feedback, to be given.

An important features test, which must be defined for each mutually exclusive alternate solution case, is the distinction test, which may be used to determine which of the model solutions is most related to the attempt of the student.

The initial effort of the educator should be directed towards identifying a *perfect harbinger* within each of the model solutions. A *perfect harbinger* is an element (or, likely, a combination of elements) which defines the key difference which distinguishes the model solution. Although other elements within the model solution may be uncommon, it is likely that they could have occurred as a *consequence* of the choice of elements in the *perfect harbinger*. The task of the educator is, then, to create a distinction test based upon the *perfect harbinger* which returns helpful, domain-specific feedback based upon the reasons why the model solution is distinguished.

All mutually exclusive solution cases must, by definition, contain an uncommon element (or combination of elements). If a model solution contains no *perfect harbinger*, then an element or combination of elements must be used as the basis for the distinction test which fulfils the minimum criterion of being *unique* to the model solution. This still allows the system to make a definite distinction between alternative model solutions. It is, however, less ideal from pedagogic standpoint and is thus referred to as an *imperfect harbinger*. The construction of useful feedback may prove a more difficult task for the educator when using *imperfect harbingers*.

7.2.2.3 The weighting system

Unlike most of the advice which is summarised in this section, the system of weighting might be most easily understood by those with the least experience in setting CourseMarker exercises. Weights are attached to features tests using the same mechanism as for standard CourseMarker exercises. However, their meaning within formative assessment exercises changes.

In summative assessment exercises, weight was assigned to features tests to represent the relative weight of the features test in *assigning grades*. Therefore, the highest weights were awarded to the most difficult features tests in order to designate credit fairly to the more able students. In formative assessment exercises, however, the weights refer to the *priority* of the feedback. Therefore, the highest weights are awarded to the most fundamental (usually the easiest) features tests, since the most fundamental aspects of a student solution must be corrected first, before moving on to the more advanced features of the student solution at subsequent stages, when the student has successfully attained the basics.

7.2.2.4 Configuring and specifying aesthetic and structural measures

The key difference between aesthetic and structural measures is that the former are domain-independent, while the latter are domain-specific. New aesthetic measures will be implemented rarely where a measurable criterion can be demonstrated to have domain-independent assessment validity. The need for structural measures must, however, be examined when each new domain is to be assessed for the first time. Many domains will require the specification of no structural measures. In this case only the prioritisation of the aesthetic measures will need to be considered. If structural measures are required then these, together with their priority relative to the aesthetic measures, will need to be specified.

A measure is based upon any algorithm which, when applied to a student drawing, produces a numeric value to indicate success (or compliance with the criterion). A variety of aesthetic measures have already been designed and implemented. Suitable structural measures depend upon the properties of the domain to be assessed, and their representation as an algorithm. For example, given a domain in which all nodes of type *b* must be located exactly vertically underneath a corresponding node of type *a*, a structural measure could be defined as the proportion of nodes of type *b* which do, in fact, reside vertically underneath an *a* node. Suitable properties to be examined include the positions of nodes, since their centres and dimensions can be determined by the marking tools.

Configuring the marking tools involves first specifying the leniency of the tool. The leniency lowers the threshold at which good feedback is returned to the student by a measure. It is represented as a percentage. A good method of determining suitable

leniency for an exercise is to use the measures, configured to have no leniency, to assess the model solutions, in order to develop an idea of what is realistically possible within context.

Prioritising the marking tools is a straightforward concept, but great care must be taken to achieve a useful balance. Priority is determined by using an integer. Priorities are *relative*; the numbers could represent percentages in the mind of the educator, but any system may be used so long as consistency is maintained throughout. It is especially important not to weight measures too disproportionately. If disproportionate weighting is applied, then certain measures may never qualify to return feedback to the student since their priority would be constantly overridden.

7.2.2.5 Specifying and configuring prioritisation and truncation strategies

Once the assessment of aesthetic and structural measures, together with that of the features cases, has been achieved, a raw tree of *all* feedback comments is generated by the system. At this point it is necessary to prioritise the feedback and truncate the tree to leave only that feedback which is most important for the purposes of the student. This process is accomplished through a four-stage process. In the first stage, the most relevant mutually exclusive solution case is determined. The feedback associated with all other cases may be discarded at this stage. In the second stage, the priority of the feedback generated by the features testing is determined and sorting carried out. The feedback nodes from the common features test case may be either merged with those from the most relevant mutually exclusive solution case, or kept separate, dependent upon context. In the third stage, priority of aesthetic and structural measure feedback is determined and the feedback nodes sorted. In the fourth and final stage, the resultant feedback tree is pruned according to a method of truncation.

At each stage, it is necessary to visualise the process of prioritising and truncating the tool as if completing the task by hand. Careful examination of the processes used, within the context of the domain, to choose feedback to return to the student in a manual process may result in the visualisation of a suitable algorithm. Examination of previously implemented algorithms may be a further source of inspiration (or even reveal suitability for simple re-use).

When specifying an algorithm it will reduce future development effort if algorithms are specified generally and allowed to be parameterised for the purposes of configuration. For example, consider a simple algorithm to remove all feedback except the 2 highest priority features feedback nodes. It would be better to specify an algorithm which removes all feedback except the *n* highest priority features feedback nodes, and specify n = 2 through parameterisation. This increases the scope for re-use in future contexts and maximises the resource-savings associated with the courseware.

7.2.2.6 Writing good feedback comments

Specifying good feedback comments is a non-trivial undertaking and likely to consume a large proportion of an educator's development time. [JMM+04] provides a useful overview of feedback comments and their relationship to conceptual frameworks of student-centred learning. In general, good CBA practice encourages student research after each submission. Such student research can be encouraged through the linking of feedback comments to assessment materials. One solution to this problem is to develop extensive teaching materials which can be directly referenced by the feedback. The student can then refer to the materials directly. This approach has the disadvantage that very large amounts of time and resources are required to develop the materials. A successful mechanism for the encouragement of student research which is more common is to provide references to appropriate texts which are available to the student online or using institutional infrastructure such as library facilities.

The feedback comment itself should be of a positive, motivational nature. The feedback comment should emphasise good practice related to the shortcoming within the student solution which caused the comment to be returned, rather than stating the failing of the student solution directly. An example of a scenario would be a student diagram in which a required connection line between two existing nodes is absent from the student solution. Suitable feedback would explain the options for connecting nodes of the type in question and provide a suitable reference to further, relevant, information. Less suitable feedback would state the missing link to the student.

7.2.3 Summary

Section 7.2 provided essential guidance for both educators and developers. Section 7.2.1 provided an overview of the issues arising in the development of CBA assessment materials for formative assessment using the courseware and demonstrated the way in which implementation of the exercises would occur in practice. Useful references were provided to existing documentation. The implementation and configuration of features testing regimes, layout tools and prioritisation and truncation strategies was examined. Finally, a simple marking scheme was presented with a step-by-step explanation attached.

Section 7.2.2 provided an overview from the point of view of the educator. Knowledge prerequisites were indicated, and the topics of identifying *harbingers* and distinction tests, configuring and specifying aesthetic and structural measures and prioritisation and truncation strategies and the writing of good feedback comments were discussed.

7.3 Summary

This chapter provided an overview of the issues arising from the implementation of the extensions and their integration into the CourseMarker architecture, together with useful advice for developers and educators in the formative assessment of new domains and the setting of exercises. The aim of the implementation, to facilitate research into the feasibility and usefulness of automating the formative assessment process, within diagram-based domains, using CBA courseware was discussed. To this end, the implementation itself was described, and the way in which the implementation can be used by a combination of educators and developers to produce assessable course exercises was discussed in detail.

Chapter 8

Use and evaluation

Introduction

This chapter argues that the development of the extensions and their integration into the existing CourseMarker courseware has resulted in a system which allows the formative assessment of diagram-based domains to be automated in a manner which is both feasible and useful.

Following the implementation overview presented in chapter 7, the purpose of this chapter is to illustrate the use of the system, discuss initial results in the development of formative, diagram-based CBA, evaluate the courseware from the perspectives of CBA, formative assessment and educational diagramming, evaluate the integration between the extensions and the existing architecture and discuss general conclusions with regard to the three research areas related to the work.

Formative exercises, utilising the implemented extensions, have been implemented in two domains. These exercises were evaluated by being provided to students. Results from the exercises were available for scrutiny and great attention was paid to comments from students and to the responses to questionnaires.

Section 8.1 outlines the objectives of the chapter. Section 8.2 provides an overview of the exercises in terms of their development process, use by students and evaluation. Sections 8.3 to 8.5 evaluate each of the extensions in turn with respect to CBA, formative assessment and educational diagramming considerations. Section 8.6 draws together general conclusions in order to argue that the central objective of the work has been met.

8.1 Objectives

This chapter has two main objectives:

- To evaluate the implemented extensions in terms of criteria defined by the three research areas of CBA, formative assessment and educational diagramming and to determine the effectiveness of their integration into existing courseware;
- To test formative, diagram-based CBA in practice and draw initial conclusions about the benefits such an approach brings.

A further objective is to reflect upon the feasibility and usefulness of conducting formative computer-based assessment in diagram-based domains.

The objective of the three extensions is to enhance the functionality of the CourseMarker / DATsys courseware to take into account the shortcomings of the existing system with regard to conducting formative exercises, which were identified during the initial phase of research, summarised in chapter 4, and by applying detailed consideration of the requirements, as demonstrated in chapter 5. By integrating the extensions into the existing CourseMarker / DATsys courseware it is possible to take advantages of existing features, such as the ability to define representations for new diagram domains without programming, the ability to specify customised student diagram editors and a stable, reliable platform for delivering CBA across a departmental network and collecting administrative data.

8.2 Examples of formative, computer-based assessment exercises in diagram-based domains

8.2.1 The process of exercise creation

The authoring of a formative, diagram-based CBA exercise, based upon a problem specification, involves a series of stages.

Firstly, the Daidalos editor must be used to build a tool library which represents the domain notation, including the nodes and connection lines associated with the domain and the Border Tool. This stage must be undertaken once for each new domain.

Secondly, the marking tools must be developed and configured on a domain-specific basis. The `DiagramFeaturesTool` can be used to assess diagram features using several generic operators, but if more specific functionality is required then this tool must be extended or a new, suitable tool developed. Evaluation of the domain requirements must be used to indicate whether domain-specific structural measures need to be developed. In all cases, the relative weighting of the aesthetic layout measures must be considered. Finally, the prioritisation and truncation strategies must be decided. If existing prioritisation and truncation strategies are suitable within context, then simple parameterisation occurs. Otherwise, new prioritisation and truncation strategies must be developed.

Thirdly, the Ariadne editor must be used to build the individual exercises. A subset of the tools from the tool library is defined, application features are selected and configuration of exercise options is undertaken. Model solutions for the exercise are drawn on Ariadne's drawing canvas for later reference. Configuration of the marking tools on a per-exercise basis can be achieved through the text editors associated with Ariadne (or with a simple editor such as Notepad).

Once a domain has been defined and exercises developed, CourseMarker can be used to manage the full lifecycle of the CBA exercises. This involves the same stages for formative exercises developed using the extensions described in this work as for previous CourseMarker exercises, namely:

- The testing and deployment of the exercise using CourseMarker;
- The running of the exercise and the marking of student solutions;
- Exercise administration.

The administering of the exercise involves the collecting of student solutions, marking results and feedback for the purposes of evaluation of the results.

Formative assessment exercises in two domains were assessed using the courseware. The exercises were offered to students on a voluntary basis only for reasons of institutional administration. The authored exercise domains were in UML Class Diagrams and in UML Use Case Diagrams.

The task of authoring new exercise domains is very lengthy, but straightforward. The outcome benefits both students and educators and, furthermore, each domain need only be developed once and added to the repertoire of the system. Both implemented domains are features marked by the `DiagramFeaturesTool` and had no special layout requirements, meaning that only configuration of the existing aesthetic measures was required.

The use of Daidalos to create tool libraries involves drawing the diagram elements on the canvas, selecting the elements and defining the connectivity properties (if required). The data model of the elements is specified, including whether the

elements are editable and their Names. The new, composite, element is then placed into the tool library to be used repeatedly, at will.

The use of Ariadne to author exercises in those domains which have already been developed involves several operations. Parameters for Theseus must be specified to configure the menus, toolbar and other options which are available to student in the course of developing the exercise. Ariadne can be used to develop the marking scheme, including specifying the features test cases. In order to accomplish this, Ariadne invokes its own text editor. The exercise specification can also be input in this way, along with the editing of the properties files which are required for all exercises within CourseMarker.

Deployment and testing of the exercise through CourseMarker is then undertaken. Theseus is invoked as the student diagram editor from within CourseMarker by clicking the 'Develop' button after exercise set up. The exercise model solutions can be pasted into Theseus from Ariadne in turn, and used for the purposes of testing the marking and feedback results of the exercise and tweaking any problems. Specifically, a recommended way to determine the leniency value for the aesthetic and structural measures is to submit the model solutions, with no leniency applied, and examine the raw scores awarded to get an idea of what it is reasonable for the student to achieve within the constraints of the exercise. To access all the marking data a temporary concrete truncation strategy can be used which performs no truncation and changes the feedback from the measures to a string containing the raw score. Care should be taken to remove this strategy and replace it prior to making the exercise available to students.

Evaluation of the exercises was accomplished through two means. Firstly, the results of the exercises were stored by CourseMarker and made available for analysis. Secondly, questionnaires were distributed to students containing two types of questions. The majority of the questions asked the student to agree with a series of statements which were then scored on a five point Likert scale [Lr32], from 1-disagree to 5-agree. Finally, the questionnaire contained some open ended questions where student could make further, free-form comments. The questionnaires will be examined in further detail, along with the results obtained from students, in section 8.2.3.

8.2.2 Exercise domains and methodology

Prototype exercises were developed in two domains: UML Class Diagrams and UML Use Case Diagrams. UML Class Diagrams are used in the design process of object-oriented systems to describe the classes within the system and their relationships to each other. UML Use Case Diagrams are used to describe sets of scenarios which describe interactions between external actors and the system.

Section 8.2.2.1 provides a brief outline of the UML Use Case Diagram exercises, whilst section 8.2.2.2 provides an outline of the UML Class Diagram exercises. Section 8.2.2.3 outlines the methodology.

8.2.2.1 UML Use Case Diagram exercises

UML Use Case Diagrams are used to describe interactions between users and the system. They are conceptually easy for many students to understand and so were a suitable choice for the first domain to be implemented.

The authoring of prototypical exercises in the UML Use Case Diagram domain involves first specifying the nodes and connections to be used by students to construct their solutions. Two types of domain nodes are available to the student, namely actors and use cases, while one type of domain connection, interaction, is available. The final tool which is made available is the Border Tool object, which is used by the students to describe the physical borders of their diagrams.

Figure 8.1 demonstrates the tool library which is developed for UML Use Case Diagram exercises, while figure 8.2 shows a simple diagram constructed using the tool library.

The task of using Daidalos to create the tool library did not require much effort. The tool bar components are composed of groups of standard shapes, graphical primitives and text elements. Each tool has a data model which defines the name of the tool (for example, Actor), while each of the sub-components has also been named for reference purposes. The naming of these tools facilitates the basic mechanism for features marking of the resulting diagrams in the same way as in the earlier entity-relationship diagram coursework.

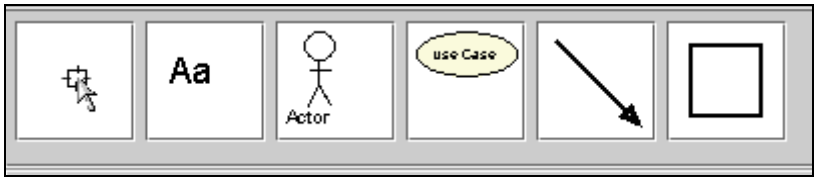

Figure 8.1: The tool library for UML use case diagrams

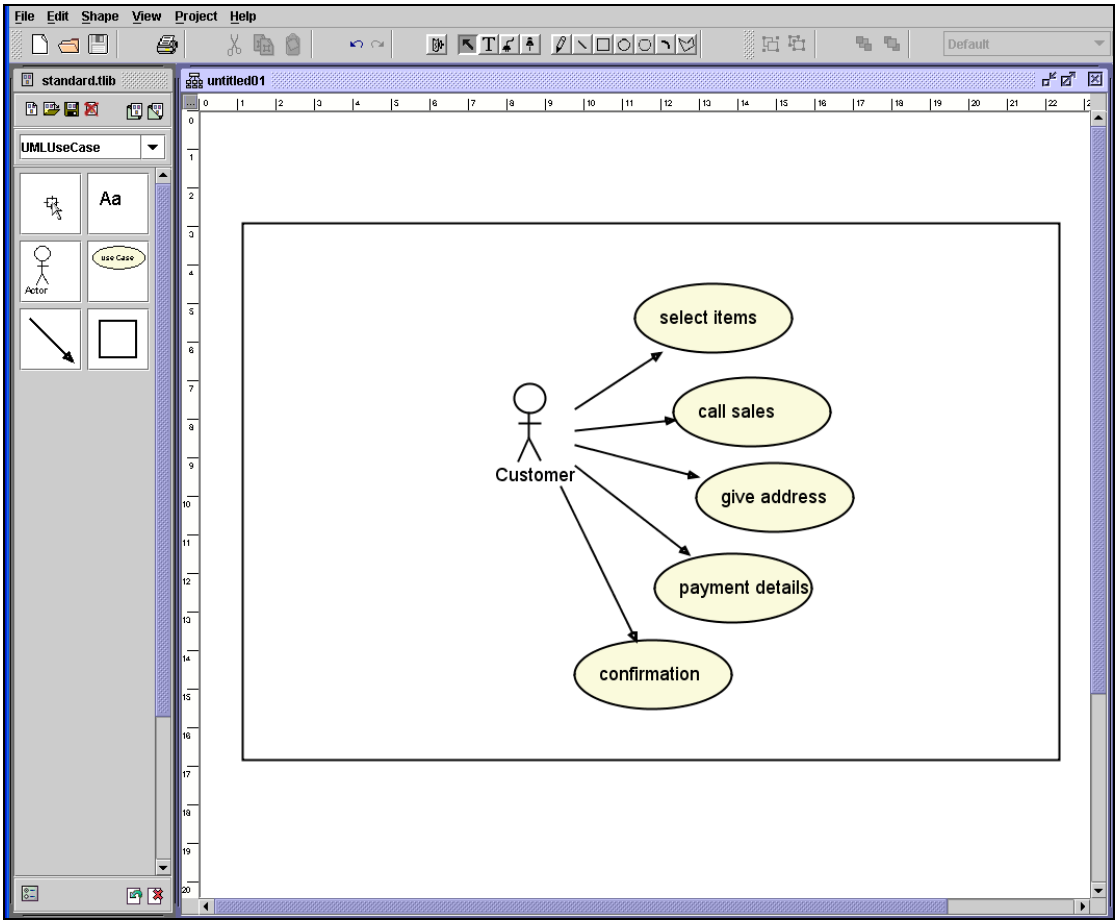

Figure 8.2: A simple use case diagram using the tool library

With the tool library complete, Ariadne is used to configure the features available to the student with the Theseus student diagram editor. Ariadne is then used to develop the marking scheme, configure the marking tools and the CBA exercise.

The `DiagramFeaturesTool` was used for features testing of the UML use case diagram solutions. The *exist*, *exact*, *connection* and *exactConnection* operators can be applied by stating the Name and, if required, the Text Content of the nodes (*Actor* and *UseCase*) and the Name, Start Node and End Node of the connection (*Interaction*) in the same way as for entity-relationship diagrams.

The process of developing the marking tools, exercises and feedback has been described in section 8.2.1. The application of this process to the UML Use Case Diagram Exercises is evaluated in section 8.2.3.

8.2.2.2 UML Class Diagram exercises

UML Class Diagrams are used in the design process of object-oriented systems to describe the classes within the system and their relationships to each other. The authoring of prototypical exercises in the UML Class diagram domain involves, again, first specifying the nodes and connections to be used by students to construct their solutions.

Two types of domain nodes are available to the student, both of which represent classes. The first simply allows the class name to be defined, while the second has editable text components for the class name, attributes and operations. Four types of domain connections are available: one-way associations, two-way associations, generalisation and implementation. Furthermore, generalisation must be configured as an "elbow-type" connection line. The final tool which is made available is the Border Tool object, which is used by the students, again, to define the physical borders of their diagrams.

Figure 8.3 demonstrates the tool library which is developed for UML Use Case Diagram exercises, while figure 8.4 shows a simple diagram constructed using the tool library.

Once again, the task of using Daidalos to create the tool library is straightforward. The tool bar components are composed of groups of standard shapes, graphical primitives and text elements. Each tool's data model is defined by naming the object after inserting it into the tool library, while each of the sub-components is named for reference purposes. Ariadne is again used to configure the features available to the student with the Theseus student diagram editor. Ariadne is then used to develop the marking scheme, configure the marking tools and the CBA exercise.

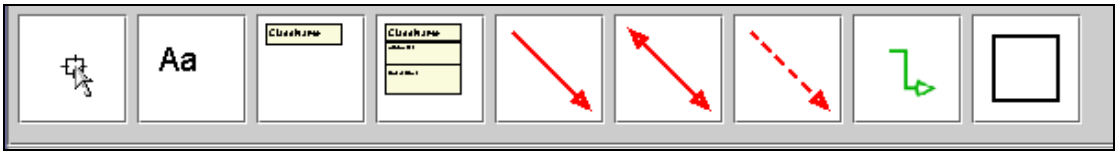

Figure 8.3: The tool library for UML class diagrams

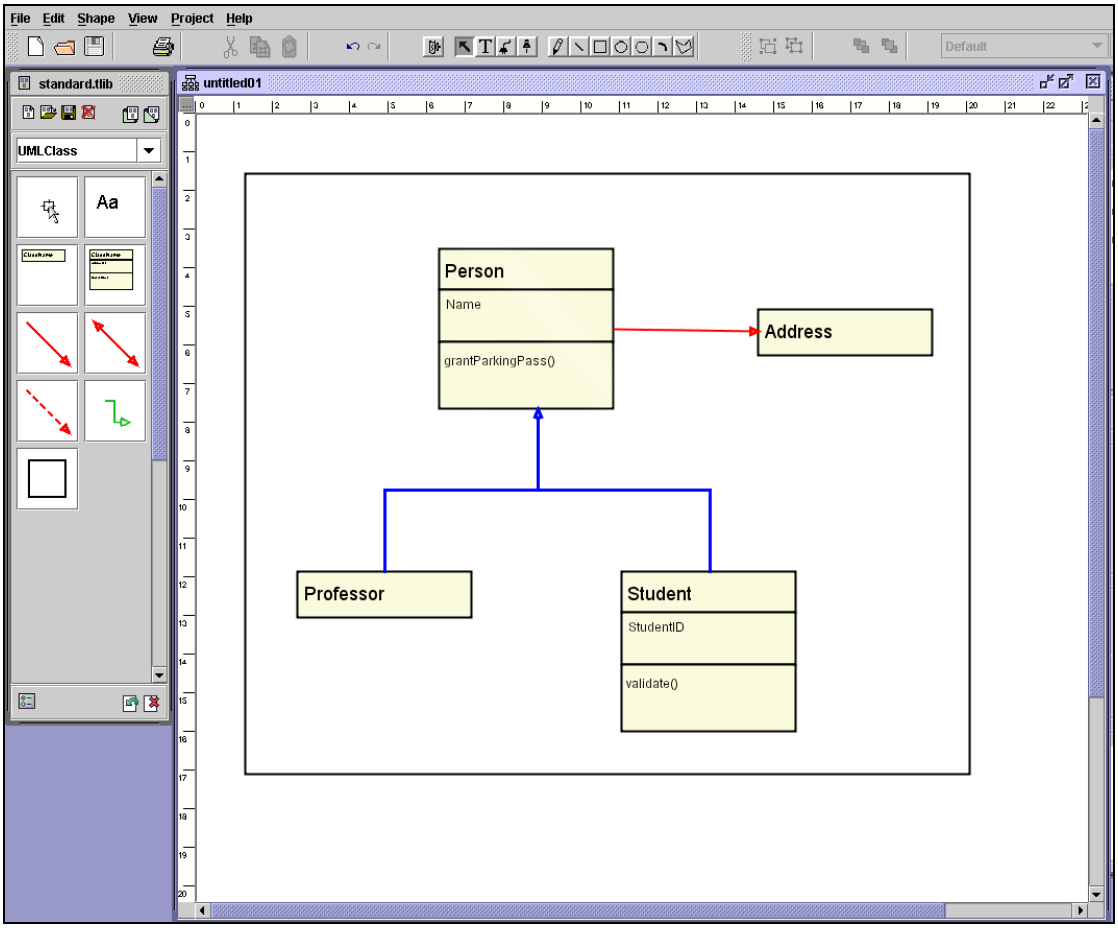

Figure 8.4: A simple class diagram using the tool library

The `DiagramFeaturesTool` was used for features testing of the UML Class Diagram solutions. The operators can be applied by stating the Name and, if required, the Text Content of the nodes and the Name, Start Node and End Node of the connection in the same way as for the use case diagrams.

The general process of developing the marking tools, exercises and feedback has been outlined in section 8.2.1. The application of this process to the UML Class Diagram Exercises is evaluated in section 8.2.3.

8.2.2.3 Methodology

30 undergraduate Computer Science students in their second year undertook the Use Case Diagram exercises, whilst 28 students (a subset of the 30) undertook the Class Diagram exercises. The students were volunteers; unlike the previous experiment described in chapter 4, there was no attached summative element to the exercises. Three exercises were set within each diagram domain, of gradually increasing complexity.

Quantitative data was collected, as before, by using CourseMarker's Archiving Server and by using Likert scale questions in student surveys. The student solution at each submission was captured using CourseMarker's Archiving Server, together with the hidden marks and the feedback.

In the previous experiment the questionnaires were poorly responded because their dissemination had been inadequately planned. For the Use Case and Class Diagram exercises the students were provided with the questionnaire soon after they were registered to take the exercises and were asked to complete and return the questionnaire once they had finished the course. The response rate was improved in comparison with the previous experiment.

In the questionnaires, the students were asked to indicate their agreement with a series of statements by choosing their level of agreement with the statement on a 5-point Likert scale. The statements were designed to assess whether the students had found the exercises easy to comprehend, whether the feedback provided by the system was considered useful, whether the students thought the system had assisted their learning process and whether the exercises were motivational, provoked the students to conduct further research to improve their answers and were considered a good use of time. The questionnaires were kept brief (11 statements) to try to minimise the extent to which the students found them tiresome. The 11 statements are presented in table 8.2.

Again, qualitative data was collected through the use of open-ended questions at the end of the student surveys. For these exercises there were no formal laboratory sessions and no paid lab tutors. Instead, students engaged with the course at their own pace and at their own time and asked for assistance by contacting a course email

address. Further qualitative data was obtained by keeping records of the emails sent by the students as a result of the course. This substituted for the tutor interviews.

8.2.3 Use and evaluation of the prototypical exercises

8.2.3.1 Constructing and running the exercises

Three use case diagram exercises and three class diagram exercises were set as coursework at the University of Nottingham. For logistical reasons, the students were essentially 'volunteers' with no system of compulsion possible to induce the students to register for the formative exercises. Students were sent information about the formative exercises and asked to register by email to receive access to the courses. Once a student had been added to the course list then viewing the course material involved loading the CourseMarker client and entering the standard username and password. No problems were encountered with this initial stage of the process for two reasons:

- Access to the CourseMarker client is available in all terminal rooms within the Computer Science building;
- The students had already used CourseMarker for previous exercises, especially the Java programming exercises which are compulsory for all Computer Science first year undergraduates, and were therefore familiar with the principles of logging on to the system, choosing the course to view and setting up their exercises.

The use case diagram exercises are attempted by students first, since they are the easiest to understand conceptually and, moreover, since the exercise model solutions constitute simpler diagrams than those for the class diagram exercises. The unit specification gives reference to a small example exercise specification and model solution. This is for the purposes of demonstrating good practice to the students. Furthermore, the first exercise varies only slightly from the example and has only one, simple model solution. The idea is to allow the student to concentrate on becoming comfortable with the Theseus student diagram editor and to provide an initial "confidence boost" before the second and third exercises present the student with more substantial domain problems.

Since the first exercise does not require mutually exclusive solution cases, then the `DiagramFeaturesTool` is invoked only once by the exercise marking scheme. For the subsequent exercises, mutually exclusive solution cases are identified and the marking tool is invoked repeatedly using the method demonstrated in figure 7.2. The identification of mutually exclusive solution cases is straightforward and the repeated invocation of the marking tool is rendered a trivial task. The creation of suitable feedback content is a non-trivial and very time-consuming process. Traditional CBA feedback comments such as "connection *x* absent" are scrupulously avoided but at extreme cost in exercise development time.

Model solutions are initially submitted to reveal the raw scores allocated by the aesthetic measures. The aesthetic measures' leniency is subsequently set to equal the initial raw scores so that unrealistic layout expectations were avoided. This process is straightforward, but the relative weighting of the aesthetic measures is set equal for each measure.

The prioritisation and truncation of student feedback is achieved through the use of four concrete strategies. The strategy for distinguishing between mutually exclusive alternate solution cases was as discussed in section 6.3.5. Sorting the feedback for all features tests was achieved as described in section 6.4.4. Since no structural measures were deployed, then comments were prioritised according to equation 6.6.

The truncation strategy utilised the concrete strategy outlined in section 6.4.4, parameterised such that the 2 highest priority features comments were retained along with the 1 highest priority aesthetic layout comment.

The UML class diagram exercises were attempted by students after the UML use case exercises had been completed. Again, the unit specification gives reference to a small example exercise specification and model solution for the purposes of clarifying good practice. The first exercise varies only slightly from the example and has only one, simple model solution, for motivational purposes.

The `DiagramFeaturesTool` is again used for features marking. Many of the features of developing UML Class Diagram exercises are similar to those for the development of the earlier, UML Use Case diagram exercises. The features tool is

repeatedly invoked in marking exercises 2 and 3, with the effort required to develop feedback being the most arduous stage of the exercise development.

For the UML Class Diagram exercises, the relative weighting of the non-interception aesthetic measure is reduced relative to the other aesthetic measures because the Generalisation connection routinely intercepts other generalisation connections in a manner which is not detrimental to the aesthetic layout of the diagram. Such an interception is present, for example, in figure 8.4. The other aesthetic measures are set to be equal for the purposes of the exercise, with leniency values which differ from those of the UML Use Case diagram exercise but which are initially defined in the same, standard manner.

Again, the prioritisation and truncation of student feedback is achieved through the use of four concrete strategies. The concrete strategies utilised are the same as for the Use Case Diagram exercises.

8.2.3.2 Evaluation of the exercises

A total of 36 students, all Computer Science undergraduates, volunteered and had their usernames added to the course list. Of these students, 6 subsequently failed to set up any exercises while 30 attempted the UML Use Case Diagram exercises. Of these 30 students, 28 continued on to attempt the UML Class Diagram exercises.

Student marks for all exercises were consistently high. In all exercises, nearly all students achieved "effective" full marks by the time of their final submission. The term "effective" full marks is used here to indicate that the student submissions were of a very high standard but did not achieve exactly 100% in features and aesthetics marking. This situation resulted because:

- Students were presented only with feedback, and not marks. Consequently, so long as the feedback was good then the student would not even realise that a mark of 100% had not been achieved and would move on to the next exercise.

- Aesthetics measures did not always return 100% due to the nature of the exercises.

The fact that students were presented with only feedback, with the underlying marks withheld, combined with the fact that the assessment was purely formative, seemed to reduce the temptation for gambling and perfectionism. The maximum number of submissions made for any exercise in either domain was 17. The average number of submissions for each of the exercises is shown in table 8.1.

Exercise	*UC1*	*UC2*	*UC3*	*Class1*	*Class2*	*Class3*
Average submissions	3	4	5	1	5	3

Table 8.1: Average submission numbers for the prototype exercises

Students were asked to complete a brief questionnaire, in order to summarise their experience of using the exercises and learning from the feedback. The majority of the questions asked the student to agree with a series of statements which were then scored on a five point Likert scale [Lr32], from 1-disagree to 5-agree. Table 8.2 shows the statements and the mean score. 22 completed questionnaires were returned.

Students found the courseware easy to use. The questions were regarded as easy to comprehend in both domains. Generally, students felt that the feedback was useful in improving their diagram and re-submitting a better version and many students were motivated to further research between submissions to find information which elaborated on the feedback comments. This was helped, no doubt, by the references included in the feedback comments themselves. Also, in general, students thought the exercises helped their learning process and were a good use of their time.

A notable trend in the questionnaire results, however, is that the UML Use Case Diagram exercises were more popular with students than the UML Class Diagram exercises. Several students noted, in response to the request for free-form comments at the end of the questionnaire, that the Class elements were difficult to edit so that the result was aesthetically pleasing due to the poor flexibility of the text elements

holding the attributes and operations. It is clear that the authoring of a new domain notation for class diagrams might improve student response in future.

Statement	*Mean score (N = 22)*
The system was easy to use	4.2
The Use Case Diagram questions were easy to comprehend	4.4
The feedback provided to my submitted Use Case Diagram coursework helped me to improve my diagram	4.0
The Use Case Diagram questions helped my learning process	4.0
The Class Diagram questions were easy to comprehend	4.1
The feedback given when I submitted the Class Diagram coursework helped me to improve my diagram	3.5
The Class Diagram questions helped my learning process	3.5
The feedback comments on appearance helped me to lay out my diagrams more clearly	3.8
The feedback I received for my submissions motivated me to research further	3.5
I made improvements to my solution as a result of the feedback I received and re-submitted the improved version	4.0
The exercises were a good use of my time	3.8

Table 8.2: Results of the student questionnaire

A notable trend across questionnaires was that those who responded more favourably to the statement questions tended to leave no further comments, while those who had responded less favourably were more likely to leave (critical) comments. The most common critical comment was that there were *not enough* exercises. It seemed that many students had hoped for a comprehensive series of courses to assist them up to modular examination level. Unfortunately, the development of such courses was not feasible due to time constraints and the difficulty encountered in constructing good formative feedback comments for the features tests. Howsoever critical, these comments do imply that the students wanted

more formative assessment using these methods. This is a desire which only the continued development of the exercises would help to fulfil.

This section has described the process of constructing and running the prototypical exercises. Subsequent sections will apply more rigorous examination of the performance of each of the extensions in turn, according to the criteria laid down by the three research areas of CBA, formative assessment and educational diagramming.

8.3 Assessing the aesthetic layout of student diagrams: evaluating performance

To assess the performance of the extension to allow the assessment of the aesthetic layout of student diagrams to occur, it is necessary to link the experience of designing exercise domains, authoring exercises, running the exercises and generating feedback to students to the requirements from each of the disciplines of CBA, formative assessment and educational diagramming identified in section 5.2.1. Section 6.2.1 linked the design of the extension to its requirements. Here, we relate the experience in use to those requirements.

8.3.1 Evaluating the extension as CBA

The prototypical exercises constitute good examples of formative assessment. The domain notations are designed online, the exercises are developed online and the management of the full lifecycle of a CBA exercise is achieved through an integrated, online system.

The three main requirements of the extension in relation to CBA are met. The aesthetic measures successfully provide a basis for the assessment of diagram aesthetics in both assessed domains. The system of aesthetic measures is comprehensive, such that the implementation of structural measures has not been required. However, given that structural measures operate in precisely the same way as aesthetic measures, there is no reason to believe that their use would have been any less successful than for aesthetic measures should additional layout criteria have proved necessary for the domains.

It proves possible to take into account educator preferences and differences between domains when assessing the aesthetic layout of different domains. The system of weighting allows the non-interception aesthetic measure to be "downgraded" in relative importance for the UML Class Diagrams with a trivial amount of effort.

It has become obvious, however, that the system of weighting would be more effective if the relative weights were defined based upon research outcomes rather than what amounts to carefully considered guesswork. It must be noted, however, that increased usage of the system would be a good platform for the research itself to be carried out, with weights improved in accuracy, year upon year.

The other requirements are met in entirety. The extension is successfully integrated into the marking and feedback systems and is transparent to students, who showed no awareness that any of the "behind the scenes" processes had changed. Indeed, the students seemed to regard the prototype exercises in much the same way as they had the compulsory Java programming exercises they had completed previously, since the exercise specifications were delivered and feedback was returned using a consistent format for both.

8.3.2 Evaluating the extension as formative assessment

Most students agreed that the feedback comments relating to the aesthetic layout of their solutions had helped to improve the appearance of their diagrams. The feedback was motivational since many students were inspired to improve their diagrams and re-submit with better versions because of it. Section 8.3.1 has discussed the integration of the extensions into the feedback system, and the evaluation of the extension for prioritising and truncating feedback will consider this issue further.

8.3.3 Evaluating the extension as educational diagramming

Within the context of educational diagramming, the key requirements have been met. The system provides a basis for assessing diagrams generically through the application of aesthetic criteria, which have successfully assessed diagrams in two domains based upon different weighting and leniency configurations. A platform for extension to accommodate new domains has been provided by allowing future developers to develop further layout criteria, on a domain specific basis, which operate according to the same design principles. Currently, the aesthetic measures

operate on a domain-independent basis apart from exercise-specific configuration, and are based upon established aesthetic principles from research fields. The relative importance of criteria can be taken into account through the system of weighting, although section 8.3.1 has noted already that further research into precise weighting values on a per-domain basis would be useful. The system of leniency values, on the other hand, benefits from being able to be configured using a defined mechanism. Fundamentally, the system has succeeded in improving the appearance of student diagrams and in assessing the clarity of the diagram as well as its features-based correctness. This, therefore, represents a positive achievement in educational diagramming.

8.4 Assessing solutions with mutually exclusive solution cases: evaluating performance

This section considers how the extension to allow the assessment of solutions with more than one acceptable model solution through defining mutually exclusive alternate solution cases has performed, relative to the requirements in CBA, formative assessment and educational diagramming which were established in section 5.2.2. Section 6.2.2 previously linked the design of the extension to its requirements. Here, we consider the experience in running prototypical exercises.

8.4.1 Evaluating the extension as CBA

The requirements for the extension from a CBA perspective have been met. Defining mutually exclusive solution cases is a straightforward, repetitive process once the multiple model solutions have been developed. It is necessary to consider all possible model solutions — barring the minor variations in labelling which can be taken into account by defining flexible regular expressions as Oracles — which raises the possibility that a student with a particularly novel solution might not receive a suitable response. However, given that exercises constructed using the extension are intended for formative assessment purposes, there is no possibility of a student losing credit through originality, and, furthermore, this issue did not arise during the running of the prototypical coursework.

The exercise developer is able to specify the common case and each of the mutually exclusive solution cases using a consistent notation, with little more effort than for

conventional CBA exercises using CourseMarker. Furthermore, the extension is integrated into the marking and feedback system in such a way that complete transparency is achieved from the point-of-view of the student.

8.4.2 Evaluating the extension as formative assessment

From the perspective of formative assessment, the central requirement is feedback. The difficulty of developing good formative feedback comments for CBA exercises must not be underestimated when the process of exercise development begins. Developing good feedback is a time-consuming process requiring the construction of carefully phrased, motivational comments relating to the principles tested by the associated features expressions within the features test. Furthermore, it is necessary to link the comments to learning materials. This could be achieved, firstly, by locating good reference material for the various feedback comments — trying not to reference the same text repeatedly if student research is to be nurtured — or, secondly, by developing a wide selection of bespoke research material for integration into the CBA course. The first option is feasible, if sufficient priority is attached to the exercise feedback by the educator. The second option may only be feasible in extraordinary cases. A third solution which could be considered in the future is the integration of courseware into content management systems, with active linking between assessment feedback and teaching materials within the CMS.

The assessment process was able to determine which version of the model solution the student was able to attain and to tailor the feedback accordingly. This process occurred smoothly across both prototypical domains. Again, with reference to the creation of feedback by the educator, most identifiable distinction tests were based upon *imperfect harbingers*, since it proved difficult to identify precise pedagogical reasons which encapsulated the difference between different model solutions in many cases. However, with effort, it proved possible to construct useful, motivational feedback.

The success of the feedback when the formative assessment framework criteria, summarised in section 2.2.5, are applied is considered in section 8.5.2.

8.4.3 Evaluating the extension as educational diagramming

The requirements within the context of educational diagramming are similar to those considered in section 8.3.3. The system of aesthetic and structural measures, further parameterised by relative weighting and leniency values, allowed a basis for assessment in a wide variety of educational diagram domains to be provided.

The specification of common and mutually exclusive features is a standard process involving the determination of common and uncommon solution elements in the various model solutions. So long as the `DiagramFeaturesTool` is used to assess diagram features in a generic way then consistency across domains is also achieved, even down to the level of the features expressions which are evaluated by the marking tools. Any operator available within the `DiagramFeaturesTool` may be applied freely to any aspect of the student diagram, and this, indeed, occurred successfully within the prototypical exercises. Further potential in allowing educators to develop their own criteria is presented by the decoupling of the extension from the features tool used. Although the `DiagramFeaturesTool` was developed to provide generic functionality, new marking tools may be developed and substituted for the `DiagramFeaturesTool` by merely changing the invocation in the exercise marking scheme.

8.5 Prioritising and truncating the feedback: evaluating performance

The extension responsible for the prioritisation and truncation of feedback has requirements from the perspective of CBA and formative assessment. This section provides an overview of how the experience gained from the prototypical exercises demonstrates that the requirements from these areas, defined in section 5.2.3, have been fulfilled.

8.5.1 Evaluating the extension as CBA

The central requirements from the perspective of CBA were the integration of the mechanism for prioritising and truncating feedback into the architecture, which was achieved. The prioritisation and extension successfully occurs for each submission without human marker intervention. Comment priority can be specified by the exercise developer in terms of the system of weights; although some care must be

taken when defining the weights, the system was shown to operate successfully for the prototypical exercises and the students were motivated by the resulting feedback.

8.5.2 Evaluating the extension as formative assessment

From a formative assessment perspective, the central requirement was to provide only the most relevant feedback comments in order to allow the student to concentrate on key improvements to their solutions which are required, without being "overloaded" by irrelevant comments. Flexibility for the exercise developer has been achieved through the implementation of the system of *Strategies*, in which each *Strategy* represents a sub-problem within the prioritisation and truncation of feedback.

The purpose of the prioritisation and truncation strategy was to adapt the feedback provided by automated assessment into a form which would, if combined with a flexible marking system, motivational feedback and a CBA platform, provide a framework for effective feedback for formative assessment to be delivered.

Section 2.2.5 outlined the properties to which a framework for effective formative feedback should conform. Firstly, formative assessment should facilitate the development of self assessment, or *reflection*, in learning. The assessment programme was not educator-led in any sense. The students were free to work through the exercises at their own pace and relied for feedback upon the courseware assessment. Feedback comments had been engineered to be motivational, to encourage further research and to provide good references as a starting point for that research. The students were therefore compelled to reflect upon how the information they had been directed to would help to improve their coursework solution, a task involving both critical self-assessment and the gradual improvement of a student's internal perception of what is required from the coursework.

Secondly, formative assessment should encourage teacher and peer dialogue around learning. Students were aware that they could confer and collaborate on exercises to the extent that they wished. Indeed, it would have been impossible to prevent students from acting in this way due to the lack of restrictions on the availability of the exercises through CourseMarker. Students could collaborate and assists each

other during exercises. Furthermore, several students sent emails containing queries about coursework issues.

Thirdly, formative assessment should clarify what constitutes good performance. The unit specification within each exercise domain was authored to include a simple specification of an example exercise, together with links to diagram solutions which would satisfy the requirements of the specification. Including the example in the unit specification allowed students to grasp the goals of the unit at an early stage, so that the correlation between the internal perception of the student and the actual goals of the educator was maximised. Questionnaire results demonstrate both that students found the exercises easy to comprehend and that students felt their coursework solutions had been improved as a result of the feedback to earlier submissions.

Fourthly, formative assessment should provide opportunities to improve performance. Table 8.1 demonstrates that students made multiple submissions for exercises in the majority of cases. Students agreed that they utilised the feedback provided to early submissions in order to improve their performance and re-submit. The fact that the assessment process is fully automated allows students to submit solutions several times with no increased workload for the educator.

Fifthly, formative assessment should deliver information focused on student learning. The feedback was always delivered in good time due to the nature of the automated assessment process. In practical terms, feedback is delivered instantaneously in all cases. The prioritisation and truncation extension is a key component in the process of delivering feedback which is not overwhelming in quantity. A large number of features and aesthetic layout comments are sorted by priority, with truncation allowed according to the specification of the educator, who can control the number of criteria about which feedback is given. The student perceives the feedback as targeted and hence does not lose the view of the exercise as a holistic entity.

Sixthly, formative assessment should encourage positive motivational beliefs and self-esteem. Students were aware that the prototypical exercises carried no summative assessment weight or course credit. Therefore, students were able to relax and enjoy the process of learning rather than concentrating upon the achievement of good marks. Feedback concentrated on learning goals by specifying good practice

and referring to the educational literature. Questionnaire results demonstrate that, overall, students thought that the exercises were a good use of their time, even though no course credit was gained from them.

Seventhly, formative feedback should provide information to educators that can be used to help shape the teaching. CourseMarker archives all submissions across a course. Administrators can retrieve submissions, precise marks and the feedback provided for each submission. Even details such as the time of the submissions are stored. Substantial material for further research can be generated by the implementation of exercises within such a CBA context.

8.6 Conclusions

Chapter 9 will review the key points of the thesis to show how the evaluation of the system relates to the general objectives for research stated at the beginning. The general objectives of this work asked several specific questions, which can now be answered. The purpose of this section is to discuss each of these questions in turn and to argue that the formative CBA of diagram-based domains is both feasible and useful.

This work has demonstrated that the formative CBA of diagram-based domains is certainly possible. Feasibility is assessed by determining whether the level of difficulty encountered in developing and deploying the exercises would render the process too difficult to be developed by educators. Usefulness, on the other hand, is primarily assessed by determining whether the exercises were useful to students and enhanced their learning process.

The automated system for the marking of the aesthetic layout of student diagrams was thought by students to have provided useful feedback which improved the layout of their diagrams. Prototypical exercises have demonstrated that domain-specific layout rules are not required for each domain. The trade-offs required to ensure generality across domains while at the same time allowing specialisation involve the need to specify general functionality while also allowing that functionality to be extended by future developers in a carefully defined way. The system of aesthetic and structural measures accomplishes this through defining a basic range of domain-independent functionality which can be invoked in many

domains, while allowing similar marking tools to be developed around domain-specific criteria in the form of structural domains. Furthermore, the distinction between aesthetic and structural domains allows clarity when considering those measures necessary to assess each new domain. Aesthetic measures should be used by default and only sidelined with justification. Conversely, structural measures are intended to be domain-specific, and therefore do not need to be considered for inclusion when new domains are developed, without specific reason on the behalf of the educator.

The extent to which it is possible for the educator to provide formative feedback in many diagram-based domains by configuring the system and writing feedback comments is constrained by the similarity of the domains. In the worst case, the development of a marking tool, structural measures and new concrete prioritisation and truncation strategies would have to be carried out once for each domain. Configuring of exercises within the domain is possible through configuration and the writing of feedback content. However, such a worst case scenario is not inevitable or even common. The development of those components required for a new domain can often take advantage of the similarity between many educational diagram domains. Adaptation or even complete re-use of components designed for use with a previous domain is plausible in many cases and the probability of existing components being useful increases as new domains are developed for assessment and more components are created. Furthermore, the extensions were created to provide a generic basis for formative assessment, including a domain-independent tool for features marking, a general suite of aesthetic measures and example concrete strategies to solve the problem of prioritising and truncating student feedback.

The main area where standardisation of CBA processes has failed to occur is in the creation of the formative feedback. This chapter has shown, through the documentation of prototypical exercises, that CBA can be used to deliver good formative assessment. It must be emphasised, however, that the effort involved in creating the exercises was great due to the feedback requirements. With this in mind, it must be understood that formative assessment can be rendered less resource-intensive through the use of CBA technology, so long as the potential for long-term re-use of the exercises is considered. In fact, re-use of formative assessment diagrams does not pose problems of question security in the same way that re-using

summative questions does. Furthermore, courses can be incrementally improved, year-on-year, both by adding new exercises each year to increase the coverage of the domain, and by taking into account student comments to improve existing exercises.

A formative assessment process which is automated using CBA technology has been shown to enhance student learning. The exercises were popular with those students who enrolled and can be shown, as in section 8.5.2, to conform to the framework for good formative assessment practice.

Chapter 9 builds upon the experience documented within this chapter, and the evaluation which followed, to initiate a discussion surrounding the two fundamental questions which formed the basis of this work. Furthermore, the contributions of the work are examined and future work is proposed.

Chapter 9

Conclusions

Introduction

This chapter reviews the key points of the thesis to show how the evaluation of the system relates to the general objectives for research stated at the outset. The contributions of the research are discussed and areas for further research to be carried out in the future are considered.

Section 9.1 discusses the way in which the research has approached the problem of conducting formative, computer-based assessment in diagram-based domains. A summary is provided of how the work has met its general and specific objectives according to the requirements set out in chapter 5. Section 9.2 provides a summary of the contributions of this work while section 9.3 outlines areas for future research in the key topic areas of the thesis. Finally, section 9.4 concludes with an epilogue on CBA, formative assessment and educational diagramming.

9.1 Meeting the objectives

Chapter 5 demonstrated that, in order to prove that the automation of the formative assessment of diagram-based coursework using CBA courseware is both feasible and useful, the design, implementation and integration into the existing courseware of three key extensions was necessary. The three identified areas of extension are:

- Extending the marking system to assess the aesthetics of student diagrams;
- Extending the marking system to allow the assessment of mutually exclusive solution cases;
- Changing the system of feedback to provide only the highest priority comments to students.

This section revisits these objectives and demonstrates that the key objectives have been accomplished.

9.1.1 Assessing the aesthetic layout of student diagrams

Allowing the assessment of the aesthetic layout of student diagrams is necessary if the formative assessment is to assist student learning within a diagram domain. Educational diagrams convey domain-specific information through their convention

of meaning, but if the aesthetic layout of the diagram is poor then the meaning of the diagram may be poorly understood. Initial experimentation, described in chapter 4, showed that students often produced diagrams of poor aesthetic appearance which conveyed information in an unclear way. If the aesthetic appearance of student diagrams is not assessed, and no feedback to the student provided, then the student will not be given the incentive to improve their diagram-based solution in this key aspect.

The design and implementation of the extensible mechanism to allow the aesthetic layout of student diagrams to be assessed has been successful. The extension was realised and integrated successfully into the existing CourseMarker architecture. The potential for domain coverage is large. Representations for new domains can be authored easily using the Theseus diagram editor. The only constraint, if the aesthetic layout of the domain is to be assessed, is necessity of inclusion within the domain notation of a Border Tool which can be used by the student to indicate the boundaries of their diagram; this is a trivial task. The system of aesthetic and structural measures allows the aesthetic layout of any diagram domain to be assessed. Aesthetic measures are built around domain-independent, general purpose aesthetic layout criteria which can be used to assess the diagram appearance of a large number of educational diagram domains. Structural measures can be used to extend the layout marking to incorporate any domain-specific criterion necessary to assess aesthetics within a new domain that may arise.

The construction of new structural measures requires programming from the developer and, as such, is non-trivial. However, layout tools do not require extensive coding and, furthermore, are not required at all for many educational diagram domains which may be assessed successfully using the provided aesthetic measures alone. Once a structural measure has been created, it can be used for all exercises of the same type.

The extension to allow the assessment of the aesthetic layout of student diagrams was designed with the requirements to provide a basis for assessment of aesthetic layout, to allow extension to support future aesthetic layout requirements and to integrate into existing courseware architecture as prime concerns. The first two requirements were met by considering the commonality and variation across layout

criteria for diagram domains. Commonality was represented by aesthetic measures, whilst variation was represented through allowing extension as structural measures. The distinction between aesthetic and structural measures was essential in clarifying cross-domain requirements to the educator. A design in which layout measures were applied generally to each domain would result in greater effort on the part of educators to consider suitability on a domain-specific basis. Inevitably, this would result in the number of applied measures for each domain being reduced due to the amount of time required to determine suitability. With the design described by this work, the educator need only consider the *special* requirements of the domain in order to determine the necessary layout tools.

The extension has been successfully integrated into the CourseMarker marking system. It is implemented as a series of marking tools and interfaces and provides feedback in a form which can be utilised by the CourseMarker feedback system. This final requirement allows the extension to be deployed as part of CourseMarker's high-performing, platform neutral architecture. This allows the creation and delivery of aesthetic-based feedback comments to be delivered to the student.

9.1.2 Assessing solutions with mutually exclusive solution cases

The system of features marking within CourseMarker had been developed as part of the Generic Marking Mechanism as a general tool to allow marking to be accomplished across domains with widely varying notions of quality [Ta02]. In the process of summative assessment to which CourseMarker was applied, students were often to be tested using very specific problem specifications which designated one technique of solving a problem to be used and often forbade all others. This was especially true in programming exercises where students were expected to concentrate on a new programming construct, or a new set of programming constructs, in each week's assessment.

Within the context of the formative assessment of diagrams this method was demonstrated to be insufficiently flexible. Chapter 4 outlined the problems associated with the approach. Initial, simple exercises could be appropriately marked in many cases, but as the exercise specifications grew more complex the possibility of multiple model solutions being acceptable meant that features testing was markedly less comprehensive.

It became clear that, if mutually exclusive solution cases could not be assessed, then features marking would be reduced either to marking the common subset of features, or to restricting learning through precise problem specifications. Both of these possibilities would have a negative effect on student learning in diagram-based domains and, hence, on the formative assessment process.

The design and implementation of the mechanism to allow the assessment of solutions with mutually exclusive solution cases has been successful, with the extension integrated successfully into the existing CourseMarker architecture. The mechanism is generic, allowing any marking tool which has been defined to be used. A `DiagramFeaturesTool` is implemented to allow generic features testing in common educational diagram domains, but the functionality can be extended to cover domain-specific features tests by designing and implementing a new marking tool and invoking it within the marking scheme in place of the `DiagramFeaturesTool`. This approach is consistent with the notion, applied throughout this work, that a basis of existing functionality should be supplemented by the possibility of expansion by developers to accommodate new domains, which may raise previous unforeseen requirements, in the future.

Defining the feedback comments which will be delivered to students according to the evaluation of each individual features expression is a non-trivial task which takes much time. Despite this, the fact that the nature of formative assessment allows great potential for exercise re-use means that resource-savings can be made over the medium and long-term, resulting in a set of consistently marked exercises which can be incrementally improved or added to over time.

The extension has been successfully integrated into the CourseMarker marking system. The `DiagramFeaturesTool` is implemented as a CourseMarker marking tool with generic functionality while the design takes into account the flexibility of the exercise marking schemes to repeatedly invoke the marking tool to assess each mutually exclusive solution case in turn. Feedback is returned in the `MarkingLeafResult` and `MarkingCompositeResult` structure which can be utilised by the CourseMarker feedback system, thus allowing the extension to be deployed as an integrated part of CourseMarker's high-performing, platform neutral infrastructure.

9.1.3 Prioritising and truncating student feedback

CourseMarker's feedback system was developed for the purposes of providing a concise, expandable representation of the feedback generated for each stage in the marking process. For summative assessment purposes, this representation provided a useful breakdown of the grades awarded for the exercise, maximised for ease of referral. For the purposes of formative assessment, however, the feedback was unwieldy, unfocused, un-motivational and contained many feedback comments which were unhelpful and irrelevant. The extension allows the feedback to be prioritised according to defined criteria and strategies before being truncated according to educator preferences.

The design of the mechanism for prioritisation and truncation of feedback divides the task into four sub-problems: distinguishing between mutually exclusive solution cases, prioritising features feedback, prioritising aesthetic layout feedback and truncating the feedback tree. The implementation and integration into the existing CourseMarker infrastructure have been successful. The potential for domain coverage is large: any feedback can be prioritised and truncated so long as the raw feedback tree can be generated using marking tools and appropriate concrete strategies to solve each of the four sub-problems are defined. A basis for prioritisation and truncation has been provided through the implementation of example concrete strategies, while the interfaces and abstract classes defined offer a precise extension point for developers in future.

Implementing and integrating new truncation strategies in the future, based upon educator preferences, is a straightforward task. So long as the new concrete strategy extends the correct abstract class according to the sub-problem it is developed to solve, only the construction of a simple algorithm to encapsulate the strategy is left to the developer.

Invocation in the exercise marking scheme represents the point of direct integration into CourseMarker. The process of invoking the `PrioritiseTruncateTool` and parameterising it using the correct concrete strategy objects is logical and straightforward. The `PrioritiseTruncateTool` returns the truncated feedback as a new composite marking tree, which is returned transparently to the student.

9.2 Contributions

The main contributions of this work are in the area of CBA, but advances can also be demonstrated in the fields of formative assessment and educational diagramming. Novel experience in these fields has been gained through the design, implementation, integration into courseware and evaluation of the extensions to allow aesthetic layout marking, assessment of multiple model solutions through mutually exclusive solution cases and the prioritisation and truncation of student feedback. The following sections summarise the key contributions in each of the three areas of research.

9.2.1 CBA

The most obvious contribution to CBA is in the development and deployment of a new type of CBA. Formative assessment of a free-response domain such as educational diagrams has not been attempted prior to this work in such a manner as to take into account such factors as mutually exclusive features correctness, aesthetic layout and configuration of feedback. Free-response CBA still constitutes a minority of systems in the field because of its perceived "difficulty"; this work demonstrates that useful, formative assessment can be deployed within a free-response domain such as educational diagrams and provides a platform for deployment across many domains and an example of the incorporation of such features into an existing CBA architecture. Another contribution to CBA lies in advancing understanding within the CBA community as to what constitutes good formative assessment; previous CBA work, including that of Tsintsifas [Ta02] has argued that formative assessment is merely summative assessment with the marks discarded. This work has developed the understanding, within a CBA context, that this is not the case.

A deeper contribution to CBA is that the software deliverable can be used as the basis for further research. Although prototypical exercises have been deployed for the purposes of assessing the feasibility and usefulness of the concepts, it is clear that each of the extensions provides clear potential for further work by CBA developers and researchers into areas such as the automated marking of diagram layout, the automated marking of coursework with multiple valid model solutions and the truncation and prioritisation of student feedback. Section 9.3 considers the potential for future research based upon the research described here.

9.2.2 Formative assessment

The contribution to formative assessment is in the application of CBA to the problem of formative assessment decline. Chapter 2 highlighted the solutions suggested in the literature to adapt formative assessment to a changing educational climate characterised by less favourable staff to student ratios. The literature highlights that "mechanisation" may be an option but the scope of ambition is fairly modest. CBA courseware offers potential for considerable resource-savings through a total automation of the formative processes for assessment and the return of feedback, provided that the infrastructure is sufficiently flexible and has been adapted to formative assessment requirements. This work has demonstrated that such an approach is feasible and useful, provided advice for educators related to the issues involved and provided useful experience for developers.

Moreover, the integration of the extensions into CourseMarker provides a basis for impact upon formative assessment automation within higher education institutions. CourseMarker has been successfully deployed at at least 15 other higher academic institutions and is prominently cited within the literature. The contributions made by this work therefore have the potential to impact upon future research in formative assessment with a view to changing the common perception that the assessment form is necessarily resource-intensive.

9.2.3 Educational diagramming

The contribution to the field of educational diagramming lies in the creation of an extensible theoretical framework for the assessment of educational diagram aesthetics, the implementation of the framework and its deployment as part of a CBA courseware system. Research into educational diagrams has previously taken into account automated layout algorithms, whereby an algorithm is used to place nodes and edges in such a way that the result is pleasing to a human eye. This work provides a basis for the inverse process — that of assessing the aesthetics of a diagram which has been drawn by a student. Again, the contribution further lies in the use of the deliverables for the purposes of further research. Research into the relative importance of aesthetic layout criteria can be facilitated through the system. Section 9.3 considers the potential for future research based upon the research described here.

9.3 Future Work

A central contribution of this work is the development of a solid basis for future extension by researchers. The work described here can be used to facilitate future research in numerous ways which span various research areas. This section considers the potential for future research in the areas of CBA, formative assessment and educational diagramming.

9.3.1 CBA

As well as being deployed within several educational institutions as a live CBA system, CourseMarker has been, and continues to be, an active research platform for academic researchers and students. Interesting student projects would include the development of courses, including exercises and concrete designs for further structural measures, diagram marking tools to be incorporated into the mutually exclusive features case extension, and concrete strategies.

An interesting piece of future research would be to investigate the extent to which the extension described here for mutually exclusive solution cases could be usefully deployed in other domains, such as programming exercises. It has been noted that, currently, programming exercise specifications are used to "shepherd" students into making use of an exact programming construct, which is subsequently the subject of features testing. Research could be conducted to determine the extent to which mutually exclusive solution cases could be used to "relax" specifications in existing programming courses with a view to providing increased flexibility to students in problem-solving and a less rigid problem specification.

Further research could be conducted which uses the aesthetic layout marking tools as a starting point. To what extent could the layout marking mechanism be adapted to a summative assessment scenario in which the *raison d'être* is the exact summative mark returned by the system, rather than the feedback. This requirement would place greater emphasis on the need for accurate grading, both between the various aesthetic and structural measures and between the layout measures and other grade-contributing factors from other marking tools. A special emphasis would need to be placed upon finding a mechanism to adequately aggregate marks to produce a fair overall summative result.

Further research could also aim to further augment the usefulness of the feedback. Currently, the feedback is modified based upon only the current submission. Further research could aim to take previous submissions' feedback into account, developing for the student a "satisfying" sense that their individual progress is being aided. The idea of using AI agents to monitor student progress and tailor feedback through gathering data from both submissions and administrative data has been proposed previously [Ta02], but has yet to be realised.

Furthermore, there is a need to address the interoperability of CBA exercises. At the time of writing, research is already underway within the LTR research group into defining free-response CBA questions in an interoperable way. It would be useful if formative assessment exercises within diagram-based domains were to be approached using such a methodology so that, when further courseware incorporates functionality similar to that described by this work, exercises could be created and used on a cross-platform basis.

9.3.2 Formative assessment

Several future research projects within the area of formative assessment could use this work as a departure point. This work examines the formative assessment using CBA courseware of diagram-based domains. Diagram-based domains were chosen because of their free-response nature, which allows higher-order cognitive levels to be more easily assessed, and their wide potential for cross-disciplinary application. This research, however, raises questions as to whether CBA automation could be applied to formative assessment in other domains such as programming using the approach described by this work.

Further research could seek to investigate whether different methodologies in feedback construction for the formative exercises resulted in better assistance for the learning process of the student. This research provided a set of brief guidance to educators which touched upon the subject of feedback construction. However, this area could be the subject of considerable further research. As well as the direct methods of constructing and phrasing feedback comments, projects could also investigate strategies for providing extensive teaching material content through the courseware and providing direct links to this content from the feedback area.

9.3.3 Educational diagrams

The extensions described by this work, and their integration into courseware, facilitates research into several areas of educational diagramming.

This research provides a platform to determine whether theoretical criteria for the assessment of educational diagram aesthetics will accurately provide an indicator of the appearance of a student diagram. Research can now be conducted to determine the perception of diagram layout among domain novices rather than experts, as is usually the case. Methods could also be investigated to optimise the priority accorded to the various layout measures through interactive modification of the weight values assigned to the measures in the marking scheme.

Furthermore, in the general area of diagramming, this research offers the potential to carry out research which reverses the traditional methodologies of those such as Purchase et al [PAC02]. Perception of aesthetic layout criteria could be tested by asking volunteers to interactively construct diagrams with priority placed upon a defined aesthetic layout criterion. The drawing could then be assessed objectively by the marking system to see the actual level, as opposed to the level perceived by the volunteer, that the drawing had achieved according to the criteria. Similarly, interactions between criteria when combined could be investigated. This reverses the traditional methodology in which diagrams are prepared as part of the research materials, with volunteers assessing the layout of the pre-drawn diagrams.

9.4 Epilogue

This research investigated the feasibility and usefulness of the idea of automating formative assessment coursework using CBA courseware, in free-response diagram-based domains. Work on free-response, diagram-based CBA is sparse. It is for this reason that formative assessment using CBA in such domains has not been reported prior to this work.

An initial research phase involved the construction of CBA exercises using existing courseware with only minor, obvious, modification. The initial phase of the research pointed to the fact that CBA courseware certainly had the potential to deliver formative assessment courses in free-response domains, but that the current techniques and methodology associated with the assessment required augmentation.

Nevertheless, the initial research did indicate that several key concepts, such as the use of two-part assessment as a motivator, held true in real world courses.

The design of extensions to available CBA courseware capability and the integration of the extensions into an existing CBA courseware architecture subsequently formed the core of this work. The deliverables of the work have been used to demonstrate that the assessment could be achieved feasibly and could be useful for students, in a formative assessment context, by aiding their learning process.

The first extension, to allow the aesthetic layout of student diagram solutions to be assessed, resulted from the principle that formative assessment, in being responsible for the learning process, must take great care to teach good practice. The second extension, to allow assessment to occur in situations where multiple model solutions are plausible, took into account the idea that formative assessment must attempt to allow scope for learning and student variation. The third extension, to prioritise and truncate student feedback, resulted from the notion that large lists of marks and feedback, incorporating many irrelevant feedback items, may result in student 'overload' and a subsequent failure to apply the feedback to further learning. In all cases, the extensions were designed to be extensible to adapt to subsequent developments, and were successfully integrated into the CourseMarker courseware architecture.

Two central questions formed the inspiration for this work. Firstly, to what extent can CBA techniques be used to reduce the resource required in setting a formatively assessed coursework in a diagram-based domain, marking student submissions and returning feedback, while still adhering to good formative assessment principles? Secondly, to what extent would current, successful CBA practices need to be changed to conform to formal formative assessment guidelines?

This research has provided answers to both of these questions. CBA techniques can be used extensively to reduce the resource-intensiveness of formative assessment while conforming to good formative assessment principles, but the processes of determining requirements for new domains and authoring exercises can be lengthy and involved. The resource saving of CBA exercises is derived from the fact that, once designed and deployed, they can be used repeatedly over many academic years. Indeed, assessment materials can be gathered over the course of several

academic years, with the amount of formative assessment conducted through traditional methods gradually reduced in parallel as both educator and student confidence with the new technology increases. Furthermore, the marking of submissions and returning of feedback opens up new avenues of formative assessment, such as the potential for repeated re-submission, which would *never* have been plausible using traditional marking methods.

Conversely, CBA techniques can benefit greatly through being subject to the scrutiny of assessment guidelines like those available for formative assessment. Technology-led solutions such as CBA courseware are often motivated by the desire to automate those types of assessment, and feedback, which are the easiest to achieve in practice. Studying the CourseMarker CBA system through "the lens" of formative assessment criteria has opened up new lines of research enquiry and resulted in considerable functionality being added to the system of assessment and feedback. It is only when rigorous scrutiny is applied across the board that CBA techniques will attain a true level of acceptance within higher education and finally convince their many detractors of their merit.

Bibliography

All URLs were last accessed on 31 March 2007.

AK01 Anderson L.W. and Krathwohl D.R. (eds.), A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, Longman, 2001, ISBN 0321084055.

AMW06 Axelsson K., Melin U. and Wedland T., Student Activity in Seminars – Designing Multi-functional Assessment Events, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 06), Bologna, Italy, p93-97, June 26-28, 2006.

ANSI05 American National Standards Institute, Overview of the U.S. Standardization System: Understanding the U.S. Voluntary Consensus Standardization and Conformity Assessment Infrastructure, ANSI, July, 2005. Available from: http://www.ansi.org/

App89 Apple Computer Inc., MacAppII Programmer's Guide, Apple Computer Inc., 1989.

APR06 Amelung M., Piotrowski M. and Rösner D., EduComponents: Experiences in E-Assessment in Computer Science Education, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 06), Bologna, Italy, p88-92, June 26-28, 2006.

As04 Ambler S.W., General Diagramming Guidelines, The Official Agile Modeling (AM) Site, 2004. Available from http://www.agilemodeling.com/style/general.htm

ASM02 Almond R.G., Steinberg L.S. and Mislevy R.J., Enhancing the Design and Delivery of Assessment Systems: A Four Process Architecture, Journal of Technology, Learning and Assessment 1(5), 2002. Available from http://www.jtla.org/

Asy94 Asymetrix Inc, Toolbook 3.0 User's Manual, Asymetrix Incorporated, 1994.

AUT05 AUT Research, Packing them in: The student-to-staff ratio in UK Higher Education, Association of University Teachers Research, October 2005. Available from http://www.aut.org.uk/media/pdf/c/j/ssr_packingthemin.pdf

Ba79 Borning A., Thinglab – A Constraint-Oriented Simulation Laboratory, Technical Report STAN-CS-79-746, Stanford University, USA, March, 1979.

Bb03 Bligh B., CourseMarker and DATsys: next generation automated assessment systems, talk delivered to Use of CAA in ICS Education, LTSN conference, University of Brighton, UK, July 16, 2003.

BBF+93 Benford S., Burke E., Foxley E., Gutteridge N. and Zin A.M., Experiences with the Ceilidh System, Proceedings of the 1st International Conference on Computer Based Learning in Science (CBLIS'93), Vienna, Austria, 1993.

BBF+95 Benford S., Burke E., Foxley E. and Higgins C., The Ceilidh System for the Automatic Grading of Students on Programming Courses, ACM Press, Proceedings of the 33rd Annual ACM Southeast Conference, Clemson, South Carolina, March 1995.

BBF96 Benford S., Burke E. and Foxley E., Developer's Guide to Ceilidh, LTR Report, Computer Science Department, The University of Nottingham, UK, 1996.

BC82 Biggs J.B. and Collis K.F., Evaluating the Quality of Learning: the SOLO Taxonomy, Academic Press, 1982.

BD04 Bull J. and Danson M., Computer-assisted Assessment (CAA), LTSN Generic Centre: Assessment Series No. 14, January 2004.

BE98 Blackwell A. and Engelhardt Y., A taxonomy of diagram taxonomies, Proceedings of Thinking With Diagrams 98: Is there a Science of Diagrams?, p60-70, 1998. Available from: http://www.cl.cam.ac.uk/~afb21/publications/TwD98.html

BEF+56 Bloom B.S., Englehart M.D., Furst E.J., Hill W.H. and Krathwohl, D.R., Taxonomy of educational objectives: the classification of educational goals, Handbook 1: Cognitive Domain, Longmans, Green, Co., 1956.

BEH+05 Burrow M., Evdorides H., Hallam B. and Freer-Hewish R., Developing formative assessments for postgraduate students in engineering, European Journal of Engineering Education 30(2), p255-263, May 2005.

BER+99 Baklavas G., Economides A.A. and Roumeliotis M., Evaluation and Comparison of Web-based testing tools, Proceedings of the World Conference on the WWW and Internet (WebNet-99), Association for the Advancement of Computing in Education, Honolulu HI, USA, October 25-28, 1999.

BET+94 Battista D., Eades P., Tamassia R., and Tollis I., Annotated bibliography on graph drawing algorithms, Computational Geometry: Theory and Applications, Vol 4, p235-282, 1994.

BFK+04 Belton M., Fair K., Kleeman J., Phaup J. and Shepherd E., Perception to Go: Empowering Disconnected Delivery of Assessments, Questionmark White Paper, 2004. Available from: http://www.questionmark.com/

Bg01 Brown G., Assessment: A Guide for Lecturers, LTSN Generic Centre: Assessment Series, November 2001.

Bg93 Booch G., Object-oriented Analysis and Design with Applications, 2nd Edition, Benjamin Cummings, 1993. ISBN 0-8053-5340-2.

BG97 Beck K. and Gamma E., Advanced Design with Patterns in Java, Object-Oriented Programming Systems, Languages and Applications (OOPSLA'97), Tutorial 30.

BGK+00 Bridgeman S., Goodrich M.T., Kobourov S.G. and Tamassia R., PILOT: An Interactive Tool for Learning and Grading, Proceedings of SIGCSE 2000, Austin, TX, USA, p139-143, March 8-12, 2000.

BGK+96 Broy M., Grosu R., Klein C. and Rumpe B., State Transition Diagrams, Technical Report TUM-I-9630, Technical University of Munchen, 1996.

BGL+97 Di Battista G., Garg A., Liotta G, Tamassia R., Tassinari E. and Vargiu F., An experimental comparison of four graph drawing algorithms, Computational Geometry 7(5-6), p303-325, April, 1997.

Bj93 Bull J., Using Technology to Assess Student Learning, TLTP Project Alter, December 1993, ISBN 1 85889 091 8.

BL06 Brusilovsky P. and Loboda T.D., WADEIn II: A Case for Adaptive Explanatory Visualization, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 06), Bologna, Italy, p48-52, June 26-28, 2006.

Br02 Bennett R.E., Inexorable and inevitable: The continuing story of technology and assessment, Journal of Technology, Learning and Assessment 1(1), 2002. Available from http://www.jtla.org/

BRS96 Brown S., Race P. and Smith B., 500 Tips on Assessment, Kogan Page, 1996, ISBN 0749419415.

BSP+03 Belton M., Shephard E., Phaup J., Fair K. and Kleeman J., Questionmark's Holistic Approach: Assessments systems for the enterprise, Questionmark White Paper, 2003. Available from: http://www.questionmark.com/

Bt00 Buchanan T., The efficacy of a World-Wide Web mediated formative assessment, Journal of Computer Assisted Learning 16(3), p193-200, 2000.

BW98 Black P. and William D., Assessment and classroom learning, Assessment in Education 5(1), p7-74, 1998.

CB98 Culverhouse P.F. and Burton C.J., Mastertutor: a tutorial shell for the support of problem solving skill acquisition, Bringing Information Technology to Education (BITE): Integrating Information & Communication Technology in Higher Education, Maastricht, p433-443, March 25-27, 1998. Available from: http://www.cis.plym.ac.uk/cis/publications/BITE1998.pdf

CDE+03 Carter J., Dick M., English J., Ala-Mukta K., Fone W., Fuller U., Sheard J., How Shall We Assess This?, Proceedings of the 8th Annual Joint Conference Integrating Technology into Computer Science Education, Thessaloniki, Greece, p107-123, June 30 to July 2, 2003. ISSN 0097-8418.

CE98a Charman D. and Elmes A., Computer Based Assessment (Volume 1): A guide to good practice, SEED Publications, University of Plymouth, 1998, ISBN 1-84102-024-9.

CE98b Charman D. and Elmes A., Computer Based Assessment (Volume 2): Case studies in Science and Computing, SEED Publications, University of Plymouth, 1998, ISBN 1-84102-02-7.

CE98c Charman D. and Elmes A., A Computer-based Formative Assessment Strategy for a Basic Statistics Module in Geography, Journal of Geography in Higher Education 22(3), p381-385, November, 1998.

Cf98 Culwin F., Web hosted assessment: possibilities and policy, Proceedings of the 6th Annual Conference on the Teaching of Computing/3rd Annual ITiCSE Conference on Changing the Delivery of Computer Science Education, p55–58, 1998.

CM03 Chok S.S. and Marriott K., Automatic generation of intelligent diagram editors, ACM Transactions on Computer-Human Interaction (TOCHI) 10(3), p244-276, September, 2003.

CO97 Chung G.K.W.K. and O'Neil H.F. Jnr, Methodological approaches to online scoring of essays, Document Reproduction Service, Educational Resources Information Center (ERIC), U.S. Department of Education Office of Educational Research and Improvement, ED-418-101, 1997.

CS96 Coleman M. and Stott Parker D., Aesthetics-based graph layout for human consumption, Software — Practice and Experience 26(12), p1415-1438, December, 1996. Available from: http://www.cs.ucla.edu/~stott/aglo/

CS98 Canup M. and Shackelford R., Using software to solve problems in large computing courses, Proceedings of the 29th SIGCSE, Technical Symposium on Computer Science Education, Atlanta GA, USA, February 26 to March 1, 1998, p135-139.

Cyg98 Cygwin.com, Cygwin User's Guide. Available from http://cygwin.com/cygwin-ug-net/

Dc99 Daly C., RoboProf and an Introductory Computer Programming Course, Proceedings of the 4th Annual SIGCSE / SIGCUE on Innovation and Technology in Computer Science Education, Krakow, Poland, p155-158, June 27-30, 1999.

Dd99 Dodson D., Diagrammatic Interaction, Tutorial, Computer Science Department, City University, London, 12 February, 1999.

Dfa04 Defense Finance and Accounting Service, Diagramming Guidelines, DFAS/DCII Development Standards and Guidelines, 2004.

Dfes05 Department for Education and Skills, Trends in Education and Skills, Learning & Skills Gateway, DfES website. Updated frequently, accessed on 3 January 2006. Available from http://www.dfes.gov.uk/trends/

DK01 Duke-Williams E. and King T., Testing Higher Learning Outcomes with CBA, Handbook of ILTAC 2001, The Institute for Learning and Teaching in Higher Education Conference: Professionalism in Practice, University of York, Session 42, 4-6 July, 2001. Available from: http://www.tech.port.ac.uk/%7Ekingt/research/ILT01/ILTAC01caa.html

DLO+05 Douce C., Livingstone D., Orwell J., Grindle S. and Cobb J., A Technical Perspective on ASAP – Automated System for Assessment of Programming, Proceedings of the 9th International Conference on Computer Aided Assessment, Loughborough, July, 2005. Available from: http://dircweb.king.ac.uk/Ris/Queries/Pages/home_page.asp?authorID=4

Dp03 Denton P., Returning Feedback to Students via Email Using Electronic Feedback 9, Learning and Teaching in Action 2(1), Manchester Metropolitan University: Learning and Teaching Unit, February 2003, ISSN 1477-1241. Available from http://www.ltu.mmu.ac.uk/ltia/

eb11 Encyclopaedia Britannica 11th Edition, Cambridge University Press, 1911.

EG03 Eichelberger H. and von Gudenberg J.W., UML Class Diagrams – State of the Art in Layout Techniques, Proceedings of the 2nd Annual "Designfest" on Visualizing Software for Understanding and Analysis (VISSOFT 2003), Amsterdam, Netherlands, September 22, 2003. Available from: http://www.cs.uvic.ca/~mstorey/vissoft2003/

Ej02 English J., Experience with a computer-assisted formal programming examination, ACM SIGCSE Bulletin, Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education, p51-54, Aarhus, Denmark, June 24-26, 2002.

Ej04 English J., Automated Assessment of GUI Programs using JEWL, Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 04), Leeds, UK, p137-141, June 25-27, 2001.

En01 Eaton N., Microsoft Visio Version 2002 Inside Out, Microsoft Press, June, 2001. ISBN 0735612854.

FHG96 Foxley E., Higgins C. and Gibbon C., The Ceilidh System: A general overview, LTR Report, Department of Computer Science, University of Nottingham, UK, 1996.

FHH+01 Foxley E., Higgins C., Hegazy T., Symeonidis P. and Tsintsifas A., The CourseMaster CBA System: Improvements over Ceilidh, Proceedings of the 5th Annual Computer Assisted Assessment Conference, Loughborough, UK, 2-4 July 2001, p189-201, ISBN 0-9539572-0-9.

FHS+01 Foxley E., Higgins C., Symeonidis P. and Tsintsifas A., The CourseMaster Automated Assessment System – a next generation Ceilidh, Computer Assisted Assessment Workshop, Warwick, UK, 5-6 April 2001.

FHT+99 Foxley E., Higgins C., Tsintsifas A. and Symeonidis P., Ceilidh: A System for the Automatic Evaluation of Student Programming Work, Proceedings of the 4th International Conference on Computer Based Learning in Science (CBLIS'99), University of Twente, Netherlands, July 2-6 1999.

Fj01 Foster J., Improved coursework assessment for undergraduate and postgraduate courses, Teaching and Learning Innovation Fund: Final Report, Department of Electronic Systems Engineering, University of Essex, UK, April 2001. Available from: http://www.essex.ac.uk/innovations/Foster.htm

FL94 Foxley E. and Lou B., A Simple Text Automatic Marking System, Artificial Intelligence and Simulation of Behaviour 94 Conference for Computational Linguistics for Speech and Handwriting Recognition Workshop, Leeds, UK, April 12, 1994.

FLM98 Frosini G., Lazzerini B., Marcelloni F., Performing automatic exams, Computers & Education 31, 1998.

FSZ97 Foxley E., Salman O. and Shukur Z., The automatic assessment of Z specifications, Working group reports and supplemental proceedings, Uppsala, Sweden, p129-131 June 1-5, 1997.

FW65 Forsythe G. and Wirth N., Automatic grading programs, Communications of ACM 8(5), May 1965, p275-278.

FWW00 Ferguson R., Hunter A. and Hardy C., MetaBuilder: The diagrammer's diagrammer, First International Conference on Theory and Applications of Diagrams, Lecture Notes in Artificial Intelligence, Springer Verlag, Edinburgh, p407-421, September, 2000.

Gc97 Gibbon C.A., Heuristics for Object-Oriented Design, Ph.D. thesis, University of Nottingham, October 1997.

GH06 Gray G.R. and Higgins C.A., An Introspective Approach to Marking Graphical User Interfaces, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 06), Bologna, Italy, p43-47, June 26-28, 2006.

GHJ+94 Gamma E., Helm R., Johnson R. and Vlissides J., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.

GJS97 Gosling J., Joy B. and Steele G., The Java Language Specification, Addison Wesley, 1997.

Gm00 Greenhow M., Setting objective tests in mathematics with QM Designer, MSOR Connections 0(1), p21-26, Learning Technology Support Network, February, 2000.

GMW88 Gamma E., Marty R. and Weinand A., ET++ — an object oriented application framework for C++, Proceedings of Object-Oriented Programming Systems, Languages and Applications (OOPSLA'88), San Diego CA, USA, p46-57, September 25-30, 1988.

GS95 Gaines B. and Shaw M., Concept maps as hypermedia components, Knowledge Science Institute, University of Calgary, 1995.

GV47 Goldstine H. and von Neuman J., Planning and Coding Problems for an Electronic Computing Instrument, Volume 1, Van Nostrand, 1947.

GW01 Lee G. and Weerakoon P., The role of computer aided assessment in health professional education: A comparison of student performance in computer based and paper and pen multiple choice tests, The Vice-Chancellor's Showcase of Scholarly Inquiry in Teaching and Learning, Institute for Teaching and Learning, The University of Sydney, Australia, 2001. Available from: http://www.itl.usyd.edu.au/itl/Showcase2001/

HB06 Higgins C.A. and Bligh B., Formative Computer Based Assessment in Diagram Based Domains, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 06), Bologna, Italy, p98-102, June 26-28, 2006.

Hd88 Harel D., On visual formalisms, Communications of the ACM 31(5), p514-530, May, 1988.

HGS+06 Higgins C.A., Gray G., Symeonidis P. and Tsintsifas A., Automated Assessment and Experiences of Teaching Programming, ACM Journal on Educational Resources in Computing (JERIC) 5, Special Issue on Automated Assessment of Programming Assignments, ISSN 1531-4278. To appear.

Hi88 Hirmanpour I., A student system development diagrammer, Proceedings of the 19th SIGCSE Technical Symposium on Computer Science Education, Atlanta GA, USA, p104-108, 25-26 February, 1988.

Hj59 Hollingsworth J., An educational program in computing, Communications of the ACM 2(8), August 1959, p6.

Hj60 Hollingsworth J., Automatic graders for programming classes, Communications of ACM 3(10), October 1960, p528-529.

HL98 Hoggarth G. and Lockyer M., An automated student diagram assessment system, Proceedings of the 6th Annual Conference on the Teaching of Computing / 3rd Annual Conference on Integrating Technology into Computer Science Education: Changing the Delivery of Computer Science Education, Dublin, Ireland, 18-21 August 1998, p122-124, ISSN 0097-8418.

HM93 Hyvönen J. and Malmi L., TRAKLA — A system fro teaching algorithms using email and a graphical editor, Proceedings of HYPERMEDIA in Vaasa'93, Vaasa, Finland, p141-147, 1993.

HRT+98 Hall M.J., Robinson D.J., Tucknott G. and Carlton T., A multimedia tutorial shell with qualitative assessment in biology, in Charman D. and Elmes A. (eds), Computer Based Assessment (Volume 2): Case studies in Science and Computing, p33-38, SEED Publications, University of Plymouth, 1998, ISBN 1-84102-02-7.

Hs90 Hekmatpour S., Templa and Graphica: A Generic Graphical Editor for the Macintosh, Prentice Hall, 1990.

HST02 Higgins C., Symeonidis P. and Tsintsifas T., The Marking System for CourseMaster, Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science, University of Aarhus, Denmark, June 24-26, 2002, p46-50. ISSN 1-58113-499-1.

Ht98 Hawkes T., An Experiment in Computer-Assisted Assessment, Interactions 2(3), Educational Technology Service, University of Warwick, 1998. Available from: http://www.warwick.ac.uk/ETS/interactions/vol2no3/index.htm

HW69 Hext J. and Winings J., An automatic grading scheme for simple programming exercises, Communications of ACM 12(5), May 1969, p272-275.

HW96 Harrslev V. and Wessel M., GenEd: an editor with generic semantics for formal reasoning about visual notations, IEEE Symposium on Visual Languages 1996, Boulder CO, USA, September 3-6, 1996. Available from: http://citeseer.ist.psu.edu/haarslev96gened.html

Ib84 Imrie B.W., In search of academic excellence: samples of experience, Proceedings of the Tenth International Conference on Improving University Experience, University of Maryland, University College, p160-183, 1984.

Ib95 Imrie B.W., Assessment for Learning: quality and taxonomies, Assessment & Evaluation in Higher Education, 20(2), p175-189, 1995.

ISO05 International Standards Organisation, ISO in brief: International Standards for a sustainable world, ISO, March, 2005. ISBN: 92-67-10401-2. Available from: http://www.iso.org/

Iy01 Inoue Y., Questionnaire Surveys: Four Survey Instruments in Educational Research, Educational Resources Information Center (ERIC), U.S. Department of Education Office of Educational Research and Improvement, 2001. Available from: http://eric.ed.gov/

JA00 Johnstone A.H. and Ambusaidi A., Fixed Response: What are we testing?, Chemistry Education: Research and Practice in Europe 1(3), p323-328, 2000.

JBR98 Jacobson I., Booch G. and Rumbaugh J., The Unified Software Development Process, Addison Wesley Longman, 1998. ISBN 0-201-57169-2.

Jd00 Jackson D., A semi-automated approach to online assessment, Proceedings of the 5th Annual SIGCSE / SIGCUE Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland, p164-167, 11-13 July, 2000.

JG04 Joy M. and Griffiths N., Online Submission of Coursework – a Technological Perspective, Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT2004), Joensuu, Finland, 2004, p430-434. Available from: http://www.dcs.warwick.ac.uk/research/edtech/

JL98 Joy M. and Luck M., Effective electronic marking for on-line assessment, Proceedings of the 6th Annual Conference on the Teaching of Computing / 3rd Annual Conference on Integrating Technology into Computer Science Education: Changing the Delivery of Computer Science Education, Dublin, Ireland, 18-21 August 1998, p134-138, ISSN 0097-8418.

JMM+04 Juwah C., Macfarlane-Dick D., Matthew B., Nicol D., Ross D. and Smith B., Enhancing student learning through effective formative feedback, The Higher Education Academy (Generic Centre), June, 2004. ISBN 1-904190-58-8.

JU97 Jackson D. and Usher M., Grading student programs using ASSYST, Proceedings of the 28th SIGCSE technical symposium on Computer Science Education, San Jose CA, USA, p335-339, February 27 to March 1, 1997.

KM00 Korhonen A. and Malmi L., Algorithm simulation with automatic assessment, Proceedings of the 5th Annual SIGCSE / SIGCUE Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland, p160-163, 11-13 July, 2000.

Kp01 Knight P., A Briefing on Key Concepts: Formative and summative, criterion and norm-referenced assessment, LTSN Generic Centre: Assessment Series, November 2001.

LBW+94 Lohse G., Biolski K., Walker N. and Rueter H., A classification of visual representations, Communications of the ACM 37(12), p36-49, 1994.

LD97 Landauer T.K. and Dumais S.T., A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge, Psychological Review 104, p211-240, 1997.

LHL98 Landauer T.K., Holtz P.W. and Laham D., Introduction to Latent Semantic Analysis, Discourse Processes 25, p259-284, 1998.

Lr32 Likert R., A Technique for the Measurement of Attitudes, Archives of Psychology 140, p55, 1932.

Mac95 Macromedia Inc, Macromedia Authorware 7.0. Available from: http://www.macromedia.com/software/authorware/

MB99 McKenna C. and Bull J., Designing effective objective test questions: an introductory workshop, CAA Centre, June 17, 1999. Available from: http://www.caacentre.ac.uk/dldocs/otghdout.pdf

Md99 Mackenzie D., Recent developments in the Tripartite Interactive Assessment Delivery System (TRIADS), Proceedings of the 3rd Annual Computer Assessment Conference, Loughborough, UK, June 16-17, 1999.

Mf04 McMartin F., MERLOT: A Model for User Involvement in Digital Library Design and Implementation, Journal of Digital Information 5(3), September 2004.
Available from: http://jodi.ecs.soton.ac.uk/Articles/v05/i03/McMartin/

MGH98 Mansouri F.Z., Gibbon C.A. and Higgins C.A., PRAM: PRolog Automatic Marker, Proceedings of the 6th Annual Conference on the Teaching of Computing / 3rd Annual Conference on Integrating Technology Into Computer Science Education: Changing the delivery of computer science education (ITiCSE 98), Dublin City University, Ireland, p166-170, 1998.

MK04 Malmi L. and Korhonen A., Automatic Feedback and Resubmissions as Learning Aid, Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 04), Joensuu, Finland, p186-190, August 30 to September 1, 2004.

Ml97 Markham L., Staff-Student Ratios in Commonwealth Countries, Commonwealth Higher Education Management Service (CHEMS) Paper 16, Commonwealth Higher Education Support Scheme, 1997. Available from http://www.acu.ac.uk/chems/

MLC03 Murphy A.J., Lockie R.G. and Coutts A. J., Kinematic determinants of early acceleration in field sport athletes, Journal of Sports Science and Medicine 2(4), p144-150, 2003.

Mm02 McAlpine M., Principles of Assessment, Bluepaper Number 1, CAA Centre, University of Luton, February, 2002. ISBN 1-904020-01-1. Available from: http://www.caacentre.ac.uk/dldocs/Bluepaper1.pdf

Mp95 Martin P., EQL International's Interactive Assessor for Windows reviewed, Monitor 6, CTI Computing, University of Ulster, 1995.

Mr86 Myers R., Computerized Grading of Freshman Chemistry Laboratory Experiments, Journal of Chemical Education 63, 1986, p507-509.

Mu94 von Matt U., Kassandra: The automatic grading system, Technical Report UMIACS-TR-94-59, Institute for Advanced Computer Studies, Department of Computer Science, University of Maryland, USA, 1994.

MW98 Mason D. and Woit D., Integrating technology into computer science examinations, Proceedings of the 29th SIGCSE, Technical Symposium on Computer Science Education, Atlanta GA, USA, February 26 to March 1, 1998, p140-144.

MW99 Mason D. and Woit D., Providing mark-up and feedback to students with online marking, Proceedings of the 30th SIGCSE, Technical Symposium on Computer Science Education, Atlanta GA, USA, March 24 to 28, 1998, p140-144.

NB01 Ngo D.C.L. and Byrne J.G., Another Look at a Model for Evaluating Interface Aesthetics, International Journal of Applied Mathematical Computer Science 11(2), p515-535, 2001.

ND02 Neven F. and Duval E., Reusable Learning Objects: a survey of LOM-based repositories, Proceedings of the 10th ACM international conference on Multimedia, Juan-les-Pins, France, p291-294, December 1-6, 2002.

NTB00 Ngo D.C.L., Teo L.S. and Byrne J.G., A Mathematical Theory of Interface Aesthetics, Visual Mathematics 2(4), Serbian Academy of Sciences and Arts: Mathematical Institute, 2000. Available from: http://www.mi.sanu.ac.yu/vismath/

Or98 Oliver R., Experiences of assessing programming assignments by computer, in Charman D. and Elmes A. (eds), Computer Based Assessment (Volume 2): Case studies in Science and Computing, p47-49, SEED Publications, University of Plymouth, 1998, ISBN 1-84102-02-7.

PAC02 Purchase H.C., Allder J. and Carrington D., Graph Layout Aesthetics in UML Diagrams: User Preferences, Journal of Graph Algorithms and Applications 6(3), p255-279, World Scientific Publishing, 2002. ISSN 1526-1719. Available from: http://www.cs.brown.edu/publications/jgaa/

PB98 Paul C.R.C. and Boyle A.P., Computer-based assessment in palaeontology, in Charman D. and Elmes A. (eds), Computer Based Assessment (Volume 2): Case studies in Science and Computing, p51-56, SEED Publications, University of Plymouth, 1998, ISBN 1-84102-02-7.

Pc65 Petri C., Kommunikation mit Automaten, Ph.D. thesis, Translation by Greene C.F., Supplement to Technical Report RADC-TR-65-337, Volume 1, Rome Labs, Griffiss Air Force Base, New York, USA, 1965.

Pc99 Power C., Designer — a logic diagram design tool, Proceedings of the 4th Annual SIGCSE / SIGCUE on Innovation and Technology in Computer Science Education, Krakow, Poland, p211, June 27-30, 1999.

Pe94 Page E.B., Computer grading of student prose: Using modern concepts and software, Journal of Experimental Education 62(2), p127-142, 1994.

Pm95 Petre M., Why looking isn't always seeing: readership skills and graphical programming, Communications of the ACM 38(6), p33-44, June, 1995.

PPK97 Page E.B., Poggio J.P. and Keith T.Z., Computer analysis of student essays: Finding trait differences in the student profile, Proceedings of the AERA / NCME Symposium on Grading Essays by Computer, 1997.

PT00 Papakostas A. and Tollis I., Efficient orthogonal drawings of high degree graphs, Algorithmica 26(1), p100-125, Springer-Verlag, January, 2000.

Qt99 Quatrani T., Visual Modeling with Rational Rose 2000 and UML, Addison-Wesley Professional, 1999. ISBN 0201699613.

Rc01 Rust C., A Briefing on Assessment of Large Groups, LTSN Generic Centre: Assessment Series, November 2001.

Rd87 Rowntree D., Assessing Students: How shall we know them?, London: Kogan Page, 1987.

RH83 Rottmann R.M. and Hudson H.T., Computer Grading as an Instructional Tool, Journal of College Science Teaching 12, 1983, p152-156.

RJE02 Rawles S., Joy M. and Evans M., Computer Assisted Assessment Review Exercise, LTSN Centre for Information and Computer Sciences, February 4, 2002.

RL02 Rudner L.M. and Liang T., Automated essay scoring using Bayes' theorem, Journal of Technology, Learning and Assessment 1(2), 2002. Available from http://www.jtla.org/

Rp01 Race P., A Briefing on Self, Peer and Group Assessment, LTSN Generic Centre: Assessment Series, November 2001.

RT81 Reingold E. and Tilford J., Tidier Drawings of Trees, IEEE Transactions on Software Engineering 7(2), p223-228, IEEE Computer Society Press, March, 1981.

Sa01 Stedile A., JMFGraph — A Modular Framework for Drawing Graphs in Java, M.Sc. Thesis, Institute for Information Processing and Computer Supported New Media (IICM), Graz University of Technology, Austria, November 18th, 2001. Available from: http://www.iicm.edu/thesis/

SC01 Stefani L. and Carroll J., A Briefing on Plagiarism, LTSN Generic Centre: Assessment Series, November 2001.

SHP+06 Spacco J., Hovemeyer D., Pugh W., Emad F., Hollingsworth J.K. and Padua-Perez N., Experiences with Marmoset: Designing and Using an Advanced Submission and Testing System for Programming Courses, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 06), Bologna, Italy, p13-17, June 26-28, 2006.

Sk02 Sugiyama K., Graph Drawing and Applications for Software and Knowledge Engineers, Series on Software Engineering and Knowledge Engineering Volume 11, World Scientific, March 2002, ISBN 981-02-4879-2.

SM97 Stephens D. and Mascia J., Results of a Survey into the use of Computer-Assisted Assessment in Institutions of Higher Education in the UK, DILS, Loughborough University, January 1997.

SMK01 Saikkonen R., Malmi L. and Korhonen A., Fully automatic assessment of programming exercises, Proceedings of the 6th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 01), Canterbury, UK, p133-136, June 25-27, 2001.

Sp02 Symeonidis P., Setting Up Exercises Within CourseMaster, LTR Report, School of Computer Science and IT, University of Nottingham, UK, 2002.

SP03 Stephens D. and Percik D., Constructing a Test Bank for Information Science based upon Bloom's Principles, Innovations in Teaching and Learning in Information and Computer Sciences (ITALICS) 2(1), July 2003.

SP04 Shneiderman B. and Plaisant C., Designing the User Interface (fourth edition), Addison Wesley, 2004. ISBN 0-321-19786-0.

Sp06 Symeonidis P., Automated Assessment of Java Programming Coursework for Computer Science Education, Ph.D. thesis (unpublished manuscript), University of Nottingham, February 2006.

STW04 Smith N., Thomas P.G. and Waugh K., Interpreting Imprecise Diagrams, Proceedings of the Third International Conference in the Theory and Application of Diagrams, Cambridge, UK, p239-241, March 22-24, 2004. Available from: http://mcs.open.ac.uk/ns938/publications/diagrams-04-poster.pdf

Ta02 Tsintsifas A., A Framework for the Computer Based Assessment of Diagram Based Coursework, Ph.D. thesis, University of Nottingham, March 2002. Available from http://www.cs.nott.ac.uk/~azt/research.htm

Tb93 Buzan T., The Maind Map Book: Radiant Thinking — the major evolution in human thought, BBC Publications, 1993.

TBF97 Tinoco L., Barnette D. and Fox E., Online evaluation in WWW-based courseware, Proceedings of the 28th SIGCSE technical symposium on Computer Science Education, San Jose CA, USA, p194-198, February 27 to March 1, 1997.

TD76 Taylor J. and Deever D., Constructed-response, computer-graded homework, American Journal of Physics 44(6), June 1976, p598-599.

Tek87 Tektronix Computer Research Laboratory, Semantic Drawing with HotDraw, Technical Report CR-87-34, April, 1987.

Tp04 Thomas P.G., Drawing Diagrams in an Online Examination, Technical Report 2004/14, Department of Computing, The Open University, Milton Keynes, UK, April 23, 2004. Available from: http://computing-reports.open.ac.uk/index.php/2004/200414

Tr87 Tamassia R., On embedding a graph in the grid with the minimum number of bends, SIAM Journal on Computing 16(3), p421-444, Society for Industrial and Applied Mathematics, June, 1987.

TTV00 Tamassia R., Tollis I.G. and Vitter J.S., A Parallel Algorithm for Planar Orthogonal Grid Drawings, Parallel Processing Letters, March 2000. Available from: http://www.cs.duke.edu/~jsv/Papers/catalog/node140.html

TWS05 Thomas P.G., Waugh K. and Smith N., Experiments in the Automatic Marking of ER-Diagrams, Proceedings of the 10th annual SIGCSE conference on Innovation and Technology in Computer Science Education (ITiCSE 05), Monte de Caparica, Portugal, p158-162, June 27-29, 2005.

TWS06 Thomas P., Waugh K. and Smith N., Using Patterns in the Automatic Marking of ER-Diagrams, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 06), Bologna, Italy, p83-87, June 26-28, 2006.

Vg95 Viehstaedt G., A Generator for Diagram Editors, Ph.D. thesis, University of Erlangen-Nürnberg, 1995.

VL89 Vlissides J. and Linton M., Unidraw: A Framework for Building Domain-Specific Graphical Editors, Technical Report CSL-TR-89-380, Stanford University, July, 1989.

VS02 Vendlinski T. and Stevens R., Assessing Student Problem-Solving Skills With Complex Computer-Based Tasks, Journal of Technology, Learning and Assessment 1(3), 2002. Available from http://www.jtla.org/

WC53 Watson J.D. and Crick F.H.C., A Structure for Deoxyribose Nucleic Acid, Nature 171, p737-738, April 25, 1953. Available from: http://www.nature.com/nature/dna50/watsoncrick.pdf

Wl98 Wybrew L., The use of computerised assessment in Health Science modules, in Charman D. and Elmes A. (eds), Computer Based Assessment (Volume 2): Case studies in Science and Computing, p61-65, SEED Publications, University of Plymouth, 1998, ISBN 1-84102-02-7.

Wt04 Winters T.D., Analysis, Design, Development, and Deployment of a Generalized Framework for Computer-Aided Assessment, M.Sc. thesis, University of California Riverside, June 2004.

Ym01 Yorke M., Assessment: A Guide for Senior Managers, LTSN Generic Centre: Assessment Series, November 2001.

ZF92 Zin A. M. and Foxley E., The "oracle" program, LTR Report, Dept. of Computer Science, University of Nottingham, UK, 1992. Available from: http://www.cs.nott.ac.uk/~ceilidh/papers.html

ZF94 Zin A.M. and Foxley E., Analyse: An Automatic Program Assessment System, Malaysian Journal of Computer Science 7, 1994. Available from: http://www.cs.nott.ac.uk/CourseMarker/more_info/html/ASQA.HTM